

Hosting Environment (Daemon)

Generated by Doxygen 1.6.1

Sat Feb 2 01:00:55 2013

Contents

1	ARC Compute Library (libarccompute)	1
1.1	ARC Compute Library (libarccompute)	1
1.1.1	Typedef Documentation	2
1.1.1.1	JobListRetriever	2
1.1.1.2	ServiceEndpointRetriever	2
1.1.1.3	TargetInformationRetriever	2
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Namespace Index	7
4.1	Namespace List	7
5	Data Structure Index	9
5.1	Class Hierarchy	9
6	Data Structure Index	19
6.1	Data Structures	19
7	File Index	29
7.1	File List	29
8	Module Documentation	33
8.1	Structures holding resource information	33
8.1.1	Detailed Description	33
8.2	JobDescription related classes	34
8.2.1	Detailed Description	34
8.3	ARC Compute Library (libarccompute)	35
8.3.1	Typedef Documentation	36

8.3.1.1	JobListRetriever	36
8.3.1.2	ServiceEndpointRetriever	36
8.3.1.3	TargetInformationRetriever	36
8.4	Plugin related classes for compute specialisations	37
8.5	Classes for controlling output of compute test plugins	38
8.5.1	Detailed Description	38
8.6	ARC data staging (libarcdatastaging)	39
8.6.1	Detailed Description	40
8.6.2	Typedef Documentation	41
8.6.2.1	DTR_ptr	41
8.6.2.2	DTRLogger	41
8.6.3	Enumeration Type Documentation	41
8.6.3.1	CacheState	41
8.6.3.2	ProcessState	42
8.6.3.3	StagingProcesses	42
9	Namespace Documentation	43
9.1	Arc Namespace Reference	43
9.1.1	Detailed Description	58
9.1.2	Typedef Documentation	58
9.1.2.1	AttrConstIter	58
9.1.2.2	AttrIter	59
9.1.2.3	AttrMap	59
9.1.2.4	get_plugin_instance	59
9.1.3	Enumeration Type Documentation	59
9.1.3.1	escape_type	59
9.1.3.2	LogFormat	59
9.1.3.3	LogLevel	59
9.1.3.4	PeriodBase	60
9.1.3.5	ServiceType	60
9.1.3.6	StatusKind	60
9.1.3.7	TimeFormat	61
9.1.3.8	WSAFault	61
9.1.4	Function Documentation	61
9.1.4.1	addVOMSAC	61
9.1.4.2	CanonicalDir	61
9.1.4.3	ContentFromPayload	61

9.1.4.4	CreateThreadFunction	62
9.1.4.5	createVOMSAC	62
9.1.4.6	DirCreate	62
9.1.4.7	DirDelete	62
9.1.4.8	DirDelete	62
9.1.4.9	EnvLockAcquire	62
9.1.4.10	EnvLockRelease	62
9.1.4.11	EnvLockUnwrap	63
9.1.4.12	EnvLockUnwrapComplete	63
9.1.4.13	EnvLockWrap	63
9.1.4.14	escape_chars	63
9.1.4.15	FileCopy	63
9.1.4.16	FileCreate	63
9.1.4.17	FileDelete	63
9.1.4.18	FileLink	64
9.1.4.19	FileRead	64
9.1.4.20	FileRead	64
9.1.4.21	FileReadLink	64
9.1.4.22	FileStat	64
9.1.4.23	final_xmlsec	64
9.1.4.24	get_cert_str	64
9.1.4.25	get_key_from_certfile	64
9.1.4.26	get_key_from_certstr	64
9.1.4.27	get_key_from_keyfile	65
9.1.4.28	get_key_from_keyst	65
9.1.4.29	get_node	65
9.1.4.30	getCredentialProperty	65
9.1.4.31	GUID	65
9.1.4.32	init_xmlsec	65
9.1.4.33	inttostr	65
9.1.4.34	inttostr	66
9.1.4.35	inttostr	66
9.1.4.36	inttostr	66
9.1.4.37	inttostr	66
9.1.4.38	inttostr	66
9.1.4.39	istring_to_level	66

9.1.4.40	load_key_from_certfile	67
9.1.4.41	load_key_from_certstr	67
9.1.4.42	load_key_from_keyfile	67
9.1.4.43	load_trusted_cert_file	67
9.1.4.44	load_trusted_cert_str	67
9.1.4.45	load_trusted_certs	67
9.1.4.46	OpenSSLInit	67
9.1.4.47	operator<<	67
9.1.4.48	parseVOMSAC	68
9.1.4.49	parseVOMSAC	68
9.1.4.50	parseVOMSAC	68
9.1.4.51	passphrase_callback	69
9.1.4.52	string	69
9.1.4.53	strtobool	69
9.1.4.54	strtoint	69
9.1.4.55	strtoint	69
9.1.4.56	strtoint	70
9.1.4.57	strtoint	70
9.1.4.58	strtoint	70
9.1.4.59	strtoint	70
9.1.4.60	TmpDirCreate	70
9.1.4.61	TmpFileCreate	70
9.1.4.62	uri_encode	70
9.1.4.63	VOMSACSeqEncode	71
9.1.4.64	VOMSACSeqEncode	71
9.1.4.65	VOMSDecode	71
9.1.4.66	VOMSEncode	71
9.1.4.67	WSAFaultAssign	71
9.1.4.68	WSAFaultExtract	71
9.1.5	Variable Documentation	71
9.1.5.1	CredentialLogger	71
9.1.5.2	plugins_table_name	71
9.1.5.3	Version	72
9.2	ArcCredential Namespace Reference	73
9.2.1	Detailed Description	73
9.2.2	Enumeration Type Documentation	73

9.2.2.1	certType	73
9.3	DataStaging Namespace Reference	75
9.3.1	Detailed Description	76
10	Data Structure Documentation	77
10.1	ArcCredential::ACACI Struct Reference	77
10.2	ArcCredential::ACATTHOLDER Struct Reference	78
10.3	ArcCredential::ACATTR Struct Reference	79
10.4	ArcCredential::ACATTRIBUTE Struct Reference	80
10.5	ArcCredential::ACC Struct Reference	81
10.6	ArcCredential::ACCERTS Struct Reference	82
10.7	ArcCredential::ACDIGEST Struct Reference	83
10.8	ArcCredential::ACFORM Struct Reference	84
10.9	ArcCredential::ACFULLATTRIBUTES Struct Reference	85
10.10	ArcCredential::ACHOLDER Struct Reference	86
10.11	ArcCredential::ACIETFATTR Struct Reference	87
10.12	ArcCredential::ACINFO Struct Reference	88
10.13	ArcCredential::ACIS Struct Reference	89
10.14	ArcCredential::ACSEQ Struct Reference	90
10.15	ArcCredential::ACTARGET Struct Reference	91
10.16	ArcCredential::ACTARGETS Struct Reference	92
10.17	ArcCredential::ACVAL Struct Reference	93
10.18	Arc::Adler32Sum Class Reference	94
10.18.1	Detailed Description	94
10.18.2	Member Function Documentation	94
10.18.2.1	add	94
10.18.2.2	end	94
10.18.2.3	print	95
10.18.2.4	scan	95
10.18.2.5	start	95
10.19	Arc::AdminDomainAttributes Class Reference	96
10.20	Arc::AdminDomainType Class Reference	97
10.21	ArcSec::AlgFactory Class Reference	98
10.21.1	Detailed Description	98
10.21.2	Member Function Documentation	98
10.21.2.1	createAlg	98
10.22	ArcSec::AnyURIAttribute Class Reference	99

10.22.1 Member Function Documentation	99
10.22.1.1 encode	99
10.22.1.2 getId	99
10.22.1.3 getType	99
10.23 Arc::ApplicationEnvironment Class Reference	100
10.23.1 Detailed Description	100
10.24 Arc::ApplicationType Class Reference	101
10.24.1 Field Documentation	101
10.24.1.1 Error	101
10.24.1.2 Executable	101
10.24.1.3 Input	101
10.24.1.4 LogDir	101
10.24.1.5 Output	101
10.24.1.6 PostExecutable	101
10.24.1.7 PreExecutable	102
10.24.1.8 RemoteLogging	102
10.25 Arc::ArcLocation Class Reference	103
10.25.1 Detailed Description	103
10.25.2 Member Function Documentation	103
10.25.2.1 GetPlugins	103
10.25.2.2 Init	103
10.26 ArcSec::ArcPeriod Struct Reference	104
10.27 Arc::ARCPolicyHandlerConfig Class Reference	105
10.28 Arc::ArcVersion Class Reference	106
10.28.1 Detailed Description	106
10.29 ArcSec::Attr Struct Reference	107
10.29.1 Detailed Description	107
10.30 ArcSec::AttributeFactory Class Reference	108
10.30.1 Detailed Description	108
10.31 Arc::AttributeIterator Class Reference	109
10.31.1 Detailed Description	109
10.31.2 Constructor & Destructor Documentation	109
10.31.2.1 AttributeIterator	109
10.31.2.2 AttributeIterator	110
10.31.3 Member Function Documentation	110
10.31.3.1 hasMore	110

10.31.3.2 key	110
10.31.3.3 operator*	110
10.31.3.4 operator++	110
10.31.3.5 operator++	110
10.31.3.6 operator->	111
10.31.4 Friends And Related Function Documentation	111
10.31.4.1 MessageAttributes	111
10.31.5 Field Documentation	111
10.31.5.1 current_	111
10.31.5.2 end_	111
10.32ArcSec::AttributeProxy Class Reference	112
10.32.1 Detailed Description	112
10.32.2 Member Function Documentation	112
10.32.2.1 getAttribute	112
10.33ArcSec::AttributeValue Class Reference	113
10.33.1 Detailed Description	113
10.33.2 Member Function Documentation	114
10.33.2.1 encode	114
10.33.2.2 equal	114
10.33.2.3 getId	114
10.33.2.4 getType	114
10.34ArcSec::Attrrs Class Reference	115
10.34.1 Detailed Description	115
10.35ArcSec::AuthzRequest Struct Reference	116
10.36ArcSec::AuthzRequestSection Struct Reference	117
10.36.1 Detailed Description	117
10.37Arc::AutoPointer< T > Class Template Reference	118
10.37.1 Detailed Description	118
10.38Arc::Base64 Class Reference	119
10.38.1 Detailed Description	119
10.39Arc::BaseConfig Class Reference	120
10.39.1 Detailed Description	120
10.39.2 Member Function Documentation	120
10.39.2.1 MakeConfig	120
10.40ArcSec::BooleanAttribute Class Reference	122
10.40.1 Member Function Documentation	122

10.40.1.1 encode	122
10.40.1.2 getId	122
10.40.1.3 getType	122
10.41 Arc::Broker Class Reference	123
10.42 Arc::BrokerPlugin Class Reference	124
10.43 Arc::BrokerPluginArgument Class Reference	125
10.44 Arc::BrokerPluginLoader Class Reference	126
10.45 Arc::BrokerPluginTestACCCControl Class Reference	127
10.46 ArcCredential::cert_verify_context Struct Reference	128
10.47 Arc::CertEnvLocker Class Reference	129
10.47.1 Detailed Description	129
10.48 AuthN::certInfo Struct Reference	130
10.49 Arc::ChainContext Class Reference	131
10.49.1 Detailed Description	131
10.49.2 Member Function Documentation	131
10.49.2.1 operator PluginsFactory *	131
10.50 Arc::Checksum Class Reference	132
10.50.1 Detailed Description	132
10.50.2 Member Function Documentation	132
10.50.2.1 add	132
10.50.2.2 end	133
10.50.2.3 print	133
10.50.2.4 scan	133
10.50.2.5 start	133
10.51 Arc::ChecksumAny Class Reference	135
10.51.1 Detailed Description	135
10.51.2 Member Enumeration Documentation	135
10.51.2.1 type	135
10.51.3 Member Function Documentation	136
10.51.3.1 add	136
10.51.3.2 end	136
10.51.3.3 FileChecksum	136
10.51.3.4 print	137
10.51.3.5 scan	137
10.51.3.6 start	137
10.52 Arc::CStringValue Class Reference	138

10.52.1 Detailed Description	138
10.52.2 Constructor & Destructor Documentation	138
10.52.2.1 CStringValue	138
10.52.2.2 CStringValue	138
10.52.2.3 CStringValue	138
10.52.3 Member Function Documentation	138
10.52.3.1 equal	138
10.52.3.2 operator bool	139
10.53Arc::ClassLoader Class Reference	140
10.54Arc::ClassLoaderPluginArgument Class Reference	141
10.55Arc::ClientHTTP Class Reference	142
10.55.1 Detailed Description	142
10.55.2 Member Function Documentation	142
10.55.2.1 GetEntry	142
10.55.2.2 Load	142
10.56Arc::ClientHTTPAttributes Class Reference	143
10.56.1 Detailed Description	143
10.57Arc::ClientHTTPwithSAML2SSO Class Reference	144
10.57.1 Constructor & Destructor Documentation	144
10.57.1.1 ClientHTTPwithSAML2SSO	144
10.57.2 Member Function Documentation	144
10.57.2.1 process	144
10.58Arc::ClientInterface Class Reference	145
10.58.1 Detailed Description	145
10.58.2 Member Function Documentation	145
10.58.2.1 Load	145
10.59Arc::ClientSOAP Class Reference	146
10.59.1 Detailed Description	146
10.59.2 Constructor & Destructor Documentation	146
10.59.2.1 ClientSOAP	146
10.59.3 Member Function Documentation	146
10.59.3.1 AddSecHandler	146
10.59.3.2 GetEntry	146
10.59.3.3 Load	147
10.59.3.4 process	147
10.59.3.5 process	147

10.60Arc::ClientSOAPwithSAML2SSO Class Reference	148
10.60.1 Constructor & Destructor Documentation	148
10.60.1.1 ClientSOAPwithSAML2SSO	148
10.60.2 Member Function Documentation	148
10.60.2.1 process	148
10.60.2.2 process	148
10.61Arc::ClientTCP Class Reference	149
10.61.1 Detailed Description	149
10.61.2 Member Function Documentation	149
10.61.2.1 GetEntry	149
10.61.2.2 Load	149
10.62Arc::ClientX509Delegation Class Reference	150
10.62.1 Constructor & Destructor Documentation	150
10.62.1.1 ClientX509Delegation	150
10.62.2 Member Function Documentation	150
10.62.2.1 acquireDelegation	150
10.62.2.2 createDelegation	150
10.63ArcSec::CombiningAlg Class Reference	152
10.63.1 Detailed Description	152
10.63.2 Member Function Documentation	152
10.63.2.1 combine	152
10.63.2.2 getalgId	152
10.64Arc::EntityRetriever< T >::Common Class Reference	153
10.65Arc::ComputingEndpointAttributes Class Reference	154
10.66Arc::ComputingEndpointType Class Reference	155
10.67Arc::ComputingManagerAttributes Class Reference	156
10.68Arc::ComputingManagerType Class Reference	157
10.68.1 Field Documentation	157
10.68.1.1 ApplicationEnvironments	157
10.69Arc::ComputingServiceAttributes Class Reference	158
10.70Arc::ComputingServiceRetriever Class Reference	159
10.70.1 Detailed Description	159
10.70.2 Constructor & Destructor Documentation	159
10.70.2.1 ComputingServiceRetriever	159
10.70.3 Member Function Documentation	160
10.70.3.1 addConsumer	160

10.70.3.2 addEndpoint	160
10.70.3.3 addEntity	160
10.70.3.4 getAllStatuses	161
10.70.3.5 GetExecutionTargets	161
10.70.3.6 removeConsumer	161
10.70.3.7 wait	161
10.71Arc::ComputingServiceType Class Reference	162
10.72Arc::ComputingServiceUniq Class Reference	163
10.72.1 Member Function Documentation	163
10.72.1.1 addEntity	163
10.73Arc::ComputingShareAttributes Class Reference	164
10.73.1 Field Documentation	164
10.73.1.1 FreeSlotsWithDuration	164
10.73.1.2 MaxDiskSpace	164
10.73.1.3 MaxMainMemory	164
10.73.1.4 MaxVirtualMemory	164
10.73.1.5 Name	164
10.74Arc::ComputingShareType Class Reference	165
10.75Arc::Config Class Reference	166
10.75.1 Detailed Description	166
10.75.2 Constructor & Destructor Documentation	166
10.75.2.1 Config	166
10.75.2.2 Config	166
10.75.3 Member Function Documentation	167
10.75.3.1 print	167
10.76Arc::ConfigEndpoint Class Reference	168
10.76.1 Detailed Description	168
10.76.2 Member Enumeration Documentation	168
10.76.2.1 Type	168
10.76.3 Constructor & Destructor Documentation	169
10.76.3.1 ConfigEndpoint	169
10.76.4 Field Documentation	169
10.76.4.1 RequestedSubmissionInterfaceName	169
10.77Arc::ConfusaCertHandler Class Reference	170
10.77.1 Detailed Description	170
10.77.2 Constructor & Destructor Documentation	170

10.77.2.1 ConfusaCertHandler	170
10.77.3 Member Function Documentation	170
10.77.3.1 createCertRequest	170
10.77.3.2 getCertRequestB64	170
10.78Arc::ConfusaParserUtils Class Reference	171
10.78.1 Detailed Description	171
10.78.2 Member Function Documentation	171
10.78.2.1 destroy_doc	171
10.78.2.2 evaluate_path	171
10.78.2.3 extract_body_information	171
10.78.2.4 get_doc	171
10.78.2.5 handle_redirect_step	172
10.78.2.6 urlencode	172
10.78.2.7 urlencode_params	172
10.79Arc::CountedPointer< T > Class Template Reference	173
10.79.1 Detailed Description	173
10.80Arc::Counter Class Reference	174
10.80.1 Detailed Description	175
10.80.2 Member Typedef Documentation	176
10.80.2.1 IDType	176
10.80.3 Constructor & Destructor Documentation	176
10.80.3.1 Counter	176
10.80.3.2 ~Counter	176
10.80.4 Member Function Documentation	176
10.80.4.1 cancel	176
10.80.4.2 changeExcess	176
10.80.4.3 changeLimit	177
10.80.4.4 extend	177
10.80.4.5 getCounterTicket	177
10.80.4.6 getCurrentTime	178
10.80.4.7 getExcess	178
10.80.4.8 getExpirationReminder	178
10.80.4.9 getExpiryTime	178
10.80.4.10getLimit	178
10.80.4.11getValue	179
10.80.4.12reserve	179

10.80.4.13	setExcess	179
10.80.4.14	setLimit	180
10.81	Arc::CounterTicket Class Reference	181
10.81.1	Detailed Description	181
10.81.2	Constructor & Destructor Documentation	181
10.81.2.1	CounterTicket	181
10.81.3	Member Function Documentation	181
10.81.3.1	cancel	181
10.81.3.2	extend	182
10.81.3.3	isValid	182
10.82	Arc::CRC32Sum Class Reference	183
10.82.1	Detailed Description	183
10.82.2	Member Function Documentation	183
10.82.2.1	add	183
10.82.2.2	end	183
10.82.2.3	print	184
10.82.2.4	scan	184
10.82.2.5	start	184
10.83	Arc::Credential Class Reference	185
10.83.1	Detailed Description	186
10.83.2	Constructor & Destructor Documentation	186
10.83.2.1	Credential	186
10.83.2.2	Credential	186
10.83.2.3	Credential	187
10.83.2.4	Credential	187
10.83.2.5	Credential	187
10.83.2.6	Credential	187
10.83.3	Member Function Documentation	188
10.83.3.1	AddCertExtObj	188
10.83.3.2	AddExtension	188
10.83.3.3	AddExtension	188
10.83.3.4	GenerateEECRequest	188
10.83.3.5	GenerateEECRequest	188
10.83.3.6	GenerateEECRequest	188
10.83.3.7	GenerateRequest	189
10.83.3.8	GenerateRequest	189

10.83.3.9 GenerateRequest	189
10.83.3.10GetCAName	189
10.83.3.11GetCert	189
10.83.3.12GetCertNumofChain	189
10.83.3.13GetCertReq	189
10.83.3.14GetDN	189
10.83.3.15GetEndTime	189
10.83.3.16GetExtension	189
10.83.3.17getFormat_BIO	190
10.83.3.18GetIdentityName	190
10.83.3.19GetIssuerName	190
10.83.3.20GetLifeTime	190
10.83.3.21GetPrivKey	190
10.83.3.22GetProxyPolicy	190
10.83.3.23GetPubKey	190
10.83.3.24GetStartTime	190
10.83.3.25GetType	190
10.83.3.26GetVerification	190
10.83.3.27InitProxyCertInfo	191
10.83.3.28InquireRequest	191
10.83.3.29InquireRequest	191
10.83.3.30InquireRequest	191
10.83.3.31IsCredentialsValid	191
10.83.3.32IsValid	191
10.83.3.33LogError	191
10.83.3.34OutputCertificate	191
10.83.3.35OutputCertificateChain	192
10.83.3.36OutputPrivatekey	192
10.83.3.37OutputPublickey	192
10.83.3.38SelfSignEECRequest	192
10.83.3.39SetLifeTime	192
10.83.3.40SetProxyPolicy	192
10.83.3.41SetStartTime	192
10.83.3.42SignEECRequest	193
10.83.3.43SignEECRequest	193
10.83.3.44SignEECRequest	193

10.83.3.45SignRequest	193
10.83.3.46SignRequest	193
10.83.3.47SignRequest	193
10.83.3.48STACK_OF	193
10.84Arc::CredentialError Class Reference	194
10.84.1 Detailed Description	194
10.84.2 Constructor & Destructor Documentation	194
10.84.2.1 CredentialError	194
10.85Arc::CredentialStore Class Reference	195
10.85.1 Detailed Description	195
10.86Arc::Database Class Reference	196
10.86.1 Detailed Description	196
10.86.2 Member Function Documentation	196
10.86.2.1 connect	196
10.86.2.2 enable_ssl	196
10.87DataStaging::DataDelivery Class Reference	198
10.87.1 Detailed Description	198
10.87.2 Member Function Documentation	198
10.87.2.1 receiveDTR	198
10.88DataStaging::DataDeliveryComm Class Reference	199
10.88.1 Detailed Description	200
10.88.2 Member Enumeration Documentation	200
10.88.2.1 CommStatusType	200
10.88.3 Constructor & Destructor Documentation	200
10.88.3.1 DataDeliveryComm	200
10.88.4 Member Function Documentation	201
10.88.4.1 CheckComm	201
10.88.4.2 PullStatus	201
10.89DataStaging::DataDeliveryCommHandler Class Reference	202
10.89.1 Detailed Description	202
10.90DataStaging::DataDeliveryLocalComm Class Reference	203
10.90.1 Detailed Description	203
10.91DataStaging::DataDeliveryRemoteComm Class Reference	204
10.91.1 Detailed Description	204
10.92Arc::DataStagingType Class Reference	205
10.93ArcSec::DateAttribute Class Reference	206

10.93.1 Member Function Documentation	206
10.93.1.1 encode	206
10.93.1.2 getId	206
10.93.1.3 getType	206
10.94ArcSec::DateTimeAttribute Class Reference	207
10.94.1 Detailed Description	207
10.94.2 Member Function Documentation	207
10.94.2.1 encode	207
10.94.2.2 getId	207
10.94.2.3 getType	207
10.95Arc::DelegationConsumer Class Reference	208
10.95.1 Detailed Description	208
10.95.2 Constructor & Destructor Documentation	208
10.95.2.1 DelegationConsumer	208
10.95.2.2 DelegationConsumer	208
10.95.3 Member Function Documentation	208
10.95.3.1 Acquire	208
10.95.3.2 Acquire	209
10.95.3.3 Backup	209
10.95.3.4 Generate	209
10.95.3.5 ID	209
10.95.3.6 LogError	209
10.95.3.7 Request	209
10.95.3.8 Restore	209
10.96Arc::DelegationConsumerSOAP Class Reference	210
10.96.1 Detailed Description	210
10.96.2 Constructor & Destructor Documentation	210
10.96.2.1 DelegationConsumerSOAP	210
10.96.2.2 DelegationConsumerSOAP	210
10.96.3 Member Function Documentation	210
10.96.3.1 DelegateCredentialsInit	210
10.96.3.2 DelegatedToken	211
10.96.3.3 UpdateCredentials	211
10.96.3.4 UpdateCredentials	211
10.97Arc::DelegationContainerSOAP Class Reference	212
10.97.1 Detailed Description	212

10.97.2 Member Function Documentation	212
10.97.2.1 AddConsumer	212
10.97.2.2 CheckConsumers	213
10.97.2.3 DelegateCredentialsInit	213
10.97.2.4 DelegatedToken	213
10.97.2.5 FindConsumer	213
10.97.2.6 GetFailure	213
10.97.2.7 MatchNamespace	213
10.97.2.8 QueryConsumer	213
10.97.2.9 ReleaseConsumer	213
10.97.2.10 RemoveConsumer	213
10.97.2.11 TouchConsumer	214
10.97.2.12 UpdateCredentials	214
10.97.3 Field Documentation	214
10.97.3.1 context_lock_	214
10.97.3.2 max_duration_	214
10.97.3.3 max_size_	214
10.97.3.4 max_usage_	214
10.98 Arc::DelegationProvider Class Reference	215
10.98.1 Detailed Description	215
10.98.2 Constructor & Destructor Documentation	215
10.98.2.1 DelegationProvider	215
10.98.2.2 DelegationProvider	215
10.98.3 Member Function Documentation	215
10.98.3.1 Delegate	215
10.99 Arc::DelegationProviderSOAP Class Reference	216
10.99.1 Detailed Description	216
10.99.2 Constructor & Destructor Documentation	216
10.99.2.1 DelegationProviderSOAP	216
10.99.2.2 DelegationProviderSOAP	216
10.99.3 Member Function Documentation	217
10.99.3.1 DelegateCredentialsInit	217
10.99.3.2 DelegateCredentialsInit	217
10.99.3.3 DelegatedToken	217
10.99.3.4 ID	217
10.99.3.5 UpdateCredentials	217

10.99.3.6 UpdateCredentials	217
10.100ArcSec::DenyOverridesCombiningAlg Class Reference	218
10.100.Detailed Description	218
10.100.Member Function Documentation	218
10.100.2.lcombine	218
10.100.2.2getalgId	218
10.100Arc::DiskSpaceRequirementType Class Reference	220
10.101.Field Documentation	220
10.101.1.ICacheDiskSpace	220
10.101.1.2DiskSpace	220
10.101.1.3SessionDiskSpace	220
10.100Arc::PluginsFactory::modules_t::diterator Class Reference	221
10.100Arc::DNListHandlerConfig Class Reference	222
10.100DataStaging::DTR Class Reference	223
10.104.Detailed Description	225
10.104.Constructor & Destructor Documentation	225
10.104.2.IDTR	225
10.104.Member Function Documentation	225
10.104.3.ladd_problematic_delivery_service	225
10.104.3.2registerCallback	225
10.104.3.3reset	225
10.104.3.4set_error_status	226
10.100DataStaging::DTRCacheParameters Class Reference	227
10.105.Detailed Description	227
10.100DataStaging::DTRCallback Class Reference	228
10.106.Detailed Description	228
10.106.Member Function Documentation	228
10.106.2.lreceiveDTR	228
10.100DataStaging::DTRErrorStatus Class Reference	229
10.107.Detailed Description	229
10.107.Member Enumeration Documentation	229
10.107.2.IDTRErrorLocation	229
10.107.2.2DTRErrorStatusType	230
10.107.Constructor & Destructor Documentation	230
10.107.3.IDTRErrorStatus	230
10.100DataStaging::DTRLList Class Reference	231

10.108. Detailed Description	231
10.108. Member Function Documentation	231
10.108.2.1 dumpState	231
10.108.2.2 filter_dtrs_by_job	231
10.108.2.3 filter_dtrs_by_next_receiver	232
10.108.2.4 filter_dtrs_by_owner	232
10.108.2.5 filter_dtrs_by_status	232
10.108.2.6 filter_dtrs_by_statuses	232
10.108.2.7 filter_dtrs_by_statuses	233
10.108.2.8 filter_pending_dtrs	233
10.109. DataStaging::DTRStatus Class Reference	234
10.109. Detailed Description	234
10.109. Member Enumeration Documentation	235
10.109.2. IDTRStatusType	235
10.110. ArcSec::DurationAttribute Class Reference	236
10.110. Detailed Description	236
10.110. Member Function Documentation	236
10.110.2.1 encode	236
10.110.2.2 getId	236
10.110.2.3 getType	236
10.111. Arc::Endpoint Class Reference	237
10.111. Detailed Description	237
10.111. Member Enumeration Documentation	238
10.111.2. ICapabilityEnum	238
10.111. Constructor & Destructor Documentation	238
10.111.3.1 Endpoint	238
10.111.3.2 Endpoint	238
10.111.3.3 Endpoint	239
10.111.3.4 Endpoint	239
10.111.3.5 Endpoint	239
10.111. Member Function Documentation	239
10.111.4.1 getServiceName	239
10.111.4.2 GetStringForCapability	239
10.111.4.3 HasCapability	239
10.111.4.4 HasCapability	240
10.111.4.5 operator<	240

10.111.4.6operator=	240
10.111.4.7str	240
10.111.5Field Documentation	240
10.111.5.1Capability	240
10.111.5.2HealthState	240
10.111.5.3HealthStateInfo	240
10.111.5.4InterfaceName	240
10.111.5.5QualityLevel	241
10.111.5.6RequestedSubmissionInterfaceName	241
10.111.5.7ServiceID	241
10.111.5.8URLString	241
10.111Arc::EndpointQueryingStatus Class Reference	242
10.112.Detailed Description	242
10.112.2Member Enumeration Documentation	242
10.112.2.1EndpointQueryingStatusType	242
10.112.3Constructor & Destructor Documentation	243
10.112.3.1EndpointQueryingStatus	243
10.112.4Member Function Documentation	243
10.112.4.1getDescription	243
10.112.4.2getStatus	243
10.112.4.3operator bool	243
10.112.4.4operator!	243
10.112.4.5operator!=	244
10.112.4.6operator!=	244
10.112.4.7operator=	244
10.112.4.8operator=	244
10.112.4.9operator==	244
10.112.4.10operator==	244
10.112.4.11str	245
10.112.4.12str	245
10.111Arc::EndpointQueryOptions< T > Class Template Reference	246
10.113.Detailed Description	246
10.113.3Constructor & Destructor Documentation	246
10.113.2.1EndpointQueryOptions	246
10.111Arc::EndpointQueryOptions< Endpoint > Class Template Reference	247
10.114.Detailed Description	247

10.114. Constructor & Destructor Documentation	247
10.114.2. IEndpointQueryOptions	247
10.115. Arc::EndpointStatusMap Class Reference	248
10.116. Arc::EndpointSubmissionStatus Class Reference	249
10.116. Member Enumeration Documentation	249
10.116.1. IEndpointSubmissionStatusType	249
10.116. Constructor & Destructor Documentation	249
10.116.2. IEndpointSubmissionStatus	249
10.116.3. Member Function Documentation	249
10.116.3.1. getDescription	249
10.116.3.2. getStatus	250
10.116.3.3. operator bool	250
10.116.3.4. operator !	250
10.116.3.5. operator !=	250
10.116.3.6. operator !=	250
10.116.3.7. operator =	250
10.116.3.8. operator =	250
10.116.3.9. operator ==	251
10.116.3.10. operator ==	251
10.116.3.11. lstr	251
10.116.3.12. lstr	251
10.117. Arc::EntityConsumer< T > Class Template Reference	252
10.117. Detailed Description	252
10.117. Member Function Documentation	252
10.117.2. laddEntity	252
10.118. Arc::EntityContainer< T > Class Template Reference	253
10.118. Detailed Description	253
10.118. Member Function Documentation	253
10.118.2. laddEntity	253
10.119. Arc::EntityRetriever< T > Class Template Reference	254
10.119. Detailed Description	254
10.119. Constructor & Destructor Documentation	256
10.119.2. IEntityRetriever	256
10.119. Member Function Documentation	256
10.119.3. laddConsumer	256
10.119.3.2. addEndpoint	256

10.119.3.3addEntity	256
10.119.3.4clearEndpointStatuses	257
10.119.3.5getAllStatuses	257
10.119.3.6getServicesWithStatus	257
10.119.3.7getStatusOfEndpoint	257
10.119.3.8sDone	258
10.119.3.9removeConsumer	258
10.119.3.10moveEndpoint	258
10.119.3.11setStatusOfEndpoint	258
10.119.3.12wait	259
10.120Arc::EntityRetrieverPlugin< T > Class Template Reference	260
10.120Arc::EntityRetrieverPluginLoader< T > Class Template Reference	261
10.120Arc::EnvLockWrapper Class Reference	262
10.122.Detailed Description	262
10.122.Constructor & Destructor Documentation	262
10.122.2.IEnvLockWrapper	262
10.120ArcSec::EqualFunction Class Reference	263
10.123.Detailed Description	263
10.123.Member Function Documentation	263
10.123.2.1evaluate	263
10.123.2.2evaluate	263
10.123.2.3getFunctionName	263
10.120ArcSec::EvalResult Struct Reference	264
10.124.Detailed Description	264
10.120ArcSec::EvaluationCtx Class Reference	265
10.125.Detailed Description	265
10.125.Constructor & Destructor Documentation	265
10.125.2.IEvaluationCtx	265
10.120ArcSec::Evaluator Class Reference	266
10.126.Detailed Description	266
10.126.Member Function Documentation	266
10.126.2.1addPolicy	266
10.126.2.2addPolicy	266
10.126.2.3evaluate	267
10.126.2.4evaluate	267
10.126.2.5evaluate	267

10.126.2.6evaluate	267
10.126.2.7evaluate	267
10.126.2.8evaluate	267
10.126.2.9evaluate	267
10.126.2.10getAlgFactory	267
10.126.2.11getAttrFactory	268
10.126.2.12getFnFactory	268
10.126.2.13getName	268
10.126.2.14setCombiningAlg	268
10.126.2.15setCombiningAlg	268
10.127ArcSec::EvaluatorContext Class Reference	269
10.127.Detailed Description	269
10.127.Member Function Documentation	269
10.127.2.1operator AlgFactory *	269
10.127.2.2operator AttributeFactory *	269
10.127.2.3operator FnFactory *	269
10.128ArcSec::EvaluatorLoader Class Reference	270
10.128.Detailed Description	270
10.128.Member Function Documentation	270
10.128.2.1getEvaluator	270
10.128.2.2getEvaluator	270
10.128.2.3getEvaluator	270
10.128.2.4getPolicy	270
10.128.2.5getPolicy	270
10.128.2.6getRequest	270
10.128.2.7getRequest	271
10.129Arc::ExecutableType Class Reference	272
10.129.Detailed Description	272
10.129.Field Documentation	272
10.129.2.1Argument	272
10.129.2.2Path	272
10.129.2.3SuccessExitCode	272
10.130Arc::ExecutionEnvironmentAttributes Class Reference	273
10.130.Field Documentation	273
10.130.1.1OperatingSystem	273
10.131Arc::ExecutionEnvironmentType Class Reference	274

10.132.Arc::ExecutionTarget Class Reference	275
10.132.1.Detailed Description	275
10.132.2.Constructor & Destructor Documentation	275
10.132.2.1.ExecutionTarget	275
10.132.2.2.ExecutionTarget	275
10.132.2.3.ExecutionTarget	275
10.132.3.Member Function Documentation	276
10.132.3.1.RegisterJobSubmission	276
10.132.4.Friends And Related Function Documentation	276
10.132.4.1.operator<<	276
10.133.Arc::ExecutionTargetSorter Class Reference	277
10.133.1.Member Function Documentation	277
10.133.1.1.addEntity	277
10.134.Arc::ExpirationReminder Class Reference	278
10.134.1.Detailed Description	278
10.134.2.Member Function Documentation	278
10.134.2.1.getExpiryTime	278
10.134.2.2.getReservationID	278
10.134.2.3.operator<	278
10.135.Arc::FileAccess Class Reference	279
10.135.1.Detailed Description	280
10.135.2.Member Function Documentation	280
10.135.2.1.fa_copy	280
10.135.2.2.fa_mkdirp	280
10.135.2.3.fa_mkstemp	280
10.135.2.4.fa_setuid	280
10.136.Arc::FileAccessContainer Class Reference	281
10.136.1.Detailed Description	281
10.136.2.Member Function Documentation	281
10.136.2.1.Acquire	281
10.136.2.2.Release	281
10.137.Arc::FileLock Class Reference	282
10.137.1.Detailed Description	282
10.137.2.Constructor & Destructor Documentation	282
10.137.2.1.FileLock	282
10.137.3.Member Function Documentation	283

10.137.3.1acquire	283
10.137.3.2acquire	283
10.137.3.3check	283
10.137.3.4release	283
10.138Arc::FinderLoader Class Reference	285
10.139ArcSec::FnFactory Class Reference	286
10.139.1Detailed Description	286
10.139.2Member Function Documentation	286
10.139.2.1createFn	286
10.140ArcSec::Function Class Reference	287
10.140.1Detailed Description	287
10.140.2Member Function Documentation	287
10.140.2.1evaluate	287
10.140.2.2evaluate	287
10.141ArcSec::GenericAttribute Class Reference	288
10.141.1Member Function Documentation	288
10.141.1.1encode	288
10.141.1.2getId	288
10.141.1.3getType	288
10.142Arc::GlobusResult Class Reference	289
10.143Arc::GLUE2 Class Reference	290
10.143.1Detailed Description	290
10.143.2Member Function Documentation	290
10.143.2.1ParseExecutionTargets	290
10.144Arc::GLUE2Entity< T > Class Template Reference	291
10.145Arc::GSSCredential Class Reference	292
10.145.1Detailed Description	292
10.146Arc::HakaClient Class Reference	293
10.146.1Member Function Documentation	293
10.146.1.1processConsent	293
10.146.1.2processIdP2Confusa	293
10.146.1.3processIdPLogin	293
10.147Arc::FileAccess::header_t Struct Reference	294
10.147.1Detailed Description	294
10.148Arc::HTTPClientInfo Struct Reference	295
10.149Arc::InfoCache Class Reference	296

10.149.Detailed Description	296
10.149.Constructor & Destructor Documentation	296
10.149.2.IInfoCache	296
10.150Arc::InfoCacheInterface Class Reference	297
10.150.Member Function Documentation	297
10.150.1.IGet	297
10.151Arc::InfoFilter Class Reference	298
10.151.Detailed Description	298
10.151.Constructor & Destructor Documentation	298
10.151.2.IInfoFilter	298
10.151.Member Function Documentation	298
10.151.3.IFilter	298
10.151.3.2.Filter	298
10.152Arc::InfoRegister Class Reference	299
10.152.Detailed Description	299
10.153Arc::InfoRegisterContainer Class Reference	300
10.153.Detailed Description	300
10.153.Member Function Documentation	300
10.153.2.IaddRegistrar	300
10.153.2.2.addService	300
10.153.2.3.removeService	300
10.154Arc::InfoRegisters Class Reference	301
10.154.Detailed Description	301
10.154.Constructor & Destructor Documentation	301
10.154.2.IInfoRegisters	301
10.155Arc::InfoRegistrar Class Reference	302
10.155.Detailed Description	302
10.155.Member Function Documentation	302
10.155.2.IaddService	302
10.155.2.2.registration	302
10.156Arc::InformationContainer Class Reference	303
10.156.Detailed Description	303
10.156.Constructor & Destructor Documentation	303
10.156.2.IInformationContainer	303
10.156.Member Function Documentation	303
10.156.3.IAcquire	303

10.156.3.2Assign	303
10.156.3.3Get	304
10.156.4Field Documentation	304
10.156.4.1doc_	304
10.157Arc::InformationInterface Class Reference	305
10.157.1Detailed Description	305
10.157.2Constructor & Destructor Documentation	305
10.157.2.1InformationInterface	305
10.157.3Member Function Documentation	305
10.157.3.1Get	305
10.157.4Field Documentation	306
10.157.4.1lock_	306
10.158Arc::InformationRequest Class Reference	307
10.158.1Detailed Description	307
10.158.2Constructor & Destructor Documentation	307
10.158.2.1InformationRequest	307
10.158.2.2InformationRequest	307
10.158.2.3InformationRequest	307
10.158.2.4InformationRequest	307
10.158.3Member Function Documentation	307
10.158.3.1SOAP	307
10.159Arc::InformationResponse Class Reference	308
10.159.1Detailed Description	308
10.159.2Constructor & Destructor Documentation	308
10.159.2.1InformationResponse	308
10.159.3Member Function Documentation	308
10.159.3.1Result	308
10.160Arc::IniConfig Class Reference	309
10.160.1Detailed Description	309
10.161Arc::initializeCredentialsType Class Reference	310
10.161.1Detailed Description	310
10.161.2Member Enumeration Documentation	310
10.161.2.1initializeType	310
10.162Arc::InputFileType Class Reference	311
10.162.1Field Documentation	311
10.162.1.1Checksum	311

10.163.ArcSec::InRangeFunction Class Reference	312
10.163.1.Member Function Documentation	312
10.163.1.1.evaluate	312
10.163.1.2.evaluate	312
10.164.Arc::InterruptGuard Class Reference	313
10.164.1.Detailed Description	313
10.165.Arc::IntraProcessCounter Class Reference	314
10.165.1.Detailed Description	314
10.165.2.Constructor & Destructor Documentation	314
10.165.2.1.IntraProcessCounter	314
10.165.2.2.~IntraProcessCounter	315
10.165.3.Member Function Documentation	315
10.165.3.1.cancel	315
10.165.3.2.changeExcess	315
10.165.3.3.changeLimit	315
10.165.3.4.extend	315
10.165.3.5.getExcess	316
10.165.3.6.getLimit	316
10.165.3.7.getValue	316
10.165.3.8.reserve	316
10.165.3.9.setExcess	317
10.165.3.10.setLimit	317
10.166.Arc::ISIS_description Struct Reference	318
10.167.Arc::IString Class Reference	319
10.167.1.Detailed Description	319
10.168.Arc::JobDescriptionParserPluginLoader::iterator Class Reference	320
10.169.Arc::Job Class Reference	321
10.169.1.Detailed Description	321
10.169.2.Constructor & Destructor Documentation	321
10.169.2.1.Job	321
10.169.3.Member Function Documentation	321
10.169.3.1.operator=	321
10.169.3.2.ReadJobIDsFromFile	322
10.169.3.3.SaveToStream	322
10.169.3.4.SetFromXML	322
10.169.3.5.ToXML	323

10.169.3.6WriteJobIDsToFile	323
10.169.3.7WriteJobIDToFile	323
10.169.4Field Documentation	324
10.169.4.1JobDescription	324
10.169.4.2JobDescriptionDocument	324
10.170Arc::JobControllerPlugin Class Reference	325
10.17Arc::JobControllerPluginArgument Class Reference	326
10.17Arc::JobControllerPluginLoader Class Reference	327
10.172.Detailed Description	327
10.172.Constructor & Destructor Documentation	327
10.172.2.1JobControllerPluginLoader	327
10.172.2.2~JobControllerPluginLoader	327
10.172.3Member Function Documentation	327
10.172.3.1load	327
10.17Arc::JobControllerPluginTestACCCControl Class Reference	329
10.174Arc::JobDescription Class Reference	330
10.174.Detailed Description	330
10.174.2Member Function Documentation	330
10.174.2.1GetSourceLanguage	330
10.174.2.2Parse	331
10.174.2.3Prepare	331
10.174.2.4SaveToStream	332
10.174.2.5UnParse	332
10.174.3Field Documentation	332
10.174.3.1OtherAttributes	332
10.17Arc::JobDescriptionParserPlugin Class Reference	333
10.175.Detailed Description	333
10.176Arc::JobDescriptionParserPluginLoader Class Reference	334
10.176.Detailed Description	334
10.176.Constructor & Destructor Documentation	334
10.176.2.1JobDescriptionParserPluginLoader	334
10.176.2.2~JobDescriptionParserPluginLoader	334
10.176.3Member Function Documentation	334
10.176.3.1GetJobDescriptionParserPlugins	334
10.176.3.2load	335
10.17Arc::JobDescriptionParserPluginResult Class Reference	336

10.17	Arc::JobDescriptionParserPluginTestACCCControl Class Reference	337
10.17	Arc::JobDescriptionResult Class Reference	338
10.18	Arc::JobIdentificationType Class Reference	339
10.180.	Detailed Description	339
10.180.	Field Documentation	339
10.180.2.1	ActivityOldID	339
10.180.2.2	Annotation	339
10.180.2.3	Description	339
10.180.2.4	JobName	339
10.180.2.5	Type	340
10.18	Arc::JobInformationStorage Class Reference	341
10.181.	Detailed Description	341
10.181.	Constructor & Destructor Documentation	341
10.181.2.1	JobInformationStorage	341
10.181.	Member Function Documentation	342
10.181.3.1	Clean	342
10.181.3.2	GetName	342
10.181.3.3	Read	342
10.181.3.4	ReadAll	343
10.181.3.5	Remove	343
10.181.3.6	Write	344
10.181.3.7	Write	344
10.18	Arc::JobInformationStorageXML Class Reference	345
10.182.	Member Function Documentation	345
10.182.1.1	Clean	345
10.182.1.2	Read	345
10.182.1.3	ReadAll	346
10.182.1.4	Remove	347
10.182.1.5	Write	347
10.18	Arc::JobListRetrieverPlugin Class Reference	348
10.18	Arc::JobListRetrieverPluginTESTControl Class Reference	349
10.18	Arc::JobState Class Reference	350
10.185.	Detailed Description	350
10.185.	Member Function Documentation	350
10.185.2.1	GetGeneralState	350
10.185.2.2	GetSpecificState	350

10.185.2.3IsFinished	351
10.185.2.4operator()	351
10.186Arc::JobStateTEST Class Reference	352
10.187Arc::JobSupervisor Class Reference	353
10.187.Detailed Description	353
10.187.Constructor & Destructor Documentation	353
10.187.2.IJobSupervisor	353
10.187.3.Member Function Documentation	354
10.187.3.1addEntity	354
10.187.3.2AddJob	354
10.187.3.3Cancel	354
10.187.3.4Clean	355
10.187.3.5Migrate	355
10.187.3.6Renew	356
10.187.3.7Resubmit	356
10.187.3.8Resume	357
10.187.3.9Retrieve	358
10.187.3.10Update	358
10.188st Class Reference	359
10.189Arc::Loader Class Reference	360
10.189.Detailed Description	360
10.189.Constructor & Destructor Documentation	360
10.189.2.ILoader	360
10.189.2.2~Loader	360
10.189.Field Documentation	360
10.189.3.Ifactory_	360
10.190Arc::LocationAttributes Class Reference	361
10.191Arc::LocationType Class Reference	362
10.192Arc::LogDestination Class Reference	363
10.192.Detailed Description	363
10.193Arc::LogFile Class Reference	364
10.193.Detailed Description	364
10.193.Constructor & Destructor Documentation	364
10.193.2.ILogFile	364
10.193.3.Member Function Documentation	364
10.193.3.1log	364

10.193.3.2	setBackups	365
10.193.3.3	setMaxSize	365
10.193.3.4	setReopen	365
10.194	Arc::Logger Class Reference	366
10.194.	Detailed Description	366
10.194.	Constructor & Destructor Documentation	367
10.194.2.	ILogger	367
10.194.2.	Logger	367
10.194.	Member Function Documentation	367
10.194.3.	addDestination	367
10.194.3.2	addDestinations	367
10.194.3.3	getDestinations	367
10.194.3.4	getRootLogger	368
10.194.3.5	msg	368
10.194.3.6	msg	368
10.194.3.7	setDestinations	368
10.194.3.8	setThreadContext	368
10.194.3.9	setThreshold	368
10.194.3.10	setThresholdForDomain	369
10.194.3.11	setThresholdForDomain	369
10.195	Arc::LoggerFormat Struct Reference	370
10.195.	Detailed Description	370
10.196	Arc::LogMessage Class Reference	371
10.196.	Detailed Description	371
10.196.	Constructor & Destructor Documentation	371
10.196.2.	ILogMessage	371
10.196.2.	LogMessage	371
10.196.	Member Function Documentation	372
10.196.3.	getLevel	372
10.196.3.2	setIdentifier	372
10.196.4	Friends And Related Function Documentation	372
10.196.4.	ILogger	372
10.196.4.2	operator<<	372
10.197	Arc::LogStream Class Reference	373
10.197.	Detailed Description	373
10.197.	Constructor & Destructor Documentation	373

10.197.2. <code>ILogStream</code>	373
10.197.3. Member Function Documentation	373
10.197.3.1 <code>log</code>	373
10.198. <code>ArcSec::MatchFunction</code> Class Reference	375
10.198.1. Detailed Description	375
10.198.2. Member Function Documentation	375
10.198.2.1 <code>evaluate</code>	375
10.198.2.2 <code>evaluate</code>	375
10.198.2.3 <code>getFunctionName</code>	375
10.199. <code>Arc::MCC</code> Class Reference	376
10.199.1. Detailed Description	376
10.199.2. Constructor & Destructor Documentation	377
10.199.2.1 <code>IMCC</code>	377
10.199.3. Member Function Documentation	377
10.199.3.1 <code>AddSecHandler</code>	377
10.199.3.2 <code>Next</code>	377
10.199.3.3 <code>Next</code>	377
10.199.3.4 <code>process</code>	377
10.199.3.5 <code>ProcessSecHandlers</code>	377
10.199.3.6 <code>Unlink</code>	377
10.199.4. Field Documentation	378
10.199.4.1 <code>logger</code>	378
10.199.4.2 <code>next_</code>	378
10.199.4.3 <code>next_lock_</code>	378
10.199.4.4 <code>sechandlers_</code>	378
10.200. <code>Arc::MCC_Status</code> Class Reference	379
10.200.1. Detailed Description	379
10.200.2. Constructor & Destructor Documentation	379
10.200.2.1 <code>IMCC_Status</code>	379
10.200.3. Member Function Documentation	379
10.200.3.1 <code>getExplanation</code>	379
10.200.3.2 <code>getKind</code>	380
10.200.3.3 <code>getOrigin</code>	380
10.200.3.4 <code>isOk</code>	380
10.200.3.5 <code>operator bool</code>	380
10.200.3.6 <code>operator std::string</code>	380

10.200.3.7operator!	380
10.20Arc::MCCConfig Class Reference	381
10.201.Member Function Documentation	381
10.201.1.IMakeConfig	381
10.20Arc::MCCInterface Class Reference	382
10.202.Detailed Description	382
10.202.Member Function Documentation	382
10.202.2.lprocess	382
10.20Arc::MCCLoader Class Reference	383
10.203.Detailed Description	383
10.203.Constructor & Destructor Documentation	383
10.203.2.IMCCLoader	383
10.203.2.2~MCCLoader	383
10.203.Member Function Documentation	383
10.203.3.operator[]	383
10.20Arc::MCCPluginArgument Class Reference	385
10.20Arc::MD5Sum Class Reference	386
10.205.Detailed Description	386
10.205.Member Function Documentation	386
10.205.2.ladd	386
10.205.2.2end	386
10.205.2.3print	387
10.205.2.4scan	387
10.205.2.5start	387
10.20Arc::Message Class Reference	388
10.206.Detailed Description	388
10.206.Constructor & Destructor Documentation	389
10.206.2.IMessage	389
10.206.2.2Message	389
10.206.2.3Message	389
10.206.2.4~Message	389
10.206.Member Function Documentation	389
10.206.3.1Attributes	389
10.206.3.2Auth	389
10.206.3.3AuthContext	389
10.206.3.4AuthContext	389

10.206.3.5Context	389
10.206.3.6Context	390
10.206.3.7operator=	390
10.206.3.8Payload	390
10.206.3.9Payload	390
10.207Arc::MessageAttributes Class Reference	391
10.207.1Detailed Description	391
10.207.2Constructor & Destructor Documentation	391
10.207.2.1MessageAttributes	391
10.207.3Member Function Documentation	392
10.207.3.1add	392
10.207.3.2count	392
10.207.3.3get	392
10.207.3.4getAll	392
10.207.3.5remove	393
10.207.3.6removeAll	393
10.207.3.7set	393
10.207.4Field Documentation	393
10.207.4.1attributes_	393
10.208Arc::MessageAuth Class Reference	394
10.208.1Detailed Description	394
10.208.2Member Function Documentation	394
10.208.2.1Export	394
10.208.2.2Filter	394
10.209Arc::MessageAuthContext Class Reference	395
10.209.1Detailed Description	395
10.210Arc::MessageContext Class Reference	396
10.210.1Detailed Description	396
10.210.2Member Function Documentation	396
10.210.2.1Add	396
10.211Arc::MessageContextElement Class Reference	397
10.211.1Detailed Description	397
10.212Arc::MessagePayload Class Reference	398
10.212.1Detailed Description	398
10.213Arc::PluginsFactory::modules_t::miterator Class Reference	399
10.214Arc::ModuleDesc Class Reference	400

10.214. Detailed Description	400
10.215. Arc::ModuleManager Class Reference	401
10.215. Detailed Description	401
10.215. Constructor & Destructor Documentation	402
10.215.2. IModuleManager	402
10.215.3. Member Function Documentation	402
10.215.3.1. find	402
10.215.3.2. findLocation	402
10.215.3.3. load	402
10.215.3.4. makePersistent	402
10.215.3.5. makePersistent	402
10.215.3.6. reload	402
10.215.3.7. setCfg	402
10.215.3.8. unload	403
10.215.3.9. unload	403
10.215.3.10. use	403
10.215.3.11. use	403
10.216. Arc::MultiSecAttr Class Reference	404
10.216. Detailed Description	404
10.216. Member Function Documentation	404
10.216.2. IExport	404
10.216.2.2. operator bool	404
10.217. Arc::MySQLDatabase Class Reference	405
10.217. Detailed Description	405
10.217. Member Function Documentation	405
10.217.2.1. connect	405
10.217.2.2. enable_ssl	405
10.218. Arc::MySQLQuery Class Reference	407
10.218. Detailed Description	407
10.218. Member Function Documentation	407
10.218.2.1. execute	407
10.218.2.2. get_array	407
10.218.2.3. get_row	408
10.218.2.4. get_row	408
10.218.2.5. get_row_field	408
10.219. Arc::NotificationType Class Reference	409

10.220.Arc::NS Class Reference	410
10.220.Detailed Description	410
10.220.Constructor & Destructor Documentation	410
10.220.2.INS	410
10.221.Arc::OAuthConsumer Class Reference	411
10.221.Detailed Description	411
10.221.Constructor & Destructor Documentation	411
10.221.2.IOAuthConsumer	411
10.221.Member Function Documentation	411
10.221.3.1approveCSR	411
10.221.3.2parseDN	412
10.221.3.3processLogin	412
10.221.3.4pushCSR	412
10.221.3.5storeCert	412
10.222.Arc::OpenIdpClient Class Reference	413
10.222.Member Function Documentation	413
10.222.1.1processConsent	413
10.222.1.2processIdP2Confusa	413
10.222.1.3processIdPLogin	413
10.223.Arc::OptIn< T > Class Template Reference	414
10.224.Arc::OptionParser Class Reference	415
10.224.Detailed Description	415
10.224.Constructor & Destructor Documentation	415
10.224.2.IOOptionParser	415
10.224.Member Function Documentation	415
10.224.3.1AddOption	415
10.224.3.2AddOption	416
10.224.3.3AddOption	416
10.224.3.4AddOption	416
10.224.3.5Parse	417
10.225.ArcSec::OrderedCombiningAlg Class Reference	418
10.226.Arc::OutputFileType Class Reference	419
10.227.Arc::ParallelEnvironmentType Class Reference	420
10.228.Arc::PathIterator Class Reference	421
10.228.Detailed Description	421
10.228.Constructor & Destructor Documentation	421

10.228.2. IPathIterator	421
10.229. Arc::PayloadRaw Class Reference	422
10.229. Detailed Description	422
10.229. Constructor & Destructor Documentation	422
10.229.2. IPayloadRaw	422
10.229.2.2. ~PayloadRaw	422
10.229.3. Member Function Documentation	422
10.229.3.1. IBuffer	422
10.229.3.2. BufferPos	423
10.229.3.3. BufferSize	423
10.229.3.4. Content	423
10.229.3.5. Insert	423
10.229.3.6. Insert	423
10.229.3.7. operator[]	423
10.229.3.8. Size	423
10.229.3.9. Truncate	424
10.230. Arc::PayloadRawBuf Struct Reference	425
10.230. Field Documentation	425
10.230.1. lallocated	425
10.230.1. length	425
10.230.1. size	425
10.231. Arc::PayloadRawInterface Class Reference	426
10.231. Detailed Description	426
10.231. Member Function Documentation	426
10.231.2. IBuffer	426
10.231.2. BufferPos	426
10.231.2. BufferSize	427
10.231.2. Content	427
10.231.2. Insert	427
10.231.2. Insert	427
10.231.2. operator[]	427
10.231.2. Size	427
10.231.2. Truncate	427
10.232. Arc::PayloadSOAP Class Reference	428
10.232. Detailed Description	428
10.232. Constructor & Destructor Documentation	428

10.232.2.1PayloadSOAP	428
10.232.2.2PayloadSOAP	428
10.232.2.3PayloadSOAP	428
10.233Arc::PayloadStream Class Reference	429
10.233.1Detailed Description	429
10.233.2Constructor & Destructor Documentation	429
10.233.2.1PayloadStream	429
10.233.2.2~PayloadStream	429
10.233.3Member Function Documentation	430
10.233.3.1Get	430
10.233.3.2Limit	430
10.233.3.3operator bool	430
10.233.3.4operator!	430
10.233.3.5Pos	430
10.233.3.6Put	430
10.233.3.7Size	430
10.233.3.8Timeout	430
10.233.3.9Timeout	431
10.233.4Field Documentation	431
10.233.4.1handle_	431
10.233.4.2seekable_	431
10.234Arc::PayloadStreamInterface Class Reference	432
10.234.1Detailed Description	432
10.234.2Member Function Documentation	432
10.234.2.1Get	432
10.234.2.2Get	433
10.234.2.3Get	433
10.234.2.4Get	433
10.234.2.5Limit	433
10.234.2.6operator bool	433
10.234.2.7operator!	433
10.234.2.8Pos	433
10.234.2.9Put	433
10.234.2.10Put	434
10.234.2.11Put	434
10.234.2.12Put	434

10.234.2.1Size	434
10.234.2.1Timeout	434
10.234.2.1Timeout	434
10.235Arc::PayloadWSRF Class Reference	435
10.235.1Detailed Description	435
10.235.2Constructor & Destructor Documentation	435
10.235.2.1PayloadWSRF	435
10.235.2.2PayloadWSRF	435
10.235.2.3PayloadWSRF	435
10.236ArcSec::PDP Class Reference	436
10.236.1Detailed Description	436
10.237ArcSec::PDPCfgContext Class Reference	437
10.238ArcSec::PDPPluginArgument Class Reference	438
10.239Arc::Period Class Reference	439
10.239.1Detailed Description	439
10.240ArcSec::PeriodAttribute Class Reference	440
10.240.1Detailed Description	440
10.240.2Member Function Documentation	440
10.240.2.1encode	440
10.240.2.2getId	440
10.240.2.3getType	440
10.241ArcSec::PermitOverridesCombiningAlg Class Reference	441
10.241.1Detailed Description	441
10.241.2Member Function Documentation	441
10.241.2.1combine	441
10.241.2.2getalgId	441
10.242Arc::Plexer Class Reference	443
10.242.1Detailed Description	443
10.242.2Constructor & Destructor Documentation	443
10.242.2.1Plexer	443
10.242.2.2~Plexer	443
10.242.3Member Function Documentation	444
10.242.3.1Next	444
10.242.3.2process	444
10.242.4Field Documentation	444
10.242.4.1logger	444

10.243.Arc::PlexerEntry Class Reference	445
10.243.1.Detailed Description	445
10.244.Arc::Plugin Class Reference	446
10.244.1.Detailed Description	446
10.245.Arc::PluginArgument Class Reference	447
10.245.1.Detailed Description	447
10.245.2.Member Function Documentation	447
10.245.2.1.get_factory	447
10.245.2.2.get_module	448
10.246.Arc::PluginDesc Class Reference	449
10.246.1.Detailed Description	449
10.247.Arc::PluginDescriptor Struct Reference	450
10.247.1.Detailed Description	450
10.248.Arc::PluginsFactory Class Reference	451
10.248.1.Detailed Description	451
10.248.2.Constructor & Destructor Documentation	452
10.248.2.1.PluginsFactory	452
10.248.3.Member Function Documentation	452
10.248.3.1.FilterByKind	452
10.248.3.2.get_instance	452
10.248.3.3.load	452
10.248.3.4.report	452
10.248.3.5.scan	452
10.248.3.6.TryLoad	452
10.249.ArcSec::Policy Class Reference	454
10.249.1.Detailed Description	454
10.249.2.Constructor & Destructor Documentation	454
10.249.2.1.Policy	454
10.249.2.2.Policy	455
10.249.3.Member Function Documentation	455
10.249.3.1.addPolicy	455
10.249.3.2.eval	455
10.249.3.3.getEffect	455
10.249.3.4.getEvalName	455
10.249.3.5.getEvalResult	455
10.249.3.6.getName	455

10.249.3.7make_policy	455
10.249.3.8setEvalResult	455
10.249.3.9setEvaluatorContext	456
10.250ArcSec::PolicyStore::PolicyElement Class Reference	457
10.251ArcSec::PolicyParser Class Reference	458
10.251.1.Detailed Description	458
10.251.2.Member Function Documentation	458
10.251.2.1.parsePolicy	458
10.252ArcSec::PolicyStore Class Reference	459
10.252.1.Detailed Description	459
10.252.2.Constructor & Destructor Documentation	459
10.252.2.1.PolicyStore	459
10.253AuthN::PrivateKeyInfoCodec Class Reference	460
10.254DataStaging::Processor Class Reference	461
10.254.1.Detailed Description	461
10.254.2.Member Function Documentation	461
10.254.2.1.receiveDTR	461
10.254.2.2.start	462
10.254.2.3.stop	462
10.255Arc::Profile Class Reference	463
10.255.1.Detailed Description	463
10.256ArcCredential::PROXYCERTINFO_st Struct Reference	464
10.257ArcCredential::PROXYPOLICY_st Struct Reference	465
10.258Arc::Query Class Reference	466
10.258.1.Detailed Description	466
10.258.2.Constructor & Destructor Documentation	466
10.258.2.1.Query	466
10.258.3.Member Function Documentation	466
10.258.3.1.execute	466
10.258.3.2.get_array	467
10.258.3.3.get_row	467
10.258.3.4.get_row	467
10.258.3.5.get_row_field	467
10.259Arc::Range< T > Class Template Reference	468
10.260Arc::Register_Info_Type Struct Reference	469
10.261Arc::RegisteredService Class Reference	470

10.261.Detailed Description	470
10.261.Constructor & Destructor Documentation	470
10.261.2.IRegisteredService	470
10.262.Arc::RegularExpression Class Reference	471
10.262.Detailed Description	471
10.262.Member Function Documentation	471
10.262.2.lmatch	471
10.263.Arc::RemoteLoggingType Class Reference	472
10.263.Detailed Description	472
10.263.Field Documentation	472
10.263.2.ILocation	472
10.263.2.Optional	472
10.263.2.3ServiceType	472
10.264.ArcSec::Request Class Reference	473
10.264.Detailed Description	473
10.264.Constructor & Destructor Documentation	473
10.264.2.IRequest	473
10.264.2.Request	473
10.264.Member Function Documentation	474
10.264.3.laddRequestItem	474
10.264.3.2getEvalName	474
10.264.3.3getName	474
10.264.3.4getRequestItems	474
10.264.3.5make_request	474
10.264.3.6setAttributeFactory	474
10.264.3.7setRequestItems	474
10.265.ArcSec::RequestAttribute Class Reference	475
10.265.Detailed Description	475
10.265.Constructor & Destructor Documentation	475
10.265.2.IRequestAttribute	475
10.265.Member Function Documentation	475
10.265.3.lduplicate	475
10.266.ArcSec::RequestItem Class Reference	476
10.266.Detailed Description	476
10.266.Constructor & Destructor Documentation	476
10.266.2.IRequestItem	476

10.26	ArcSec::RequestTuple Class Reference	477
10.26	Arc::ResourceType Class Reference	478
10.26	ArcSec::Response Class Reference	479
10.269	Detailed Description	479
10.27	ArcSec::ResponseItem Class Reference	480
10.270	Detailed Description	480
10.27	ArcSec::ResponseList Class Reference	481
10.27	Arc::EntityRetriever< T >::Result Class Reference	482
10.27	Arc::Run Class Reference	483
10.273	Detailed Description	483
10.273	Member Function Documentation	484
10.273.2.1	Abandon	484
10.273.2.2	AfterFork	484
10.273.2.3	AssignStderr	484
10.273.2.4	AssignStdin	484
10.273.2.5	AssignStdout	484
10.273.2.6	Kill	484
10.273.2.7	ReadStderr	484
10.273.2.8	ReadStdout	485
10.273.2.9	Result	485
10.273.2.10	Start	485
10.273.2.11	Wait	485
10.273.2.12	Wait	485
10.273.2.13	WriteStdin	485
10.27	Arc::SAML2LoginClient Class Reference	487
10.274	Constructor & Destructor Documentation	487
10.274.1	ISAML2LoginClient	487
10.274	Member Function Documentation	487
10.274.2.1	findSimpleSAMLInstallation	487
10.274.2.2	processLogin	487
10.27	Arc::SAML2SSOHTTPClient Class Reference	488
10.275	Member Function Documentation	488
10.275.1.1	approveCSR	488
10.275.1.2	parseDN	488
10.275.1.3	processConsent	488
10.275.1.4	processIdP2Confusa	489

10.275.1.5processIdPLogin	489
10.275.1.6processLogin	489
10.275.1.7pushCSR	489
10.275.1.8storeCert	489
10.276Arc::SAMLToken Class Reference	490
10.276.1Detailed Description	490
10.276.2Member Enumeration Documentation	491
10.276.2.1SAMLVersion	491
10.276.3Constructor & Destructor Documentation	491
10.276.3.1SAMLToken	491
10.276.3.2SAMLToken	491
10.276.3.3~SAMLToken	491
10.276.4Member Function Documentation	492
10.276.4.1Authenticate	492
10.276.4.2Authenticate	492
10.276.4.3operator bool	492
10.277Arc::ScalableTime< T > Class Template Reference	493
10.278Arc::ScalableTime< int > Class Template Reference	494
10.279DataStaging::Scheduler Class Reference	495
10.279.1Detailed Description	495
10.279.2Member Function Documentation	495
10.279.2.1getInstance	495
10.279.2.2receiveDTR	496
10.279.2.3SetPreferredPattern	496
10.279.2.4start	496
10.279.2.5stop	496
10.280Arc::SecAttr Class Reference	497
10.280.1Detailed Description	497
10.280.2Member Function Documentation	497
10.280.2.1Export	497
10.280.2.2Export	498
10.280.2.3get	498
10.280.2.4getAll	498
10.280.2.5Import	498
10.280.2.6operator bool	498
10.280.2.7operator!=	498

10.280.2.8operator==	498
10.28Arc::SecAttrFormat Class Reference	499
10.281.Detailed Description	499
10.28Arc::SecAttrValue Class Reference	500
10.282.Detailed Description	500
10.282.Member Function Documentation	500
10.282.2.1operator bool	500
10.282.2.2operator!=	500
10.282.2.3operator==	500
10.28ArcSec::SecHandler Class Reference	501
10.283.Detailed Description	501
10.28ArcSec::SecHandlerConfig Class Reference	502
10.284.Detailed Description	502
10.28Arc::SecHandlerConfig Class Reference	503
10.28ArcSec::SecHandlerPluginArgument Class Reference	504
10.28ArcSec::Security Class Reference	505
10.287.Detailed Description	505
10.28Arc::Service Class Reference	506
10.288.Detailed Description	506
10.288.Constructor & Destructor Documentation	507
10.288.2.1Service	507
10.288.3.Member Function Documentation	507
10.288.3.1AddSecHandler	507
10.288.3.2getID	507
10.288.3.3operator bool	507
10.288.3.4operator!	507
10.288.3.5ProcessSecHandlers	507
10.288.3.6RegistrationCollector	507
10.288.4.Field Documentation	508
10.288.4.1logger	508
10.288.4.2sechandlers_	508
10.288.4.3valid	508
10.28Arc::ServiceEndpointRetrieverPlugin Class Reference	509
10.29Arc::ServiceEndpointRetrieverPluginTESTControl Class Reference	510
10.29Arc::ServicePluginArgument Class Reference	511
10.29Arc::SharedMutex Class Reference	512

10.292.Detailed Description	512
10.292.Member Function Documentation	512
10.292.2.lforceReset	512
10.293.Arc::SimpleCondition Class Reference	513
10.293.Detailed Description	513
10.293.Member Function Documentation	513
10.293.2.lbroadcast	513
10.293.2.2forceReset	513
10.293.2.3signal	513
10.293.2.4signal_nonblock	513
10.293.2.5wait	513
10.293.2.6wait_nonblock	514
10.294.Arc::SimpleCounter Class Reference	515
10.294.Detailed Description	515
10.294.Member Function Documentation	515
10.294.2.ldec	515
10.294.2.2forceReset	515
10.294.2.3get	515
10.294.2.4inc	515
10.294.2.5set	516
10.294.2.6wait	516
10.295.Arc::SlotRequirementType Class Reference	517
10.296.Arc::SOAPMessage Class Reference	518
10.296.Detailed Description	518
10.296.Constructor & Destructor Documentation	518
10.296.2.ISOAPMessage	518
10.296.2.2SOAPMessage	518
10.296.2.3SOAPMessage	518
10.296.2.4~SOAPMessage	518
10.296.Member Function Documentation	518
10.296.3.1Attributes	518
10.296.3.2Payload	519
10.296.3.3Payload	519
10.297.Arc::Software Class Reference	520
10.297.Detailed Description	521
10.297.Member Typedef Documentation	521

10.297.2.1ComparisonOperator	521
10.297.3Member Enumeration Documentation	521
10.297.3.1ComparisonOperatorEnum	521
10.297.4Constructor & Destructor Documentation	522
10.297.4.1Software	522
10.297.4.2Software	522
10.297.4.3Software	522
10.297.4.4Software	522
10.297.5Member Function Documentation	522
10.297.5.1convert	522
10.297.5.2empty	523
10.297.5.3getFamily	523
10.297.5.4getName	523
10.297.5.5getVersion	523
10.297.5.6operator std::string	523
10.297.5.7operator!=	524
10.297.5.8operator()	524
10.297.5.9operator<	524
10.297.5.10operator<=	524
10.297.5.11operator==	525
10.297.5.12operator>	525
10.297.5.13operator>=	526
10.297.5.14String	526
10.297.6Friends And Related Function Documentation	526
10.297.6.1operator<<	526
10.297.7Field Documentation	526
10.297.7.1VERSIONTOKENS	526
10.298Arc::SoftwareRequirement Class Reference	528
10.298.1Detailed Description	528
10.298.2Constructor & Destructor Documentation	528
10.298.2.1SoftwareRequirement	528
10.298.2.2SoftwareRequirement	529
10.298.2.3SoftwareRequirement	529
10.298.2.4SoftwareRequirement	529
10.298.3Member Function Documentation	529
10.298.3.1add	529

10.298.3.2add	530
10.298.3.3clear	530
10.298.3.4empty	530
10.298.3.5getComparisonOperatorList	530
10.298.3.6getSoftwareList	530
10.298.3.7isResolved	531
10.298.3.8isSatisfied	531
10.298.3.9isSatisfied	531
10.298.3.10isSatisfied	532
10.298.3.1operator=	532
10.298.3.12selectSoftware	532
10.298.3.13selectSoftware	533
10.298.3.14selectSoftware	533
10.299ArcSec::Source Class Reference	535
10.299.Detailed Description	535
10.299.Constructor & Destructor Documentation	535
10.299.2.1Source	535
10.299.2.2Source	535
10.299.2.3Source	535
10.300ArcSec::SourceFile Class Reference	536
10.300.Detailed Description	536
10.30Arc::SourceType Class Reference	537
10.30ArcSec::SourceURL Class Reference	538
10.302.Detailed Description	538
10.30DataStaging::DataDeliveryComm::Status Struct Reference	539
10.303.Detailed Description	539
10.304ArcSec::StringAttribute Class Reference	540
10.304.Member Function Documentation	540
10.304.1.encode	540
10.304.1.2getId	540
10.304.1.3getType	540
10.305Arc::SubmissionStatus Class Reference	541
10.306Arc::Submitter Class Reference	542
10.307Arc::SubmitterPlugin Class Reference	543
10.307.Detailed Description	543
10.307.Member Function Documentation	543

10.307.2.1Migrate	543
10.307.2.2Submit	543
10.307.2.3Submit	544
10.308Arc::SubmitterPluginArgument Class Reference	545
10.309Arc::SubmitterPluginLoader Class Reference	546
10.309.Detailed Description	546
10.309.Constructor & Destructor Documentation	546
10.309.2.1SubmitterPluginLoader	546
10.309.2.2~SubmitterPluginLoader	546
10.309.3.Member Function Documentation	546
10.309.3.1load	546
10.310Arc::SubmitterPluginTestACCCControl Class Reference	548
10.311Arc::TargetInformationRetrieverPlugin Class Reference	549
10.312Arc::TargetInformationRetrieverPluginTESTControl Class Reference	550
10.313Arc::TargetType Class Reference	551
10.314Arc::TCPSec Class Reference	552
10.315Arc::EntityRetriever< T >::ThreadArg Class Reference	553
10.316Arc::ThreadDataItem Class Reference	554
10.316.Detailed Description	554
10.316.Constructor & Destructor Documentation	554
10.316.2.1ThreadDataItem	554
10.316.2.2ThreadDataItem	554
10.316.3.Member Function Documentation	554
10.316.3.1Attach	554
10.316.3.2Attach	554
10.316.3.3Dup	555
10.316.3.4Get	555
10.317Arc::ThreadedPointer< T > Class Template Reference	556
10.317.Detailed Description	556
10.317.Member Function Documentation	556
10.317.2.1Release	556
10.317.2.2WaitInRange	556
10.317.2.3WaitOutOfRange	557
10.318Arc::ThreadInitializer Class Reference	558
10.318.Detailed Description	558
10.318.Member Function Documentation	558

10.318.2.1forceReset	558
10.318.2.2waitExit	558
10.319Arc::ThreadRegistry Class Reference	559
10.319.Detailed Description	559
10.319.Member Function Documentation	559
10.319.2.1forceReset	559
10.319.2.2WaitForExit	559
10.319.2.3WaitOrCancel	559
10.320Arc::Time Class Reference	560
10.320.Detailed Description	561
10.321ArcSec::TimeAttribute Class Reference	562
10.321.Detailed Description	562
10.321.Member Function Documentation	562
10.321.2.1encode	562
10.321.2.2getId	562
10.321.2.3getType	562
10.322Arc::TimedMutex Class Reference	563
10.322.Detailed Description	563
10.322.Member Function Documentation	563
10.322.2.1forceReset	563
10.322.2.2lock	563
10.323DataStaging::TransferParameters Class Reference	564
10.323.Detailed Description	564
10.323.Field Documentation	564
10.323.2.1max_inactivity_time	564
10.323.2.2min_average_bandwidth	564
10.323.2.3min_current_bandwidth	564
10.324DataStaging::TransferShares Class Reference	565
10.324.Detailed Description	565
10.324.Member Function Documentation	565
10.324.2.1calculate_shares	565
10.324.2.2decrease_number_of_slots	565
10.325DataStaging::TransferSharesConf Class Reference	566
10.325.Detailed Description	566
10.325.Member Enumeration Documentation	566
10.325.2.1ShareType	566

10.326.Arc::URL Class Reference	567
10.326.Detailed Description	569
10.326.Member Function Documentation	570
10.326.2.1AddHTTPOption	570
10.326.2.2AddOption	570
10.326.2.3AddOption	570
10.326.2.4CommonLocOption	570
10.326.2.5FullPathURIEncoded	571
10.326.2.6HTTPOption	571
10.326.2.7MetaDataOption	571
10.326.2.8Option	571
10.326.2.9OptionString	571
10.326.2.10RemoveHTTPOption	571
10.326.2.11RemoveMetaDataOption	572
10.326.2.12RemoveOption	572
10.326.2.13URIDecode	572
10.326.2.14URIDecode	572
10.326.2.15URIEncode	572
10.327.Arc::URLLocation Class Reference	573
10.327.Detailed Description	573
10.328.Arc::User Class Reference	574
10.328.Detailed Description	574
10.328.Constructor & Destructor Documentation	574
10.328.2.1User	574
10.328.2.2User	574
10.328.Member Function Documentation	574
10.328.3.1check_file_access	574
10.328.3.2SwitchUser	575
10.329.Arc::UserConfig Class Reference	576
10.329.Detailed Description	578
10.329.Constructor & Destructor Documentation	579
10.329.2.1UserConfig	579
10.329.2.2UserConfig	579
10.329.2.3UserConfig	580
10.329.2.4UserConfig	581
10.329.Member Function Documentation	581

10.329.3.1AddBartender	581
10.329.3.2AddRejectDiscoveryURLs	581
10.329.3.3ApplyToConfig	581
10.329.3.4Bartender	582
10.329.3.5Bartender	582
10.329.3.6Broker	582
10.329.3.7Broker	582
10.329.3.8Broker	583
10.329.3.9CACertificatePath	583
10.329.3.10CACertificatePath	584
10.329.3.11CACertificatesDirectory	584
10.329.3.12CACertificatesDirectory	584
10.329.3.13CertificateLifeTime	585
10.329.3.14CertificateLifeTime	585
10.329.3.15CertificatePath	585
10.329.3.16CertificatePath	585
10.329.3.17learRejectDiscoveryURLs	586
10.329.3.18redentialsFound	586
10.329.3.19etDefaultServices	586
10.329.3.20etService	587
10.329.3.21etServices	587
10.329.3.22etServicesInGroup	587
10.329.3.23etUser	588
10.329.3.24IPName	588
10.329.3.25IPName	588
10.329.3.26InfoInterface	588
10.329.3.27InfoInterface	589
10.329.3.28InitializeCredentials	589
10.329.3.29bbDownloadDirectory	590
10.329.3.30bbDownloadDirectory	591
10.329.3.31bbListFile	591
10.329.3.32bbListFile	591
10.329.3.33KeyPassword	592
10.329.3.34KeyPassword	592
10.329.3.35KeyPath	592
10.329.3.36KeyPath	593

10.329.3.3KeySize	593
10.329.3.3KeySize	593
10.329.3.3LoadConfigurationFile	594
10.329.3.4operator bool	594
10.329.3.4operator!	594
10.329.3.4OverlayFile	594
10.329.3.4OverlayFile	595
10.329.3.4Password	595
10.329.3.4Password	595
10.329.3.4ProxyPath	596
10.329.3.4ProxyPath	596
10.329.3.4RejectDiscoveryURLs	596
10.329.3.4RejectManagementURLs	596
10.329.3.5SaveToFile	597
10.329.3.5SetUser	597
10.329.3.5SLCS	597
10.329.3.5SLCS	597
10.329.3.5StoreDirectory	598
10.329.3.5StoreDirectory	598
10.329.3.5SubmissionInterface	598
10.329.3.5SubmissionInterface	598
10.329.3.5Timeout	599
10.329.3.5Timeout	599
10.329.3.6UserName	599
10.329.3.6UserName	600
10.329.3.6UtilsDirPath	600
10.329.3.6UtilsDirPath	600
10.329.3.6Verbosity	600
10.329.3.6Verbosity	601
10.329.3.6VOMSPath	601
10.329.3.6VOMSPath	601
10.329.4Field Documentation	602
10.329.4.1ARCUSERDIRECTORY	602
10.329.4.2DEFAULT_BROKER	602
10.329.4.3DEFAULT_TIMEOUT	602
10.329.4.4DEFAULTCONFIG	602

10.329.4.5EXAMPLECONFIG	602
10.329.4.6SYSCONFIG	602
10.329.4.7SYSCONFIGARCLOC	603
10.330Arc::UsernameToken Class Reference	604
10.330.1Detailed Description	604
10.330.2Member Enumeration Documentation	604
10.330.2.1PasswordType	604
10.330.3Constructor & Destructor Documentation	604
10.330.3.1UsernameToken	604
10.330.3.2UsernameToken	604
10.330.3.3UsernameToken	605
10.330.4Member Function Documentation	605
10.330.4.1Authenticate	605
10.330.4.2Authenticate	605
10.330.4.3operator bool	605
10.330.4.4Username	605
10.331Arc::UserSwitch Class Reference	606
10.331.1Detailed Description	606
10.332Arc::VOMSACInfo Class Reference	607
10.333Arc::VOMSTrustList Class Reference	608
10.333.1Detailed Description	608
10.333.2Constructor & Destructor Documentation	608
10.333.2.1VOMSTrustList	608
10.333.2.2VOMSTrustList	608
10.333.3Member Function Documentation	609
10.333.3.1AddChain	609
10.333.3.2AddChain	609
10.333.3.3AddRegex	609
10.334Arc::WatchdogChannel Class Reference	610
10.334.1Detailed Description	610
10.334.2Constructor & Destructor Documentation	610
10.334.2.1WatchdogChannel	610
10.335Arc::WatchdogListener Class Reference	611
10.335.1Detailed Description	611
10.335.2Member Function Documentation	611
10.335.2.1Listen	611

10.335.2.Listen	611
10.336.Arc::WSAEndpointReference Class Reference	612
10.336.Detailed Description	612
10.336.Constructor & Destructor Documentation	612
10.336.2.IWSAEndpointReference	612
10.336.2.2WSAEndpointReference	612
10.336.2.3WSAEndpointReference	612
10.336.2.4WSAEndpointReference	612
10.336.2.5~WSAEndpointReference	612
10.336.3.Member Function Documentation	613
10.336.3.1Address	613
10.336.3.2Address	613
10.336.3.3hasAddress	613
10.336.3.4MetaData	613
10.336.3.5operator XMLNode	613
10.336.3.6operator=	613
10.336.3.7ReferenceParameters	613
10.337.Arc::WSAHeader Class Reference	614
10.337.Detailed Description	614
10.337.Constructor & Destructor Documentation	615
10.337.2.IWSAHeader	615
10.337.2.2WSAHeader	615
10.337.3.Member Function Documentation	615
10.337.3.1Action	615
10.337.3.2Action	615
10.337.3.3Check	615
10.337.3.4FaultTo	615
10.337.3.5From	615
10.337.3.6hasAction	615
10.337.3.7hasMessageID	615
10.337.3.8hasRelatesTo	615
10.337.3.9hasRelationshipType	616
10.337.3.10hasTo	616
10.337.3.11MessageID	616
10.337.3.12MessageID	616
10.337.3.13NewReferenceParameter	616

10.337.3.1operator XMLNode	616
10.337.3.1ReferenceParameter	616
10.337.3.1ReferenceParameter	616
10.337.3.1RelatesTo	616
10.337.3.1RelatesTo	616
10.337.3.1RelationshipType	616
10.337.3.2RelationshipType	617
10.337.3.2ReplyTo	617
10.337.3.2To	617
10.337.3.2To	617
10.337.4Field Documentation	617
10.337.4.1header_allocated_	617
10.338Arc::WSRF Class Reference	618
10.338.1Detailed Description	618
10.338.2Constructor & Destructor Documentation	619
10.338.2.1WSRF	619
10.338.2.2WSRF	619
10.338.3Member Function Documentation	619
10.338.3.1operator bool	619
10.338.3.2set_namespaces	619
10.338.3.3SOAP	619
10.338.4Field Documentation	619
10.338.4.1allocated_	619
10.338.4.2valid_	619
10.339Arc::WSRFBBaseFault Class Reference	620
10.339.1Detailed Description	620
10.339.2Constructor & Destructor Documentation	620
10.339.2.1WSRFBBaseFault	620
10.339.2.2WSRFBBaseFault	620
10.339.3Member Function Documentation	620
10.339.3.1set_namespaces	620
10.340Arc::WSRFResourceUnavailableFault Class Reference	621
10.341Arc::WSRFResourceUnknownFault Class Reference	622
10.342Arc::WSRP Class Reference	623
10.342.1Detailed Description	623
10.342.2Constructor & Destructor Documentation	624

10.342.2.1WSRP	624
10.342.2.2WSRP	624
10.342.3Member Function Documentation	624
10.342.3.1set_namespaces	624
10.34Arc::WSRPDeleteResourceProperties Class Reference	625
10.34Arc::WSRPDeleteResourcePropertiesRequest Class Reference	626
10.34Arc::WSRPDeleteResourcePropertiesRequestFailedFault Class Reference	627
10.34Arc::WSRPDeleteResourcePropertiesResponse Class Reference	628
10.34Arc::WSRPFault Class Reference	629
10.347.Detailed Description	629
10.347.Constructor & Destructor Documentation	629
10.347.2.1WSRPFault	629
10.347.2.2WSRPFault	629
10.34Arc::WSRPGetMultipleResourcePropertiesRequest Class Reference	630
10.34Arc::WSRPGetMultipleResourcePropertiesResponse Class Reference	631
10.35Arc::WSRPGetResourcePropertyDocumentRequest Class Reference	632
10.35Arc::WSRPGetResourcePropertyDocumentResponse Class Reference	633
10.35Arc::WSRPGetResourcePropertyRequest Class Reference	634
10.35Arc::WSRPGetResourcePropertyResponse Class Reference	635
10.35Arc::WSRPInsertResourceProperties Class Reference	636
10.35Arc::WSRPInsertResourcePropertiesRequest Class Reference	637
10.35Arc::WSRPInsertResourcePropertiesRequestFailedFault Class Reference	638
10.35Arc::WSRPInsertResourcePropertiesResponse Class Reference	639
10.35Arc::WSRPInvalidModificationFault Class Reference	640
10.35Arc::WSRPInvalidResourcePropertyQNameFault Class Reference	641
10.36Arc::WSRPModifyResourceProperties Class Reference	642
10.36Arc::WSRPPutResourcePropertyDocumentRequest Class Reference	643
10.36Arc::WSRPPutResourcePropertyDocumentResponse Class Reference	644
10.36Arc::WSRPQueryResourcePropertiesRequest Class Reference	645
10.36Arc::WSRPQueryResourcePropertiesResponse Class Reference	646
10.36Arc::WSRPResourcePropertyChangeFailure Class Reference	647
10.365.Detailed Description	647
10.365.Constructor & Destructor Documentation	647
10.365.2.1WSRPResourcePropertyChangeFailure	647
10.365.2.2WSRPResourcePropertyChangeFailure	647
10.36Arc::WSRPSetResourcePropertiesRequest Class Reference	648

10.36	Arc::WSRPSetResourcePropertiesResponse Class Reference	649
10.36	Arc::WSRPSetResourcePropertyRequestFailedFault Class Reference	650
10.36	Arc::WSRPUnableToModifyResourcePropertyFault Class Reference	651
10.37	Arc::WSRPUnableToPutResourcePropertyDocumentFault Class Reference	652
10.37	Arc::WSRPUpdateResourceProperties Class Reference	653
10.37	Arc::WSRPUpdateResourcePropertiesRequest Class Reference	654
10.37	Arc::WSRPUpdateResourcePropertiesRequestFailedFault Class Reference	655
10.37	Arc::WSRPUpdateResourcePropertiesResponse Class Reference	656
10.37	ArcSec::X500NameAttribute Class Reference	657
10.375.	Member Function Documentation	657
10.375.1.	lencode	657
10.375.1.2	getId	657
10.375.1.3	getType	657
10.37	Arc::X509Token Class Reference	658
10.376.	Detailed Description	658
10.376.	Member Enumeration Documentation	658
10.376.2.	IX509TokenType	658
10.376.	Constructor & Destructor Documentation	658
10.376.3.	IX509Token	658
10.376.3.	2X509Token	658
10.376.3.3~	X509Token	659
10.376.	Member Function Documentation	659
10.376.4.1	Authenticate	659
10.376.4.2	Authenticate	659
10.376.4.3	operator bool	659
10.37	Arc::XmlContainer Class Reference	660
10.37	Arc::XmlDatabase Class Reference	661
10.37	Arc::XMLNode Class Reference	662
10.379.	Detailed Description	664
10.379.	Constructor & Destructor Documentation	664
10.379.2.	IXMLNode	664
10.379.2.2	XMLNode	664
10.379.2.3	XMLNode	664
10.379.2.4	XMLNode	664
10.379.2.5	XMLNode	664
10.379.2.6	XMLNode	664

10.379.2.7~XMLNode	665
10.379.3Member Function Documentation	665
10.379.3.1Child	665
10.379.3.2Destroy	665
10.379.3.3Exchange	665
10.379.3.4GetXML	665
10.379.3.5LogError	665
10.379.3.6Move	665
10.379.3.7Namespaces	666
10.379.3.8New	666
10.379.3.9NewChild	666
10.379.3.10NewChild	666
10.379.3.11NewChild	666
10.379.3.12operator++	666
10.379.3.13operator--	666
10.379.3.14operator=	667
10.379.3.15operator[]	667
10.379.3.16operator[]	667
10.379.3.17operator[]	667
10.379.3.18Path	667
10.379.3.19Prefix	667
10.379.3.20Swap	667
10.379.3.21Validate	668
10.379.3.22Validate	668
10.379.3.23XPathLookup	668
10.379.4Field Documentation	668
10.379.4.lis_owner_	668
10.380Arc::XMLNodeContainer Class Reference	669
10.380.Detailed Description	669
10.380.Constructor & Destructor Documentation	669
10.380.2.IXMLNodeContainer	669
10.380.3Member Function Documentation	669
10.380.3.1Add	669
10.380.3.2AddNew	669
10.38Arc::XMLSecNode Class Reference	670
10.381.Detailed Description	670

10.381.2	Constructor & Destructor Documentation	670
10.381.2.1	XMLSecNode	670
10.381.3	Member Function Documentation	670
10.381.3.1	AddSignatureTemplate	670
10.381.3.2	DecryptNode	671
10.381.3.3	EncryptNode	671
10.381.3.4	SignNode	671
10.381.3.5	VerifyNode	671
11	File Documentation	673
11.1	BrokerPlugin.h File Reference	673
11.1.1	Detailed Description	673
11.2	EntityRetrieverPlugin.h File Reference	674
11.2.1	Detailed Description	674
11.3	ExecutionTarget.h File Reference	675
11.3.1	Detailed Description	675
11.4	GLUE2Entity.h File Reference	676
11.4.1	Detailed Description	676
11.5	JobControllerPlugin.h File Reference	677
11.5.1	Detailed Description	677
11.6	JobDescription.h File Reference	678
11.6.1	Detailed Description	678
11.7	JobDescriptionParserPlugin.h File Reference	679
11.7.1	Detailed Description	679
11.8	Software.h File Reference	680
11.8.1	Detailed Description	680
11.9	SubmitterPlugin.h File Reference	681
11.9.1	Detailed Description	681
11.10	TestACCCControl.h File Reference	682
11.10.1	Detailed Description	682

Chapter 1

ARC Compute Library (libarccompute)

1.1 ARC Compute Library (libarccompute)

Data Structures

- class [Arc::Broker](#)
- class [Arc::ExecutionTargetSorter](#)
- class [Arc::ComputingServiceUniq](#)
- class [Arc::ComputingServiceRetriever](#)

Retrieves information about computing elements by querying service registries and CE information systems.

- class [Arc::EndpointStatusMap](#)
- class [Arc::Endpoint](#)

Represents an endpoint of a service with a given interface type and capabilities.

- class [Arc::EndpointQueryingStatus](#)

Represents the status in the [EntityRetriever](#) of the query process of an [Endpoint](#) (service registry, computing element).

- class [Arc::EntityConsumer< T >](#)

A general concept of an object which can consume entities use by the retrievers to return results.

- class [Arc::EntityContainer< T >](#)

An entity consumer class storing all the consumed entities in a list.

- class [Arc::EntityRetriever< T >](#)

Queries [Endpoint](#) objects (using plugins in parallel) and sends the found entities to consumers.

- class [Arc::ComputingServiceType](#)
- class [Arc::ExecutionTarget](#)

ExecutionTarget.

- class [Arc::Job](#)

Job.

- class [Arc::JobInformationStorage](#)

Abstract class for storing job information.

- class [Arc::JobDescriptionResult](#)
- class [Arc::JobDescription](#)
- class [Arc::JobState](#)
- class [Arc::JobSupervisor](#)
JobSupervisor class.
- class [Arc::SubmissionStatus](#)
- class [Arc::EndpointSubmissionStatus](#)
- class [Arc::Submitter](#)

Modules

- Structures holding resource information
- [JobDescription](#) related classes
- Plugin related classes for compute specialisations
- Classes for controlling output of compute test plugins

Typedefs

- typedef EntityRetriever< Endpoint > [Arc::ServiceEndpointRetriever](#)
- typedef EntityRetriever< ComputingServiceType > [Arc::TargetInformationRetriever](#)
- typedef EntityRetriever< Job > [Arc::JobListRetriever](#)

1.1.1 Typedef Documentation

1.1.1.1 typedef EntityRetriever<Job> Arc::JobListRetriever

The JobListRetriever is an [EntityRetriever](#) retrieving [Job](#) objects. It queries computing elements to get the list of jobs residing on the resource.

1.1.1.2 typedef EntityRetriever<Endpoint> Arc::ServiceEndpointRetriever

The ServiceEndpointRetriever is an [EntityRetriever](#) retrieving [Endpoint](#) objects. It queries service registries to get endpoints of registered services.

1.1.1.3 typedef EntityRetriever<ComputingServiceType> Arc::TargetInformationRetriever

The TargetInformationRetriever is an [EntityRetriever](#) retrieving [ComputingServiceType](#) objects. It queries computing elements to get the full [GLUE2](#) information about the resource.

Chapter 2

Todo List

Page [ARC Compute Library \(libarccompute\)](#) Write description of ARC Compute Library

Group [testaccecontrol](#) * Give examples on how to load and use the test plugins.

* Add descriptions to test control classes.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

ARC Compute Library (libarccompute)	35
Structures holding resource information	33
JobDescription related classes	34
Plugin related classes for compute specialisations	37
Classes for controlling output of compute test plugins	38
ARC data staging (libarcdatastaging)	39

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Arc (Arc namespace contains all core ARC classes)	43
ArcCredential	73
DataStaging (DataStaging contains all components for data transfer scheduling and execution) .	75

Chapter 5

Data Structure Index

5.1 Class Hierarchy

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

ArcCredential::ACACI	77
ArcCredential::ACATTHOLDER	78
ArcCredential::ACATTR	79
ArcCredential::ACATTRIBUTE	80
ArcCredential::ACC	81
ArcCredential::ACCERTS	82
ArcCredential::ACDIGEST	83
ArcCredential::ACFORM	84
ArcCredential::ACFULLATTRIBUTES	85
ArcCredential::ACHOLDER	86
ArcCredential::ACIETFATTR	87
ArcCredential::ACINFO	88
ArcCredential::ACIS	89
ArcCredential::ACSEQ	90
ArcCredential::ACTARGET	91
ArcCredential::ACTARGETS	92
ArcCredential::ACVAL	93
Arc::AdminDomainAttributes	96
Arc::ApplicationType	101
Arc::ArcLocation	103
ArcSec::ArcPeriod	104
Arc::ArcVersion	106
ArcSec::Attr	107
Arc::AttributeIterator	109
ArcSec::AttributeProxy	112
ArcSec::AttributeValue	113
ArcSec::AnyURIAttribute	99
ArcSec::BooleanAttribute	122
ArcSec::DateAttribute	206
ArcSec::DateTimeAttribute	207
ArcSec::DurationAttribute	236
ArcSec::GenericAttribute	288
ArcSec::PeriodAttribute	440

ArcSec::StringAttribute	540
ArcSec::TimeAttribute	562
ArcSec::X500NameAttribute	657
ArcSec::Attrs	115
ArcSec::AuthzRequest	116
ArcSec::AuthzRequestSection	117
Arc::AutoPointer< T >	118
Arc::Base64	119
Arc::BaseConfig	120
Arc::MCCCConfig	381
Arc::Broker	123
Arc::BrokerPluginTestACCCControl	127
ArcCredential::cert_verify_context	128
Arc::CertEnvLocker	129
AuthN::certInfo	130
Arc::ChainContext	131
Arc::Checksum	132
Arc::Adler32Sum	94
Arc::ChecksumAny	135
Arc::CRC32Sum	183
Arc::MD5Sum	386
Arc::ClientHTTPAttributes	143
Arc::ClientHTTPwithSAML2SSO	144
Arc::ClientInterface	145
Arc::ClientTCP	149
Arc::ClientHTTP	142
Arc::ClientSOAP	146
Arc::ClientSOAPwithSAML2SSO	148
Arc::ClientX509Delegation	150
ArcSec::CombiningAlg	152
ArcSec::DenyOverridesCombiningAlg	218
ArcSec::OrderedCombiningAlg	418
ArcSec::PermitOverridesCombiningAlg	441
Arc::ComputingEndpointAttributes	154
Arc::ComputingManagerAttributes	156
Arc::ComputingServiceAttributes	158
Arc::ComputingShareAttributes	164
Arc::ConfigEndpoint	168
Arc::ConfusaCertHandler	170
Arc::ConfusaParserUtils	171
Arc::CountedPointer< T >	173
Arc::CountedPointer< AdminDomainAttributes >	173
Arc::CountedPointer< ComputingEndpointAttributes >	173
Arc::CountedPointer< ComputingManagerAttributes >	173
Arc::CountedPointer< ComputingServiceAttributes >	173
Arc::CountedPointer< ComputingShareAttributes >	173
Arc::CountedPointer< ExecutionEnvironmentAttributes >	173
Arc::CountedPointer< LocationAttributes >	173
Arc::Counter	174
Arc::IntraProcessCounter	314
Arc::CounterTicket	181
Arc::Credential	185

Arc::CredentialError	194
Arc::CredentialStore	195
Arc::Database	196
Arc::MySQLDatabase	405
DataStaging::DataDeliveryComm	199
DataStaging::DataDeliveryLocalComm	203
DataStaging::DataDeliveryRemoteComm	204
DataStaging::DataDeliveryCommHandler	202
Arc::DataStagingType	205
Arc::DelegationConsumer	208
Arc::DelegationConsumerSOAP	210
Arc::DelegationContainerSOAP	212
Arc::DelegationProvider	215
Arc::DelegationProviderSOAP	216
Arc::DiskSpaceRequirementType	220
Arc::PluginsFactory::modules_t_::diterator	221
DataStaging::DTR	223
DataStaging::DTRCacheParameters	227
DataStaging::DTRCallback	228
DataStaging::DataDelivery	198
DataStaging::Processor	461
DataStaging::Scheduler	495
DataStaging::DTRErrorStatus	229
DataStaging::DTRLList	231
DataStaging::DTRStatus	234
Arc::Endpoint	237
Arc::EndpointQueryingStatus	242
Arc::EndpointQueryOptions< T >	246
Arc::EndpointQueryOptions< Endpoint >	247
Arc::EndpointStatusMap	248
Arc::EndpointSubmissionStatus	249
Arc::EntityConsumer< T >	252
Arc::EntityContainer< T >	253
Arc::EntityRetriever< T >	254
Arc::EntityConsumer< ComputingServiceType >	252
Arc::ComputingServiceUniq	163
Arc::ExecutionTargetSorter	277
Arc::EntityContainer< ComputingServiceType >	253
Arc::ComputingServiceRetriever	159
Arc::EntityConsumer< Endpoint >	252
Arc::ComputingServiceRetriever	159
Arc::EntityConsumer< Job >	252
Arc::JobSupervisor	353
Arc::EnvLockWrapper	262
ArcSec::EvalResult	264
ArcSec::EvaluationCtx	265
ArcSec::EvaluatorContext	269
ArcSec::EvaluatorLoader	270
Arc::ExecutableType	272
Arc::ExecutionEnvironmentAttributes	273
Arc::ExecutionTarget	275

Arc::ExpirationReminder	278
Arc::FileAccess	279
Arc::FileAccessContainer	281
Arc::FileLock	282
Arc::FinderLoader	285
ArcSec::Function	287
ArcSec::EqualFunction	263
ArcSec::InRangeFunction	312
ArcSec::MatchFunction	375
Arc::GlobusResult	289
Arc::GLUE2	290
Arc::GLUE2Entity< T >	291
Arc::GLUE2Entity< AdminDomainAttributes >	291
Arc::AdminDomainType	97
Arc::GLUE2Entity< ComputingEndpointAttributes >	291
Arc::ComputingEndpointType	155
Arc::GLUE2Entity< ComputingManagerAttributes >	291
Arc::ComputingManagerType	157
Arc::GLUE2Entity< ComputingServiceAttributes >	291
Arc::ComputingServiceType	162
Arc::GLUE2Entity< ComputingShareAttributes >	291
Arc::ComputingShareType	165
Arc::GLUE2Entity< ExecutionEnvironmentAttributes >	291
Arc::ExecutionEnvironmentType	274
Arc::GLUE2Entity< LocationAttributes >	291
Arc::LocationType	362
Arc::GSSCredential	292
Arc::FileAccess::header_t	294
Arc::HTTPClientInfo	295
Arc::InfoCache	296
Arc::InfoFilter	298
Arc::InfoRegister	299
Arc::InfoRegisterContainer	300
Arc::InfoRegisters	301
Arc::InfoRegistrar	302
Arc::InformationInterface	305
Arc::InfoCacheInterface	297
Arc::InformationContainer	303
Arc::InformationRequest	307
Arc::InformationResponse	308
Arc::initializeCredentialsType	310
Arc::InputFileType	311
Arc::InterruptGuard	313
Arc::ISIS_description	318
Arc::IString	319
Arc::JobDescriptionParserPluginLoader::iterator	320
Arc::Job	321
Arc::JobControllerPluginTestACCCControl	329
Arc::JobDescription	330
Arc::JobDescriptionParserPluginResult	336
Arc::JobDescriptionParserPluginTestACCCControl	337

Arc::JobDescriptionResult	338
Arc::JobIdentificationType	339
Arc::JobInformationStorage	341
Arc::JobInformationStorageXML	345
Arc::JobListRetrieverPluginTESTControl	349
Arc::JobState	350
Arc::JobStateTEST	352
list	359
Arc::EntityContainer< ComputingServiceType >	253
Arc::Loader	360
Arc::BrokerPluginLoader	126
Arc::EntityRetrieverPluginLoader< T >	261
Arc::EntityRetriever< T >::Common	153
Arc::JobControllerPluginLoader	327
Arc::JobDescriptionParserPluginLoader	334
Arc::MCCLoader	383
Arc::SubmitterPluginLoader	546
Arc::LocationAttributes	361
Arc::LogDestination	363
Arc::LogFile	364
Arc::LogStream	373
Arc::Logger	366
Arc::LoggerFormat	370
Arc::LogMessage	371
Arc::MCC_Status	379
Arc::Message	388
Arc::MessageAttributes	391
Arc::MessageAuth	394
Arc::MessageAuthContext	395
Arc::MessageContext	396
Arc::MessageContextElement	397
ArcSec::PDPCConfigContext	437
Arc::MessagePayload	398
Arc::PayloadRawInterface	426
Arc::PayloadRaw	422
Arc::PayloadSOAP	428
Arc::PayloadStreamInterface	432
Arc::PayloadStream	429
Arc::PayloadWSRF	435
Arc::PluginsFactory::modules_t::miterator	399
Arc::ModuleDesc	400
Arc::ModuleManager	401
Arc::PluginsFactory	451
Arc::ClassLoader	140
Arc::NotificationType	409
Arc::NS	410
Arc::OptIn< T >	414
Arc::OptionParser	415
Arc::OutputFileType	419
Arc::ParallelEnvironmentType	420

Arc::PathIterator	421
Arc::PayloadRawBuf	425
Arc::Period	439
Arc::PlexerEntry	445
Arc::Plugin	446
Arc::BrokerPlugin	124
Arc::EntityRetrieverPlugin< T >	260
Arc::JobControllerPlugin	325
Arc::JobDescriptionParserPlugin	333
Arc::MCCInterface	382
Arc::MCC	376
Arc::Plexer	443
Arc::Service	506
Arc::RegisteredService	470
Arc::SubmitterPlugin	543
ArcSec::AlgFactory	98
ArcSec::AttributeFactory	108
ArcSec::Evaluator	266
ArcSec::FnFactory	286
ArcSec::PDP	436
ArcSec::Policy	454
ArcSec::Request	473
ArcSec::SecHandler	501
Arc::EntityRetrieverPlugin< ComputingServiceType >	260
Arc::TargetInformationRetrieverPlugin	549
Arc::EntityRetrieverPlugin< Endpoint >	260
Arc::ServiceEndpointRetrieverPlugin	509
Arc::EntityRetrieverPlugin< Job >	260
Arc::JobListRetrieverPlugin	348
Arc::PluginArgument	447
Arc::BrokerPluginArgument	125
Arc::ClassLoaderPluginArgument	141
Arc::JobControllerPluginArgument	326
Arc::MCCPluginArgument	385
Arc::ServicePluginArgument	511
Arc::SubmitterPluginArgument	545
ArcSec::PDPPluginArgument	438
ArcSec::SecHandlerPluginArgument	504
Arc::PluginDesc	449
Arc::PluginDescriptor	450
ArcSec::PolicyStore::PolicyElement	457
ArcSec::PolicyParser	458
ArcSec::PolicyStore	459
AuthN::PrivateKeyInfoCodec	460
ArcCredential::PROXYCERTINFO_st	464
ArcCredential::PROXYPOLICY_st	465
Arc::Query	466
Arc::MySQLQuery	407
Arc::Range< T >	468
Arc::Register_Info_Type	469
Arc::RegularExpression	471
Arc::RemoteLoggingType	472

ArcSec::RequestAttribute	475
ArcSec::RequestItem	476
ArcSec::RequestTuple	477
Arc::ResourcesType	478
ArcSec::Response	479
ArcSec::ResponseItem	480
ArcSec::ResponseList	481
Arc::Run	483
Arc::SAML2LoginClient	487
Arc::OAuthConsumer	411
Arc::SAML2SSOHTTPClient	488
Arc::HakaClient	293
Arc::OpenIdpClient	413
Arc::SAMLToken	490
Arc::ScalableTime< T >	493
Arc::ScalableTime< int >	494
Arc::SecAttr	497
Arc::MultiSecAttr	404
Arc::SecAttrFormat	499
Arc::SecAttrValue	500
Arc::CIStrngValue	138
ArcSec::Security	505
Arc::ServiceEndpointRetrieverPluginTESTControl	510
Arc::SharedMutex	512
Arc::SimpleCondition	513
Arc::SimpleCounter	515
Arc::SlotRequirementType	517
Arc::SOAPMessage	518
Arc::Software	520
Arc::ApplicationEnvironment	100
Arc::SoftwareRequirement	528
ArcSec::Source	535
ArcSec::SourceFile	536
ArcSec::SourceURL	538
DataStaging::DataDeliveryComm::Status	539
Arc::SubmissionStatus	541
Arc::Submitter	542
Arc::SubmitterPluginTestACCCControl	548
Arc::TargetInformationRetrieverPluginTESTControl	550
Arc::TCPSec	552
Arc::EntityRetriever< T >::ThreadArg	553
Arc::ThreadDataItem	554
Arc::ThreadedPointer< T >	556
Arc::ThreadedPointer< SimpleCounter >	556
Arc::EntityRetriever< T >::Result	482
Arc::ThreadInitializer	558
Arc::ThreadRegistry	559
Arc::Time	560
Arc::TimedMutex	563
DataStaging::TransferParameters	564
DataStaging::TransferShares	565
DataStaging::TransferSharesConf	566

Arc::URL	567
Arc::SourceType	537
Arc::TargetType	551
Arc::URLLocation	573
Arc::User	574
Arc::UserConfig	576
Arc::UsernameToken	604
Arc::UserSwitch	606
Arc::VOMSACInfo	607
Arc::VOMSTrustList	608
Arc::WatchdogChannel	610
Arc::WatchdogListener	611
Arc::WSAEndpointReference	612
Arc::WSAHeader	614
Arc::WSRF	618
Arc::WSRFBaseFault	620
Arc::WSRFResourceUnavailableFault	621
Arc::WSRFResourceUnknownFault	622
Arc::WSRPFault	629
Arc::WSRPInvalidResourcePropertyQNameFault	641
Arc::WSRPResourcePropertyChangeFailure	647
Arc::WSRPDeleteResourcePropertiesRequestFailedFault	627
Arc::WSRPInsertResourcePropertiesRequestFailedFault	638
Arc::WSRPInvalidModificationFault	640
Arc::WSRPSetResourcePropertyRequestFailedFault	650
Arc::WSRPUnableToModifyResourcePropertyFault	651
Arc::WSRPUnableToPutResourcePropertyDocumentFault	652
Arc::WSRPUpdateResourcePropertiesRequestFailedFault	655
Arc::WSRP	623
Arc::WSRPDeleteResourcePropertiesRequest	626
Arc::WSRPDeleteResourcePropertiesResponse	628
Arc::WSRPGetMultipleResourcePropertiesRequest	630
Arc::WSRPGetMultipleResourcePropertiesResponse	631
Arc::WSRPGetResourcePropertyDocumentRequest	632
Arc::WSRPGetResourcePropertyDocumentResponse	633
Arc::WSRPGetResourcePropertyRequest	634
Arc::WSRPGetResourcePropertyResponse	635
Arc::WSRPInsertResourcePropertiesRequest	637
Arc::WSRPInsertResourcePropertiesResponse	639
Arc::WSRPPutResourcePropertyDocumentRequest	643
Arc::WSRPPutResourcePropertyDocumentResponse	644
Arc::WSRPQueryResourcePropertiesRequest	645
Arc::WSRPQueryResourcePropertiesResponse	646
Arc::WSRPSetResourcePropertiesRequest	648
Arc::WSRPSetResourcePropertiesResponse	649
Arc::WSRPUpdateResourcePropertiesRequest	654
Arc::WSRPUpdateResourcePropertiesResponse	656
Arc::WSRPModifyResourceProperties	642
Arc::WSRPDeleteResourceProperties	625
Arc::WSRPInsertResourceProperties	636
Arc::WSRPUpdateResourceProperties	653
Arc::X509Token	658

Arc::XmlContainer	660
Arc::XmlDatabase	661
Arc::XMLNode	662
Arc::Config	166
Arc::IniConfig	309
Arc::Profile	463
Arc::SecHandlerConfig	503
Arc::ARCPolicyHandlerConfig	105
Arc::DNListHandlerConfig	222
Arc::XMLSecNode	670
ArcSec::SecHandlerConfig	502
Arc::XMLNodeContainer	669

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

ArcCredential::ACACI	77
ArcCredential::ACATTHOLDER	78
ArcCredential::ACATTR	79
ArcCredential::ACATTRIBUTE	80
ArcCredential::ACC	81
ArcCredential::ACCERTS	82
ArcCredential::ACDIGEST	83
ArcCredential::ACFORM	84
ArcCredential::ACFULLATTRIBUTES	85
ArcCredential::ACHOLDER	86
ArcCredential::ACIETFATTR	87
ArcCredential::ACINFO	88
ArcCredential::ACIS	89
ArcCredential::ACSEQ	90
ArcCredential::ACTARGET	91
ArcCredential::ACTARGETS	92
ArcCredential::ACVAL	93
Arc::Adler32Sum (Implementation of Adler32 checksum)	94
Arc::AdminDomainAttributes	96
Arc::AdminDomainType	97
ArcSec::AlgFactory (Interface for algorithm factory class)	98
ArcSec::AnyURIAttribute	99
Arc::ApplicationEnvironment (ApplicationEnvironment)	100
Arc::ApplicationType	101
Arc::ArcLocation (Determines ARC installation location)	103
ArcSec::ArcPeriod	104
Arc::ARCPolicyHandlerConfig	105
Arc::ArcVersion (Determines ARC HED libraries version at runtime)	106
ArcSec::Attr (Attr contains a tuple of attribute type and value)	107
ArcSec::AttributeFactory	108
Arc::AttributeIterator (A const iterator class for accessing multiple values of an attribute)	109
ArcSec::AttributeProxy (Interface for creating the AttributeValue object, it will be used by AttributeFactory)	112

ArcSec::AttributeValue (Interface for containing different type of <Attribute> node for both policy and request)	113
ArcSec::Attrs (Attrs is a container for one or more Attr)	115
ArcSec::AuthzRequest	116
ArcSec::AuthzRequestSection	117
Arc::AutoPointer< T > (Wrapper for pointer with automatic destruction)	118
Arc::Base64 (Base64 encoding and decoding, borrowed from Axis2c project)	119
Arc::BaseConfig (Configuration for client interface)	120
ArcSec::BooleanAttribute	122
Arc::Broker	123
Arc::BrokerPlugin	124
Arc::BrokerPluginArgument	125
Arc::BrokerPluginLoader	126
Arc::BrokerPluginTestACCCControl	127
ArcCredential::cert_verify_context	128
Arc::CertEnvLocker (Class for handling X509* variables in a multi-threaded environment)	129
AuthN::certInfo	130
Arc::ChainContext (Interface to chain specific functionality)	131
Arc::Checksum (Interface for checksum manipulations)	132
Arc::ChecksumAny (Wrapper for Checksum class)	135
Arc::CIStrngValue (This class implements case insensitive strings as security attributes)	138
Arc::ClassLoader	140
Arc::ClassLoaderPluginArgument	141
Arc::ClientHTTP (Class for setting up a MCC chain for HTTP communication)	142
Arc::ClientHTTPAttributes (Proxy class for handling request parameters)	143
Arc::ClientHTTPwithSAML2SSO	144
Arc::ClientInterface (Utility base class for MCC)	145
Arc::ClientSOAP	146
Arc::ClientSOAPwithSAML2SSO	148
Arc::ClientTCP (Class for setting up a MCC chain for TCP communication)	149
Arc::ClientX509Delegation	150
ArcSec::CombiningAlg (Interface for combining algorithm)	152
Arc::EntityRetriever< T >::Common	153
Arc::ComputingEndpointAttributes	154
Arc::ComputingEndpointType	155
Arc::ComputingManagerAttributes	156
Arc::ComputingManagerType	157
Arc::ComputingServiceAttributes	158
Arc::ComputingServiceRetriever (Retrieves information about computing elements by querying service registries and CE information systems)	159
Arc::ComputingServiceType	162
Arc::ComputingServiceUniq	163
Arc::ComputingShareAttributes	164
Arc::ComputingShareType	165
Arc::Config (Configuration element - represents (sub)tree of ARC XML configuration)	166
Arc::ConfigEndpoint (Represents the endpoint of service with a given type and GLUE2 InterfaceName)	168
Arc::ConfusaCertHandler	170
Arc::ConfusaParserUtils	171
Arc::CountedPointer< T > (Wrapper for pointer with automatic destruction and multiple references)	173
Arc::Counter (A class defining a common interface for counters)	174
Arc::CounterTicket (A class for "tickets" that correspond to counter reservations)	181
Arc::CRC32Sum (Implementation of CRC32 checksum)	183

Arc::Credential	185
Arc::CredentialError	194
Arc::CredentialStore	195
Arc::Database (Interface for calling database client library)	196
DataStaging::DataDelivery (DataDelivery transfers data between specified physical locations)	198
DataStaging::DataDeliveryComm (This class provides an abstract interface for the Delivery layer)	199
DataStaging::DataDeliveryCommHandler (Singleton class handling all active DataDeliveryComm objects)	202
DataStaging::DataDeliveryLocalComm (This class starts, monitors and controls a local Delivery process)	203
DataStaging::DataDeliveryRemoteComm (This class contacts a remote service to make a Delivery request)	204
Arc::DataStagingType	205
ArcSec::DateAttribute	206
ArcSec::DateTimeAttribute	207
Arc::DelegationConsumer	208
Arc::DelegationConsumerSOAP	210
Arc::DelegationContainerSOAP	212
Arc::DelegationProvider	215
Arc::DelegationProviderSOAP	216
ArcSec::DenyOverridesCombiningAlg (Implement the "Deny-Overrides" algorithm)	218
Arc::DiskSpaceRequirementType	220
Arc::PluginsFactory::modules_t::diterator	221
Arc::DNListHandlerConfig	222
DataStaging::DTR (Data Transfer Request)	223
DataStaging::DTRCacheParameters (The configured cache directories)	227
DataStaging::DTRCallback (The base class from which all callback-enabled classes should be derived)	228
DataStaging::DTRErrorStatus (A class to represent error states reported by various components)	229
DataStaging::DTRList (Global list of all active DTRs in the system)	231
DataStaging::DTRStatus (Class representing the status of a DTR)	234
ArcSec::DurationAttribute	236
Arc::Endpoint (Represents an endpoint of a service with a given interface type and capabilities)	237
Arc::EndpointQueryingStatus (Represents the status in the EntityRetriever of the query process of an Endpoint (service registry, computing element))	242
Arc::EndpointQueryOptions< T > (Options controlling the query process)	246
Arc::EndpointQueryOptions< Endpoint > (The EntityRetriever<Endpoint> (a.k.a. ServiceEndpointRetriever) needs different options)	247
Arc::EndpointStatusMap	248
Arc::EndpointSubmissionStatus	249
Arc::EntityConsumer< T > (A general concept of an object which can consume entities use by the retrievers to return results)	252
Arc::EntityContainer< T > (An entity consumer class storing all the consumed entities in a list)	253
Arc::EntityRetriever< T > (Queries Endpoint objects (using plugins in parallel) and sends the found entities to consumers)	254
Arc::EntityRetrieverPlugin< T >	260
Arc::EntityRetrieverPluginLoader< T >	261
Arc::EnvLockWrapper (Class to provide automatic locking/unlocking of environment on creation/destruction)	262
ArcSec::EqualFunction (Evaluate whether the two values are equal)	263
ArcSec::EvalResult (Struct to record the xml node and effect, which will be used by Evaluator to get the information about which rule/policy(in xmlnode) is satisfied)	264
ArcSec::EvaluationCtx (EvaluationCtx , in charge of storing some context information for)	265

ArcSec::Evaluator (Interface for policy evaluation. Execute the policy evaluation, based on the request and policy)	266
ArcSec::EvaluatorContext (Context for evaluator. It includes the factories which will be used to create related objects)	269
ArcSec::EvaluatorLoader (EvaluatorLoader is implemented as a helper class for loading different Evaluator objects, like ArcEvaluator)	270
Arc::ExecutableType (Executable)	272
Arc::ExecutionEnvironmentAttributes	273
Arc::ExecutionEnvironmentType	274
Arc::ExecutionTarget (ExecutionTarget)	275
Arc::ExecutionTargetSorter	277
Arc::ExpirationReminder (A class intended for internal use within counters)	278
Arc::FileAccess (Defines interface for accessing filesystems)	279
Arc::FileAccessContainer (Container for shared FileAccess objects)	281
Arc::FileLock (A general file locking class)	282
Arc::FinderLoader	285
ArcSec::FnFactory (Interface for function factory class)	286
ArcSec::Function (Interface for function, which is in charge of evaluating two AttributeValue)	287
ArcSec::GenericAttribute	288
Arc::GlobusResult	289
Arc::GLUE2 (GLUE2 parser)	290
Arc::GLUE2Entity< T >	291
Arc::GSSCredential	292
Arc::HakaClient	293
Arc::FileAccess::header_t (Internal struct used for communication between processes)	294
Arc::HTTPClientInfo	295
Arc::InfoCache (Stores XML document in filesystem split into parts)	296
Arc::InfoCacheInterface	297
Arc::InfoFilter (Filters information document according to identity of requestor)	298
Arc::InfoRegister (Registration to Information Indexing Service)	299
Arc::InfoRegisterContainer	300
Arc::InfoRegisters (Handling registrations to multiple Information Indexing Services)	301
Arc::InfoRegistrar (Registration process associated with particular ISIS)	302
Arc::InformationContainer (Information System document container and processor)	303
Arc::InformationInterface (Information System message processor)	305
Arc::InformationRequest (Request for information in InfoSystem)	307
Arc::InformationResponse (Informational response from InfoSystem)	308
Arc::IniConfig (Class representing "ini-style" configuration)	309
Arc::initializeCredentialsType (Defines how user credentials are looked for)	310
Arc::InputFileType	311
ArcSec::InRangeFunction	312
Arc::InterruptGuard (Marks off a section of code which should not be interrupted by signals)	313
Arc::IntraProcessCounter (A class for counters used by threads within a single process)	314
Arc::ISIS_description	318
Arc::IString (Class used for localised output of log messages)	319
Arc::JobDescriptionParserPluginLoader::iterator	320
Arc::Job (Job)	321
Arc::JobControllerPlugin	325
Arc::JobControllerPluginArgument	326
Arc::JobControllerPluginLoader	327
Arc::JobControllerPluginTestACCCControl	329
Arc::JobDescription	330
Arc::JobDescriptionParserPlugin (Abstract class for the different parsers)	333
Arc::JobDescriptionParserPluginLoader	334

Arc::JobDescriptionParserPluginResult	336
Arc::JobDescriptionParserPluginTestACCCControl	337
Arc::JobDescriptionResult	338
Arc::JobIdentificationType (Job identification)	339
Arc::JobInformationStorage (Abstract class for storing job information)	341
Arc::JobInformationStorageXML	345
Arc::JobListRetrieverPlugin	348
Arc::JobListRetrieverPluginTESTControl	349
Arc::JobState	350
Arc::JobStateTEST	352
Arc::JobSupervisor (JobSupervisor class)	353
list	359
Arc::Loader (Plugins loader)	360
Arc::LocationAttributes	361
Arc::LocationType	362
Arc::LogDestination (A base class for log destinations)	363
Arc::LogFile (A class for logging to files)	364
Arc::Logger (A logger class)	366
Arc::LoggerFormat (Struct to contain LogFormat, to use with <code>operator<<(std::ostream&, const LoggerFormat&)</code>)	370
Arc::LogMessage (A class for log messages)	371
Arc::LogStream (A class for logging to ostreams)	373
ArcSec::MatchFunction (Evaluate whether arg1 (value in regular expression) matched arg0 (lable in regular expression))	375
Arc::MCC (Message Chain Component - base class for every MCC plugin)	376
Arc::MCC_Status (A class for communication of MCC processing results)	379
Arc::MCCConfig	381
Arc::MCCInterface (Interface for communication between MCC, Service and Plexer objects)	382
Arc::MCCLoader (Creator of Message Component Chains (MCC))	383
Arc::MCCPluginArgument	385
Arc::MD5Sum (Implementation of MD5 checksum)	386
Arc::Message (Object being passed through chain of MCCs)	388
Arc::MessageAttributes (A class for storage of attribute values)	391
Arc::MessageAuth (Contains authenticity information, authorization tokens and decisions)	394
Arc::MessageAuthContext (Handler for content of message auth* context)	395
Arc::MessageContext (Handler for content of message context)	396
Arc::MessageContextElement (Top class for elements contained in message context)	397
Arc::MessagePayload (Base class for content of message passed through chain)	398
Arc::PluginsFactory::modules_t::miterator	399
Arc::ModuleDesc (Description of loadable module)	400
Arc::ModuleManager (Manager of shared libraries)	401
Arc::MultiSecAttr (Container of multiple SecAttr attributes)	404
Arc::MySQLDatabase (Implements a MySQL version of the Database interface)	405
Arc::MySQLQuery (Implements a MySQL version of the Query database query class)	407
Arc::NotificationType	409
Arc::NS (Class to represent an XML namespace)	410
Arc::OAuthConsumer	411
Arc::OpenIdpClient	413
Arc::OptIn< T >	414
Arc::OptionParser (Command line option parser used by ARC command line tools)	415
ArcSec::OrderedCombiningAlg	418
Arc::OutputFileType	419
Arc::ParallelEnvironmentType	420
Arc::PathIterator (Class to iterate through elements of a path)	421

Arc::PayloadRaw (Raw byte multi-buffer)	422
Arc::PayloadRawBuf	425
Arc::PayloadRawInterface (Random Access Payload for Message objects)	426
Arc::PayloadSOAP (Payload of Message with SOAP content)	428
Arc::PayloadStream (POSIX handle as Payload)	429
Arc::PayloadStreamInterface (Stream-like Payload for Message object)	432
Arc::PayloadWSRF (This class combines MessagePayload with WSRF)	435
ArcSec::PDP (Base class for Policy Decision Point plugins)	436
ArcSec::PDPCfgContext	437
ArcSec::PDPPuginArgument	438
Arc::Period (A Period represents a length of time)	439
ArcSec::PeriodAttribute	440
ArcSec::PermitOverridesCombiningAlg (Implement the "Permit-Overrides" algorithm)	441
Arc::Plexer (The Plexer class, used for routing messages to services)	443
Arc::PlexerEntry (A pair of label (regex) and pointer to MCC)	445
Arc::Plugin (Base class for loadable ARC components)	446
Arc::PluginArgument (Base class for passing arguments to loadable ARC components)	447
Arc::PluginDesc (Description of plugin)	449
Arc::PluginDescriptor (Description of ARC lodable component)	450
Arc::PluginsFactory (Generic ARC plugins loader)	451
ArcSec::Policy (Interface for containing and processing different types of policy)	454
ArcSec::PolicyStore::PolicyElement	457
ArcSec::PolicyParser (A interface which will isolate the policy object from actual policy storage (files, urls, database))	458
ArcSec::PolicyStore (Storage place for policy objects)	459
AuthN::PrivateKeyInfoCodec	460
DataStaging::Processor (The Processor performs pre- and post-transfer operations)	461
Arc::Profile (Class used to convert human-friendly ini-style configuration to XML)	463
ArcCredential::PROXYCERTINFO_st	464
ArcCredential::PROXYPOLICY_st	465
Arc::Query (Class representing a database query)	466
Arc::Range< T >	468
Arc::Register_Info_Type	469
Arc::RegisteredService (RegisteredService - extension of Service performing self-registration)	470
Arc::RegularExpression (A regular expression class)	471
Arc::RemoteLoggingType (Remote logging)	472
ArcSec::Request (Base class/Interface for request, includes a container for RequestItems and some operations)	473
ArcSec::RequestAttribute (Wrapper which includes AttributeValue object which is generated ac- cording to date type of one specfic node in Request.xml)	475
ArcSec::RequestItem (Interface for request item container, <subjects, actions, objects, ctxs> tuple)	476
ArcSec::RequestTuple	477
Arc::ResourcesType	478
ArcSec::Response (Container for the evaluation results)	479
ArcSec::ResponseItem (Evaluation result concerning one RequestTuple)	480
ArcSec::ResponseList	481
Arc::EntityRetriever< T >::Result	482
Arc::Run (This class runs an external executable)	483
Arc::SAML2LoginClient	487
Arc::SAML2SSOHTTPClient	488
Arc::SAMLToken (Class for manipulating SAML Token Profile)	490
Arc::ScalableTime< T >	493
Arc::ScalableTime< int >	494

DataStaging::Scheduler (The Scheduler is the control centre of the data staging framework) . .	495
Arc::SecAttr (This is an abstract interface to a security attribute)	497
Arc::SecAttrFormat (Export/import format)	499
Arc::SecAttrValue (This is an abstract interface to a security attribute)	500
ArcSec::SecHandler (Base class for simple security handling plugins)	501
ArcSec::SecHandlerConfig	502
Arc::SecHandlerConfig	503
ArcSec::SecHandlerPluginArgument	504
ArcSec::Security (Common stuff used by security related classes)	505
Arc::Service (Service - last component in a Message Chain)	506
Arc::ServiceEndpointRetrieverPlugin	509
Arc::ServiceEndpointRetrieverPluginTESTControl	510
Arc::ServicePluginArgument	511
Arc::SharedMutex (Mutex which allows shared and exclusive locking)	512
Arc::SimpleCondition (Simple triggered condition)	513
Arc::SimpleCounter (Thread-safe counter with capability to wait for zero value)	515
Arc::SlotRequirementType	517
Arc::SOAPMessage (Message restricted to SOAP payload)	518
Arc::Software (Used to represent software (names and version) and comparison)	520
Arc::SoftwareRequirement (Class used to express and resolve version requirements on software)	528
ArcSec::Source (Acquires and parses XML document from specified source)	535
ArcSec::SourceFile (Convenience class for obtaining XML document from file)	536
Arc::SourceType	537
ArcSec::SourceURL (Convenience class for obtaining XML document from remote URL)	538
DataStaging::DataDeliveryComm::Status (Plain C struct to pass information from executing process back to main thread)	539
ArcSec::StringAttribute	540
Arc::SubmissionStatus	541
Arc::Submitter	542
Arc::SubmitterPlugin (Base class for the SubmitterPlugins)	543
Arc::SubmitterPluginArgument	545
Arc::SubmitterPluginLoader	546
Arc::SubmitterPluginTestACCCControl	548
Arc::TargetInformationRetrieverPlugin	549
Arc::TargetInformationRetrieverPluginTESTControl	550
Arc::TargetType	551
Arc::TCPsec	552
Arc::EntityRetriever< T >::ThreadArg	553
Arc::ThreadDataItem (Base class for per-thread object)	554
Arc::ThreadedPointer< T > (Wrapper for pointer with automatic destruction and multiple references)	556
Arc::ThreadInitializer (This class initializes the glibmm thread system)	558
Arc::ThreadRegistry (A set of conditions, mutexes, etc. conveniently exposed to monitor running child threads and to wait till they exit)	559
Arc::Time (A class for storing and manipulating times)	560
ArcSec::TimeAttribute	562
Arc::TimedMutex (Mutex which allows a timeout on locking)	563
DataStaging::TransferParameters (Represents limits and properties of a DTR transfer. These generally apply to all DTRs)	564
DataStaging::TransferShares (TransferShares is used to implement fair-sharing and priorities)	565
DataStaging::TransferSharesConf (TransferSharesConf describes the configuration of TransferShares)	566
Arc::URL (Class to represent general URLs)	567
Arc::URLLocation (Class to hold a resolved URL location)	573

Arc::User (Platform independent representation of system user)	574
Arc::UserConfig (User configuration class)	576
Arc::UsernameToken (Interface for manipulation of WS-Security according to Username Token Profile)	604
Arc::UserSwitch (Class for temporary switching of user id)	606
Arc::VOMSACInfo	607
Arc::VOMSTrustList	608
Arc::WatchdogChannel (This class is meant to be used in code which provides "I'm alive" ticks to watchdog)	610
Arc::WatchdogListener (This class is meant to provide interface for Watchdog executor part) . .	611
Arc::WSAEndpointReference (Interface for manipulation of WS-Adressing Endpoint Reference)	612
Arc::WSAHeader (Interface for manipulation WS-Addressing information in SOAP header) . .	614
Arc::WSRF (Base class for every WSRF message)	618
Arc::WSRFBaseFault (Base class for WSRF fault messages)	620
Arc::WSRFResourceUnavailableFault	621
Arc::WSRFResourceUnknownFault	622
Arc::WSRP (Base class for WS-ResourceProperties structures)	623
Arc::WSRPDeleteResourceProperties	625
Arc::WSRPDeleteResourcePropertiesRequest	626
Arc::WSRPDeleteResourcePropertiesRequestFailedFault	627
Arc::WSRPDeleteResourcePropertiesResponse	628
Arc::WSRPFault (Base class for WS-ResourceProperties faults)	629
Arc::WSRPGetMultipleResourcePropertiesRequest	630
Arc::WSRPGetMultipleResourcePropertiesResponse	631
Arc::WSRPGetResourcePropertyDocumentRequest	632
Arc::WSRPGetResourcePropertyDocumentResponse	633
Arc::WSRPGetResourcePropertyRequest	634
Arc::WSRPGetResourcePropertyResponse	635
Arc::WSRPInsertResourceProperties	636
Arc::WSRPInsertResourcePropertiesRequest	637
Arc::WSRPInsertResourcePropertiesRequestFailedFault	638
Arc::WSRPInsertResourcePropertiesResponse	639
Arc::WSRPInvalidModificationFault	640
Arc::WSRPInvalidResourcePropertyQNameFault	641
Arc::WSRPModifyResourceProperties	642
Arc::WSRPPutResourcePropertyDocumentRequest	643
Arc::WSRPPutResourcePropertyDocumentResponse	644
Arc::WSRPQueryResourcePropertiesRequest	645
Arc::WSRPQueryResourcePropertiesResponse	646
Arc::WSRPResourcePropertyChangeFailure	647
Arc::WSRPSetResourcePropertiesRequest	648
Arc::WSRPSetResourcePropertiesResponse	649
Arc::WSRPSetResourcePropertyRequestFailedFault	650
Arc::WSRPUnableToModifyResourcePropertyFault	651
Arc::WSRPUnableToPutResourcePropertyDocumentFault	652
Arc::WSRPUpdateResourceProperties	653
Arc::WSRPUpdateResourcePropertiesRequest	654
Arc::WSRPUpdateResourcePropertiesRequestFailedFault	655
Arc::WSRPUpdateResourcePropertiesResponse	656
ArcSec::X500NameAttribute	657
Arc::X509Token (Class for manipulating X.509 Token Profile)	658
Arc::XmlContainer	660
Arc::XmlDatabase	661
Arc::XMLNode (Wrapper for LibXML library Tree interface)	662

Arc::XMLNodeContainer (Container for multiple XMLNode elements)	669
Arc::XMLSecNode (Extends XMLNode class to support XML security operation)	670

Chapter 7

File Index

7.1 File List

Here is a list of all documented files with brief descriptions:

AlgFactory.h	??
AnyURIAttribute.h	??
ArcConfig.h	??
ArcLocation.h	??
ArcRegex.h	??
ArcVersion.h	??
AttributeFactory.h	??
AttributeProxy.h	??
AttributeValue.h	??
Base64.h	??
BooleanAttribute.h	??
Broker.h	??
BrokerPlugin.h (Plugin, loader and argument classes for broker specialisation)	673
CertUtil.h	??
Checksum.h	??
CISStringValue.h	??
ClassLoader.h	??
ClientInterface.h	??
ClientSAML2SSO.h	??
ClientX509Delegation.h	??
CombiningAlg.h	??
ComputingServiceRetriever.h	??
ConfusaCertHandler.h	??
ConfusaParserUtils.h	??
Counter.h	??
Credential.h	??
CredentialStore.h	??
DataDelivery.h	??
DataDeliveryComm.h	??
DataDeliveryLocalComm.h	??
DataDeliveryRemoteComm.h	??
DateTime.h	??
DateTimeAttribute.h	??

DBInterface.h	??
DelegationInterface.h	??
DenyOverridesAlg.h	??
deprecated.h	??
DTR.h	??
DTRLst.h	??
DTRStatus.h	??
Endpoint.h	??
EndpointQueryingStatus.h	??
EntityRetriever.h	??
EntityRetrieverPlugin.h (Plugin, loader and argument classes for EntityRetriever specialisation)	674
EqualFunction.h	??
EvaluationCtx.h	??
Evaluator.h	??
EvaluatorLoader.h	??
ExecutionTarget.h (Structures holding resource information)	675
file_access.h	??
FileAccess.h	??
FileLock.h	??
FileUtils.h	??
FinderLoader.h	??
FnFactory.h	??
Function.h	??
Generator.h	??
GenericAttribute.h	??
GlobusErrorUtils.h	??
GlobusWorkarounds.h	??
GLUE2.h	??
GLUE2Entity.h (Template class for GLUE2 entities)	676
GSSCredential.h	??
GUID.h	??
HakaClient.h	??
InfoCache.h	??
InfoFilter.h	??
InfoRegister.h	??
InformationInterface.h	??
IniConfig.h	??
InRangeFunction.h	??
IntraProcessCounter.h	??
IString.h	??
Job.h	??
JobControllerPlugin.h (Plugin, loader and argument classes for job controller specialisation)	677
JobDescription.h (Classes related to creating JobDescription objects)	678
JobDescriptionParserPlugin.h (Plugin, loader and argument classes for job description parser specialisation)	679
JobState.h	??
JobSupervisor.h	??
listfunc.h	??
Loader.h	??
Logger.h	??
MatchFunction.h	??
MCC.h	??
MCC_Status.h	??
MCCLoader.h	??

Message.h	??
MessageAttributes.h	??
MessageAuth.h	??
ModuleManager.h	??
MysqlWrapper.h	??
NSSGetPassword.h	??
nssprivkeyinfocodec.h	??
NSSUtil.h	??
OAuthConsumer.h	??
OpenIdpClient.h	??
OpenSSL.h	??
OptionParser.h	??
OrderedAlg.h	??
PayloadRaw.h	??
PayloadSOAP.h	??
PayloadStream.h	??
PayloadWSRF.h	??
PDP.h	??
PermitOverridesAlg.h	??
Plexer.h	??
Plugin.h	??
Policy.h	??
PolicyParser.h	??
PolicyStore.h	??
Processor.h	??
Profile.h	??
Proxycertinfo.h	??
RegisteredService.h	??
Request.h	??
RequestAttribute.h	??
RequestItem.h	??
Response.h	??
Result.h	??
Run.h	??
SAML2LoginClient.h	??
saml_util.h	??
SAMLToken.h	??
Scheduler.h	??
SecAttr.h	??
SecAttrValue.h	??
SecHandler.h	??
Security.h	??
Service.h	??
SOAPEnvelope.h	??
SOAPMessage.h	??
Software.h (Software and SoftwareRequirement classes)	680
Source.h	??
StringAttribute.h	??
StringConv.h	??
SubmissionStatus.h	??
Submitter.h	??
SubmitterPlugin.h (Plugin, loader and argument classes for submitter specialisation)	681
TestACCCControl.h (Classes for controlling output of compute test plugins)	682
Thread.h	??

TransferShares.h	??
URL.h	??
User.h	??
UserConfig.h	??
UsernameToken.h	??
Utils.h	??
VOMSAttribute.h	??
VOMSUtil.h	??
Watchdog.h	??
win32.h	??
WSA.h	??
WSResourceProperties.h	??
WSRF.h	??
WSRFBaseFault.h	??
X500NameAttribute.h	??
X509Token.h	??
XmlContainer.h	??
XmlDatabase.h	??
XMLNode.h	??
XMLSecNode.h	??
XmlSecUtils.h	??

Chapter 8

Module Documentation

8.1 Structures holding resource information

Data Structures

- class [Arc::ApplicationEnvironment](#)
ApplicationEnvironment.
- class [Arc::LocationAttributes](#)
- class [Arc::AdminDomainAttributes](#)
- class [Arc::ExecutionEnvironmentAttributes](#)
- class [Arc::ComputingManagerAttributes](#)
- class [Arc::ComputingShareAttributes](#)
- class [Arc::ComputingEndpointAttributes](#)
- class [Arc::ComputingServiceAttributes](#)
- class [Arc::LocationType](#)
- class [Arc::AdminDomainType](#)
- class [Arc::ExecutionEnvironmentType](#)
- class [Arc::ComputingManagerType](#)
- class [Arc::ComputingShareType](#)
- class [Arc::ComputingEndpointType](#)
- class [Arc::ComputingServiceType](#)
- class [Arc::ExecutionTarget](#)
ExecutionTarget.
- class [Arc::GLUE2Entity< T >](#)

8.1.1 Detailed Description

The listed structures are all used for holding resource information when doing resource discovery and those structures are read when doing match making.

8.2 JobDescription related classes

Data Structures

- class [Arc::OptIn< T >](#)
- class [Arc::Range< T >](#)
- class [Arc::ScalableTime< T >](#)
- class [Arc::ScalableTime< int >](#)
- class [Arc::JobIdentificationType](#)
Job identification.
- class [Arc::ExecutableType](#)
Executable.
- class [Arc::RemoteLoggingType](#)
Remote logging.
- class [Arc::NotificationType](#)
- class [Arc::ApplicationType](#)
- class [Arc::SlotRequirementType](#)
- class [Arc::DiskSpaceRequirementType](#)
- class [Arc::ParallelEnvironmentType](#)
- class [Arc::ResourcesType](#)
- class [Arc::SourceType](#)
- class [Arc::TargetType](#)
- class [Arc::InputFileType](#)
- class [Arc::OutputFileType](#)
- class [Arc::DataStagingType](#)
- class [Arc::JobDescriptionResult](#)
- class [Arc::JobDescription](#)
- class [Arc::Software](#)
Used to represent software (names and version) and comparison.
- class [Arc::SoftwareRequirement](#)
Class used to express and resolve version requirements on software.

8.2.1 Detailed Description

This list of classes is used to make up the structure of the [JobDescription](#) class.

8.3 ARC Compute Library (libarccompute)

Data Structures

- class [Arc::Broker](#)
- class [Arc::ExecutionTargetSorter](#)
- class [Arc::ComputingServiceUniq](#)
- class [Arc::ComputingServiceRetriever](#)

Retrieves information about computing elements by querying service registries and CE information systems.

- class [Arc::EndpointStatusMap](#)
- class [Arc::Endpoint](#)

Represents an endpoint of a service with a given interface type and capabilities.

- class [Arc::EndpointQueryingStatus](#)

Represents the status in the [EntityRetriever](#) of the query process of an [Endpoint](#) (service registry, computing element).

- class [Arc::EntityConsumer< T >](#)

A general concept of an object which can consume entities use by the retrievers to return results.

- class [Arc::EntityContainer< T >](#)

An entity consumer class storing all the consumed entities in a list.

- class [Arc::EntityRetriever< T >](#)

Queries [Endpoint](#) objects (using plugins in parallel) and sends the found entities to consumers.

- class [Arc::ComputingServiceType](#)
- class [Arc::ExecutionTarget](#)

[ExecutionTarget](#).

- class [Arc::Job](#)

[Job](#).

- class [Arc::JobInformationStorage](#)

Abstract class for storing job information.

- class [Arc::JobDescriptionResult](#)
- class [Arc::JobDescription](#)
- class [Arc::JobState](#)
- class [Arc::JobSupervisor](#)

[JobSupervisor](#) class.

- class [Arc::SubmissionStatus](#)
- class [Arc::EndpointSubmissionStatus](#)
- class [Arc::Submitter](#)

Modules

- Structures holding resource information
- JobDescription related classes
- Plugin related classes for compute specialisations
- Classes for controlling output of compute test plugins

Typedefs

- typedef EntityRetriever< Endpoint > [Arc::ServiceEndpointRetriever](#)
- typedef EntityRetriever< ComputingServiceType > [Arc::TargetInformationRetriever](#)
- typedef EntityRetriever< Job > [Arc::JobListRetriever](#)

8.3.1 Typedef Documentation

8.3.1.1 typedef EntityRetriever<Job> Arc::JobListRetriever

The JobListRetriever is an [EntityRetriever](#) retrieving [Job](#) objects. It queries computing elements to get the list of jobs residing on the resource.

8.3.1.2 typedef EntityRetriever<Endpoint> Arc::ServiceEndpointRetriever

The ServiceEndpointRetriever is an [EntityRetriever](#) retrieving [Endpoint](#) objects. It queries service registries to get endpoints of registered services.

8.3.1.3 typedef EntityRetriever<ComputingServiceType> Arc::TargetInformationRetriever

The TargetInformationRetriever is an [EntityRetriever](#) retrieving [ComputingServiceType](#) objects. It queries computing elements to get the full [GLUE2](#) information about the resource.

8.4 Plugin related classes for compute specialisations

Data Structures

- class [Arc::BrokerPluginArgument](#)
- class [Arc::BrokerPlugin](#)
- class [Arc::BrokerPluginLoader](#)
- class [Arc::EntityRetrieverPlugin< T >](#)
- class [Arc::EntityRetrieverPluginLoader< T >](#)
- class [Arc::ServiceEndpointRetrieverPlugin](#)
- class [Arc::TargetInformationRetrieverPlugin](#)
- class [Arc::JobListRetrieverPlugin](#)
- class [Arc::JobControllerPlugin](#)
- class [Arc::JobControllerPluginLoader](#)
- class [Arc::JobControllerPluginArgument](#)
- class [Arc::JobDescriptionParserPluginResult](#)
- class [Arc::JobDescriptionParserPlugin](#)

Abstract class for the different parsers.

- class [Arc::JobDescriptionParserPluginLoader](#)
- class [Arc::SubmitterPluginLoader](#)
- class [Arc::SubmitterPluginArgument](#)

8.5 Classes for controlling output of compute test plugins

Data Structures

- class [Arc::BrokerPluginTestACCControl](#)
- class [Arc::JobDescriptionParserPluginTestACCControl](#)
- class [Arc::JobControllerPluginTestACCControl](#)
- class [Arc::SubmitterPluginTestACCControl](#)
- class [Arc::JobStateTEST](#)
- class [Arc::JobListRetrieverPluginTESTControl](#)
- class [Arc::ServiceEndpointRetrieverPluginTESTControl](#)
- class [Arc::TargetInformationRetrieverPluginTESTControl](#)

8.5.1 Detailed Description

The listed classes are used for controlling the behaviour of the test plugins. A test plugin can be used for simulating, testing and checking how the compute library behaves and react to different inputs from plugins. Also the test plugins doesn't require a network connection in order to function.

Compute test plugins are available for the following plugin types:

- [BrokerPlugin](#)
- [JobControllerPlugin](#)
- [JobDescriptionParserPlugin](#)
- [SubmitterPlugin](#)
- [ServiceEndpointRetrieverPlugin](#)
- [TargetInformationRetrieverPlugin](#)
- [JobListRetrieverPlugin](#)

They can be loaded by using the associated plugin loader class.

Todo

- * Give examples on how to load and use the test plugins.
- * Add descriptions to test control classes.

8.6 ARC data staging (libarcdatastaging)

Data Structures

- class `DataStaging::DataDelivery`
`DataDelivery` transfers data between specified physical locations.
- class `DataStaging::DataDeliveryComm`
This class provides an abstract interface for the Delivery layer.
- struct `DataStaging::DataDeliveryComm::Status`
Plain C struct to pass information from executing process back to main thread.
- class `DataStaging::DataDeliveryCommHandler`
Singleton class handling all active `DataDeliveryComm` objects.
- class `DataStaging::DataDeliveryLocalComm`
This class starts, monitors and controls a local Delivery process.
- class `DataStaging::DataDeliveryRemoteComm`
This class contacts a remote service to make a Delivery request.
- class `DataStaging::TransferParameters`
Represents limits and properties of a `DTR` transfer. These generally apply to all `DTRs`.
- class `DataStaging::DTRCacheParameters`
The configured cache directories.
- class `DataStaging::DTRCallback`
The base class from which all callback-enabled classes should be derived.
- class `DataStaging::DTR`
Data Transfer Request.
- class `DataStaging::DTRList`
Global list of all active `DTRs` in the system.
- class `DataStaging::DTRStatus`
Class representing the status of a `DTR`.
- class `DataStaging::DTRErrorStatus`
A class to represent error states reported by various components.
- class `DataStaging::Processor`
The `Processor` performs pre- and post-transfer operations.
- class `DataStaging::Scheduler`
The `Scheduler` is the control centre of the data staging framework.
- class `DataStaging::TransferSharesConf`

TransferSharesConf describes the configuration of *TransferShares*.

- class `DataStaging::TransferShares`
TransferShares is used to implement fair-sharing and priorities.

Typedefs

- typedef `Arc::ThreadedPointer< DTR > DataStaging::DTR_ptr`
- typedef `Arc::ThreadedPointer< Arc::Logger > DataStaging::DTRLogger`

Enumerations

- enum `DataStaging::StagingProcesses` {
 `DataStaging::GENERATOR`, `DataStaging::SCHEDULER`, `DataStaging::PRE_PROCESSOR`,
 `DataStaging::DELIVERY`,
 `DataStaging::POST_PROCESSOR` }
- enum `DataStaging::ProcessState` { `DataStaging::INITIATED`, `DataStaging::RUNNING`,
 `DataStaging::TO_STOP`, `DataStaging::STOPPED` }
- enum `DataStaging::CacheState` {
 `DataStaging::CACHEABLE`, `DataStaging::NON_CACHEABLE`, `DataStaging::CACHE_-`
 `ALREADY_PRESENT`, `DataStaging::CACHE_DOWNLOADED`,
 `DataStaging::CACHE_LOCKED`, `DataStaging::CACHE_SKIP`, `DataStaging::CACHE_NOT_-`
 `USED` }

8.6.1 Detailed Description

ARC data staging components form a complete data transfer management system. Whereas data is a library for data access, enabling several types of operation on data files on the Grid using a variety of access protocols, [ARC data staging \(libarcdatastaging\)](#) is a framework for managed data transfer to and from the Grid. The data staging system is designed to run as a persistent process, to execute data transfers on demand. Data transfers are defined and fed into the system, and then notification is given when they complete. No knowledge is required of the internal workings of the Grid, a user only needs to specify URLs representing the source and destination of the transfer.

The system is highly configurable and features an intelligent priority, fair-share and error handling mechanism, as well as the ability to spread data transfer across multiple hosts using ARC's [DataDelivery](#) service. It is used by ARC's Computing Element (A-REX) for pre- and post- job data transfer of input and output files. Note that this system is primarily for data transfer to and from local files and that third-party transfer is not supported. It is designed for the case of pulling or pushing data between the Grid and a local file system, rather than a service for transfer between two Grid storage elements. It is possible to transfer data between two remote endpoints, but all data flows through the client.

The following code snippet shows a very simple example of how to use [libarcdatastaging](#). A [DTR](#) is created which describes the data transfer required. It is passed to the [Scheduler](#) and then the code waits until the [Scheduler](#) calls the callback to notify that the transfer has finished.

```
class MyGenerator : public DTRCallback {
public:
    void receiveDTR(DTR_ptr dtr);
    void run();
};
```



```

private:
    Arc::SimpleCondition cond;
};

void MyGenerator::receiveDTR(DTR_ptr dtr) {
    // DTR received back, so notify waiting condition
    std::cout << "Received DTR " << dtr->get_id() << std::endl;
    cond.signal();
}

void MyGenerator::run() {
    // start Scheduler thread
    Scheduler scheduler;
    scheduler.start();

    // create a DTR
    DTR_ptr dtr(new DTR(source, destination,...));

    // register this callback
    dtr->registerCallback(this,DataStaging::GENERATOR);
    // this line must be here in order to pass the DTR to the Scheduler
    dtr->registerCallback(&scheduler,DataStaging::SCHEDULER);

    // push the DTR to the Scheduler
    DataStaging::DTR::push(dtr, DataStaging::SCHEDULER);

    // wait until callback is called
    cond.wait();
    // DTR is finished, so stop Scheduler
    scheduler.stop();
}

```

For more information see http://wiki.nordugrid.org/index.php/Data_Staging

For more examples on using libarcdatastaging in several languages, see http://wiki.nordugrid.org/index.php/Data_Staging/API

8.6.2 Typedef Documentation

8.6.2.1 typedef Arc::ThreadedPointer<DTR> DataStaging::DTR_ptr

Provides automatic memory management of DTRs and thread-safe destruction.

8.6.2.2 typedef Arc::ThreadedPointer<Arc::Logger> DataStaging::DTRLogger

The DTR's Logger object can be used outside the [DTR](#) object with DTRLogger.

8.6.3 Enumeration Type Documentation

8.6.3.1 enum DataStaging::CacheState

Represents possible cache states of this [DTR](#).

Enumerator:

CACHEABLE Source should be cached.

NON_CACHEABLE Source should not be cached.

CACHE_ALREADY_PRESENT Source is available in cache from before.

CACHE_DOWNLOADED Source has just been downloaded and put in cache.

CACHE_LOCKED Cache file is locked.

CACHE_SKIP Source is cacheable but due to some problem should not be cached.

CACHE_NOT_USED Cache was started but was not used.

8.6.3.2 enum DataStaging::ProcessState

Internal state of StagingProcesses.

Enumerator:

INITIATED Process is ready to start.

RUNNING Process is running.

TO_STOP Process has been instructed to stop.

STOPPED Process has stopped.

8.6.3.3 enum DataStaging::StagingProcesses

Components of the data staging framework.

Enumerator:

GENERATOR Creator of new DTRs and receiver of completed DTRs.

SCHEDULER Controls queues and moves DTRs between other components when necessary.

PRE_PROCESSOR Performs all pre-transfer operations.

DELIVERY Performs physical transfer.

POST_PROCESSOR Performs all post-transfer operations.

Chapter 9

Namespace Documentation

9.1 Arc Namespace Reference

[Arc](#) namespace contains all core ARC classes.

Data Structures

- class [Config](#)
Configuration element - represents (sub)tree of ARC XML configuration.
- class [BaseConfig](#)
Configuration for client interface.
- class [ArcLocation](#)
Determines ARC installation location.
- class [RegularExpression](#)
A regular expression class.
- class [ArcVersion](#)
Determines ARC HED libraries version at runtime.
- class [Base64](#)
[Base64](#) encoding and decoding, borrowed from Axis2c project.
- class [CheckSum](#)
Interface for checksum manipulations.
- class [CRC32Sum](#)
Implementation of CRC32 checksum.
- class [MD5Sum](#)
Implementation of MD5 checksum.
- class [Adler32Sum](#)

Implementation of Adler32 checksum.

- class [ChecksumAny](#)
Wrapper for [Checksum](#) class.
- class [Counter](#)
A class defining a common interface for counters.
- class [CounterTicket](#)
A class for "tickets" that correspond to counter reservations.
- class [ExpirationReminder](#)
A class intended for internal use within counters.
- class [Period](#)
A [Period](#) represents a length of time.
- class [Time](#)
A class for storing and manipulating times.
- class [Database](#)
Interface for calling database client library.
- class [Query](#)
Class representing a database query.
- class [FileAccess](#)
Defines interface for accessing filesystems.
- class [FileAccessContainer](#)
Container for shared [FileAccess](#) objects.
- class [FileLock](#)
A general file locking class.
- class [IniConfig](#)
Class representing "ini-style" configuration.
- class [IntraProcessCounter](#)
A class for counters used by threads within a single process.
- class [IString](#)
Class used for localised output of log messages.
- struct [LoggerFormat](#)
Struct to contain [LogFormat](#), to use with `operator<<(std::ostream&, const LoggerFormat&)`.
- class [LogMessage](#)
A class for log messages.

- class [LogDestination](#)
A base class for log destinations.
- class [LogStream](#)
A class for logging to ostreams.
- class [LogFile](#)
A class for logging to files.
- class [Logger](#)
A logger class.
- class [MySQLDatabase](#)
Implements a MySQL version of the [Database](#) interface.
- class [MySQLQuery](#)
Implements a MySQL version of the [Query](#) database query class.
- class [OptionParser](#)
Command line option parser used by ARC command line tools.
- class [Profile](#)
Class used to convert human-friendly ini-style configuration to XML.
- class [Run](#)
This class runs an external executable.
- class [ThreadDataItem](#)
Base class for per-thread object.
- class [SimpleCondition](#)
Simple triggered condition.
- class [SimpleCounter](#)
Thread-safe counter with capability to wait for zero value.
- class [TimedMutex](#)
Mutex which allows a timeout on locking.
- class [SharedMutex](#)
Mutex which allows shared and exclusive locking.
- class [ThreadedPointer](#)
Wrapper for pointer with automatic destruction and multiple references.
- class [ThreadRegistry](#)
A set of conditions, mutexes, etc. conveniently exposed to monitor running child threads and to wait till they exit.
- class [ThreadInitializer](#)

This class initializes the glibmm thread system.

- class [URL](#)
Class to represent general URLs.
- class [URLLocation](#)
Class to hold a resolved [URL](#) location.
- class [PathIterator](#)
Class to iterate through elements of a path.
- class [User](#)
Platform independent representation of system user.
- class [UserSwitch](#)
Class for temporary switching of user id.
- class [ConfigEndpoint](#)
Represents the endpoint of service with a given type and [GLUE2](#) InterfaceName.
- class [initializeCredentialsType](#)
Defines how user credentials are looked for.
- class [UserConfig](#)
User configuration class
- class [CertEnvLocker](#)
Class for handling X509 variables in a multi-threaded environment.*
- class [EnvLockWrapper](#)
Class to provide automatic locking/unlocking of environment on creation/destruction.
- class [InterruptGuard](#)
Marks off a section of code which should not be interrupted by signals.
- class [AutoPointer](#)
Wrapper for pointer with automatic destruction.
- class [CountedPointer](#)
Wrapper for pointer with automatic destruction and multiple references.
- class [WatchdogListener](#)
This class is meant to provide interface for Watchdog executor part.
- class [WatchdogChannel](#)
This class is meant to be used in code which provides "I'm alive" ticks to watchdog.
- class [NS](#)
Class to represent an XML namespace.

- class [XMLNode](#)
Wrapper for LibXML library Tree interface.
- class [XMLNodeContainer](#)
Container for multiple [XMLNode](#) elements.
- class [ClientInterface](#)
Utility base class for [MCC](#).
- class [TCPSec](#)
- class [ClientTCP](#)
Class for setting up a [MCC](#) chain for TCP communication.
- struct [HTTPClientInfo](#)
- class [ClientHTTPAttributes](#)
Proxy class for handling request parameters.
- class [ClientHTTP](#)
Class for setting up a [MCC](#) chain for HTTP communication.
- class [ClientSOAP](#)
- class [SecHandlerConfig](#)
- class [DNListHandlerConfig](#)
- class [ARCPolicyHandlerConfig](#)
- class [ClientHTTPwithSAML2SSO](#)
- class [ClientSOAPwithSAML2SSO](#)
- class [ClientX509Delegation](#)
- class [Broker](#)
- class [ExecutionTargetSorter](#)
- class [BrokerPluginArgument](#)
- class [BrokerPlugin](#)
- class [BrokerPluginLoader](#)
- class [ComputingServiceUniq](#)
- class [ComputingServiceRetriever](#)
Retrieves information about computing elements by querying service registries and CE information systems.
- class [ConfusaCertHandler](#)
- class [ConfusaParserUtils](#)
- class [HakaClient](#)
- class [OpenIdpClient](#)
- class [OAuthConsumer](#)
- class [SAML2LoginClient](#)
- class [SAML2SSOHTTPClient](#)
- class [EndpointStatusMap](#)
- class [Endpoint](#)
Represents an endpoint of a service with a given interface type and capabilities.
- class [EndpointQueryingStatus](#)
Represents the status in the [EntityRetriever](#) of the query process of an [Endpoint](#) (service registry, computing element).

- class [EntityConsumer](#)

A general concept of an object which can consume entities use by the retrievers to return results.

- class [EntityContainer](#)

An entity consumer class storing all the consumed entities in a list.

- class [EntityRetriever](#)

Queries [Endpoint](#) objects (using plugins in parallel) and sends the found entities to consumers.

- class [EndpointQueryOptions](#)

Options controlling the query process.

- class [EndpointQueryOptions< Endpoint >](#)

The [EntityRetriever<Endpoint>](#) (a.k.a. [ServiceEndpointRetriever](#)) needs different options.

- class [EntityRetrieverPlugin](#)
- class [EntityRetrieverPluginLoader](#)
- class [ServiceEndpointRetrieverPlugin](#)
- class [TargetInformationRetrieverPlugin](#)
- class [JobListRetrieverPlugin](#)
- class [ApplicationEnvironment](#)

[ApplicationEnvironment](#).

- class [LocationAttributes](#)
- class [AdminDomainAttributes](#)
- class [ExecutionEnvironmentAttributes](#)
- class [ComputingManagerAttributes](#)
- class [ComputingShareAttributes](#)
- class [ComputingEndpointAttributes](#)
- class [ComputingServiceAttributes](#)
- class [LocationType](#)
- class [AdminDomainType](#)
- class [ExecutionEnvironmentType](#)
- class [ComputingManagerType](#)
- class [ComputingShareType](#)
- class [ComputingEndpointType](#)
- class [ComputingServiceType](#)
- class [ExecutionTarget](#)

[ExecutionTarget](#).

- class [GLUE2](#)

[GLUE2](#) parser.

- class [GLUE2Entity](#)
- class [Job](#)

[Job](#).

- class [JobInformationStorage](#)

Abstract class for storing job information.

- class [JobInformationStorageXML](#)
- class [JobControllerPlugin](#)
- class [JobControllerPluginLoader](#)
- class [JobControllerPluginArgument](#)
- class [OptIn](#)
- class [Range](#)
- class [ScalableTime](#)
- class [ScalableTime< int >](#)
- class [JobIdentificationType](#)
Job identification.
- class [ExecutableType](#)
Executable.
- class [RemoteLoggingType](#)
Remote logging.
- class [NotificationType](#)
- class [ApplicationType](#)
- class [SlotRequirementType](#)
- class [DiskSpaceRequirementType](#)
- class [ParallelEnvironmentType](#)
- class [ResourcesType](#)
- class [SourceType](#)
- class [TargetType](#)
- class [InputFileType](#)
- class [OutputFileType](#)
- class [DataStagingType](#)
- class [JobDescriptionResult](#)
- class [JobDescription](#)
- class [JobDescriptionParserPluginResult](#)
- class [JobDescriptionParserPlugin](#)
Abstract class for the different parsers.
- class [JobDescriptionParserPluginLoader](#)
- class [JobState](#)
- class [JobSupervisor](#)
JobSupervisor class.
- class [Software](#)
Used to represent software (names and version) and comparison.
- class [SoftwareRequirement](#)
Class used to express and resolve version requirements on software.
- class [SubmissionStatus](#)
- class [EndpointSubmissionStatus](#)
- class [Submitter](#)
- class [SubmitterPlugin](#)

Base class for the SubmitterPlugins.

- class [SubmitterPluginLoader](#)
- class [SubmitterPluginArgument](#)
- class [BrokerPluginTestACCCControl](#)
- class [JobDescriptionParserPluginTestACCCControl](#)
- class [JobControllerPluginTestACCCControl](#)
- class [SubmitterPluginTestACCCControl](#)
- class [JobStateTEST](#)
- class [JobListRetrieverPluginTESTControl](#)
- class [ServiceEndpointRetrieverPluginTESTControl](#)
- class [TargetInformationRetrieverPluginTESTControl](#)
- class [CredentialError](#)
- class [Credential](#)
- class [VOMSACInfo](#)
- class [VOMSTrustList](#)
- class [CredentialStore](#)
- class [XmlContainer](#)
- class [XmlDatabase](#)
- class [DelegationConsumer](#)
- class [DelegationProvider](#)
- class [DelegationConsumerSOAP](#)
- class [DelegationProviderSOAP](#)
- class [DelegationContainerSOAP](#)
- class [GlobusResult](#)
- class [GSSCredential](#)
- class [InfoCache](#)

Stores XML document in filesystem split into parts.

- class [InfoCacheInterface](#)
- class [InfoFilter](#)

Filters information document according to identity of requestor.

- class [InfoRegister](#)

Registration to Information Indexing [Service](#).

- class [InfoRegisters](#)

Handling registrations to multiple Information Indexing Services.

- struct [Register_Info_Type](#)
- struct [ISIS_description](#)
- class [InfoRegistrar](#)

Registration process associated with particular ISIS.

- class [InfoRegisterContainer](#)
- class [InformationInterface](#)

Information System message processor.

- class [InformationContainer](#)

Information System document container and processor.

- class [InformationRequest](#)
Request for information in InfoSystem.
- class [InformationResponse](#)
Informational response from InfoSystem.
- class [RegisteredService](#)
[RegisteredService](#) - extension of [Service](#) performing self-registration.
- class [FinderLoader](#)
- class [Loader](#)
Plugins loader.
- class [ModuleManager](#)
Manager of shared libraries.
- class [PluginArgument](#)
Base class for passing arguments to loadable ARC components.
- class [Plugin](#)
Base class for loadable ARC components.
- struct [PluginDescriptor](#)
Description of ARC loadable component.
- class [PluginDesc](#)
Description of plugin.
- class [ModuleDesc](#)
Description of loadable module.
- class [PluginsFactory](#)
Generic ARC plugins loader.
- class [MCCInterface](#)
Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects.
- class [MCC](#)
[Message](#) Chain Component - base class for every [MCC](#) plugin.
- class [MCCConfig](#)
- class [MCCPluginArgument](#)
- class [MCC_Status](#)
A class for communication of [MCC](#) processing results.
- class [MCCLoader](#)
Creator of [Message](#) Component Chains ([MCC](#)).
- class [ChainContext](#)

Interface to chain specific functionality.

- class [MessagePayload](#)
Base class for content of message passed through chain.
- class [MessageContextElement](#)
Top class for elements contained in message context.
- class [MessageContext](#)
Handler for content of message context.
- class [MessageAuthContext](#)
Handler for content of message auth context.*
- class [Message](#)
Object being passed through chain of MCCs.
- class [AttributeIterator](#)
A const iterator class for accessing multiple values of an attribute.
- class [MessageAttributes](#)
A class for storage of attribute values.
- class [MessageAuth](#)
Contains authenticity information, authorization tokens and decisions.
- class [PayloadRawInterface](#)
Random Access Payload for [Message](#) objects.
- struct [PayloadRawBuf](#)
- class [PayloadRaw](#)
Raw byte multi-buffer.
- class [PayloadSOAP](#)
Payload of [Message](#) with SOAP content.
- class [PayloadStreamInterface](#)
Stream-like Payload for [Message](#) object.
- class [PayloadStream](#)
POSIX handle as Payload.
- class [PlexerEntry](#)
A pair of label (regex) and pointer to [MCC](#).
- class [Plexer](#)
The [Plexer](#) class, used for routing messages to services.
- class [CStringValue](#)
This class implements case insensitive strings as security attributes.

- class [SecAttrValue](#)
This is an abstract interface to a security attribute.
- class [SecAttrFormat](#)
Export/import format.
- class [SecAttr](#)
This is an abstract interface to a security attribute.
- class [MultiSecAttr](#)
Container of multiple [SecAttr](#) attributes.
- class [Service](#)
[Service](#) - last component in a [Message](#) Chain.
- class [ServicePluginArgument](#)
- class [SOAPMessage](#)
[Message](#) restricted to SOAP payload.
- class [ClassLoader](#)
- class [ClassLoaderPluginArgument](#)
- class [WSAEndpointReference](#)
Interface for manipulation of WS-Addressing [Endpoint](#) Reference.
- class [WSAHeader](#)
Interface for manipulation WS-Addressing information in SOAP header.
- class [SAMLToken](#)
Class for manipulating SAML Token [Profile](#).
- class [UsernameToken](#)
Interface for manipulation of WS-Security according to Username Token [Profile](#).
- class [X509Token](#)
Class for manipulating X.509 Token [Profile](#).
- class [PayloadWSRF](#)
This class combines [MessagePayload](#) with [WSRF](#).
- class [WSRP](#)
Base class for WS-ResourceProperties structures.
- class [WSRPFault](#)
Base class for WS-ResourceProperties faults.
- class [WSRPInvalidResourcePropertyQNameFault](#)
- class [WSRPResourcePropertyChangeFailure](#)
- class [WSRPUnableToPutResourcePropertyDocumentFault](#)
- class [WSRPInvalidModificationFault](#)

- class [WSRPUnableToModifyResourcePropertyFault](#)
- class [WSRPSetResourcePropertyRequestFailedFault](#)
- class [WSRPInsertResourcePropertiesRequestFailedFault](#)
- class [WSRPUpdateResourcePropertiesRequestFailedFault](#)
- class [WSRPDeleteResourcePropertiesRequestFailedFault](#)
- class [WSRPGetResourcePropertyDocumentRequest](#)
- class [WSRPGetResourcePropertyDocumentResponse](#)
- class [WSRPGetResourcePropertyRequest](#)
- class [WSRPGetResourcePropertyResponse](#)
- class [WSRPGetMultipleResourcePropertiesRequest](#)
- class [WSRPGetMultipleResourcePropertiesResponse](#)
- class [WSRPPutResourcePropertyDocumentRequest](#)
- class [WSRPPutResourcePropertyDocumentResponse](#)
- class [WSRPModifyResourceProperties](#)
- class [WSRPInsertResourceProperties](#)
- class [WSRPUpdateResourceProperties](#)
- class [WSRPDeleteResourceProperties](#)
- class [WSRPSetResourcePropertiesRequest](#)
- class [WSRPSetResourcePropertiesResponse](#)
- class [WSRPInsertResourcePropertiesRequest](#)
- class [WSRPInsertResourcePropertiesResponse](#)
- class [WSRPUpdateResourcePropertiesRequest](#)
- class [WSRPUpdateResourcePropertiesResponse](#)
- class [WSRPDeleteResourcePropertiesRequest](#)
- class [WSRPDeleteResourcePropertiesResponse](#)
- class [WSRPQueryResourcePropertiesRequest](#)
- class [WSRPQueryResourcePropertiesResponse](#)
- class [WSRF](#)

Base class for every [WSRF](#) message.

- class [WSRFBaseFault](#)

Base class for [WSRF](#) fault messages.

- class [WSRFResourceUnknownFault](#)
- class [WSRFResourceUnavailableFault](#)
- class [XMLSecNode](#)

Extends [XMLNode](#) class to support XML security operation.

Typedefs

- typedef [EntityRetriever](#)< [Endpoint](#) > [ServiceEndpointRetriever](#)
- typedef [EntityRetriever](#)< [ComputingServiceType](#) > [TargetInformationRetriever](#)
- typedef [EntityRetriever](#)< [Job](#) > [JobListRetriever](#)
- typedef [Plugin](#) *(* [get_plugin_instance](#))([PluginArgument](#) *arg)
- typedef std::multimap< std::string, std::string > [AttrMap](#)
- typedef [AttrMap](#)::const_iterator [AttrConstIter](#)
- typedef [AttrMap](#)::iterator [AttrIter](#)

Enumerations

- enum `TimeFormat` {
`MDSTime`, `ASCTime`, `UserTime`, `ISOTime`,
`UTCTime`, `RFC1123Time`, `EpochTime` }
- enum `PeriodBase` {
`PeriodNanoseconds`, `PeriodMicroseconds`, `PeriodMiliseconds`, `PeriodSeconds`,
`PeriodMinutes`, `PeriodHours`, `PeriodDays`, `PeriodWeeks` }
- enum `LogLevel` {
`DEBUG` = 1, `VERBOSE` = 2, `INFO` = 4, `WARNING` = 8,
`ERROR` = 16, `FATAL` = 32 }
- enum `LogFormat` { `LongFormat`, `ShortFormat`, `DebugFormat`, `EmptyFormat` }
- enum `escape_type` { `escape_char`, `escape_octal`, `escape_hex` }
- enum `ServiceType` { `COMPUTING`, `INDEX` }
- enum `StatusKind` { ,
`STATUS_OK` = 1, `GENERIC_ERROR` = 2, `PARSING_ERROR` = 4, `PROTOCOL_-`
`RECOGNIZED_ERROR` = 8,
`UNKNOWN_SERVICE_ERROR` = 16, `BUSY_ERROR` = 32, `SESSION_CLOSE` = 64 }
- enum `WSAFault` { , `WSAFaultUnknown`, `WSAFaultInvalidAddressingHeader` }

Functions

- `std::ostream & operator<<` (`std::ostream &`, `const Period &`)
- `std::ostream & operator<<` (`std::ostream &`, `const Time &`)
- `std::string TimeStamp` (`const TimeFormat &`=`Time::GetFormat()`)
- `std::string TimeStamp` (`Time`, `const TimeFormat &`=`Time::GetFormat()`)
- `bool FileCopy` (`const std::string &source_path`, `const std::string &destination_path`, `uid_t uid`, `gid_t gid`)
- `bool FileCopy` (`const std::string &source_path`, `const std::string &destination_path`)
- `bool FileCopy` (`const std::string &source_path`, `int destination_handle`)
- `bool FileCopy` (`int source_handle`, `const std::string &destination_path`)
- `bool FileCopy` (`int source_handle`, `int destination_handle`)
- `bool FileRead` (`const std::string &filename`, `std::list< std::string > &data`, `uid_t uid=0`, `gid_t gid=0`)
- `bool FileRead` (`const std::string &filename`, `std::string &data`, `uid_t uid=0`, `gid_t gid=0`)
- `bool FileCreate` (`const std::string &filename`, `const std::string &data`, `uid_t uid=0`, `gid_t gid=0`, `mode_t mode=0`)
- `bool FileStat` (`const std::string &path`, `struct stat *st`, `bool follow_symlinks`)
- `bool FileStat` (`const std::string &path`, `struct stat *st`, `uid_t uid`, `gid_t gid`, `bool follow_symlinks`)
- `bool FileLink` (`const std::string &oldpath`, `const std::string &newpath`, `bool symbolic`)
- `bool FileLink` (`const std::string &oldpath`, `const std::string &newpath`, `uid_t uid`, `gid_t gid`, `bool symbolic`)
- `std::string FileReadLink` (`const std::string &path`)
- `std::string FileReadLink` (`const std::string &path`, `uid_t uid`, `gid_t gid`)
- `bool FileDelete` (`const std::string &path`)
- `bool FileDelete` (`const std::string &path`, `uid_t uid`, `gid_t gid`)
- `bool DirCreate` (`const std::string &path`, `mode_t mode`, `bool with_parents=false`)
- `bool DirCreate` (`const std::string &path`, `uid_t uid`, `gid_t gid`, `mode_t mode`, `bool with_parents=false`)
- `bool DirDelete` (`const std::string &path`, `bool recursive=true`)

- bool [DirDelete](#) (const std::string &path, bool recursive, uid_t uid, gid_t gid)
- bool [TmpDirCreate](#) (std::string &path)
- bool [TmpFileCreate](#) (std::string &filename, const std::string &data, uid_t uid=0, gid_t gid=0, mode_t mode=0)
- bool [CanonicalDir](#) (std::string &name, bool leading_slash=true)
- void [GUID](#) (std::string &guid)
- std::string [UUID](#) (void)
- const char * [FindTrans](#) (const char *p)
- const char * [FindNTrans](#) (const char *s, const char *p, unsigned long n)
- std::ostream & [operator<<](#) (std::ostream &os, const [IString](#) &msg)
- std::ostream & [operator<<](#) (std::ostream &os, const [LoggerFormat](#) &format)
- std::ostream & [operator<<](#) (std::ostream &os, [LogLevel](#) level)
- [LogLevel](#) [string_to_level](#) (const std::string &str)
- bool [istring_to_level](#) (const std::string &llStr, [LogLevel](#) &ll)
- bool [string_to_level](#) (const std::string &str, [LogLevel](#) &ll)
- std::string [level_to_string](#) (const [LogLevel](#) &level)
- [LogLevel](#) [old_level_to_level](#) (unsigned int old_level)
- template<typename T >
T [stringto](#) (const std::string &s)
- template<typename T >
bool [stringto](#) (const std::string &s, T &t)
- bool [strtoint](#) (const std::string &s, signed int &t, int base=10)
- bool [strtoint](#) (const std::string &s, unsigned int &t, int base=10)
- bool [strtoint](#) (const std::string &s, signed long &t, int base=10)
- bool [strtoint](#) (const std::string &s, unsigned long &t, int base=10)
- bool [strtoint](#) (const std::string &s, signed long long &t, int base=10)
- bool [strtoint](#) (const std::string &s, unsigned long long &t, int base=10)
- template<typename T >
std::string [tostring](#) (T t, int width=0, int precision=0)
- std::string [inttostr](#) (signed long long t, int base=10, int width=0)
- std::string [inttostr](#) (unsigned long long t, int base=10, int width=0)
- std::string [inttostr](#) (signed int t, int base=10, int width=0)
- std::string [inttostr](#) (unsigned int t, int base=10, int width=0)
- std::string [inttostr](#) (signed long t, int base=10, int width=0)
- std::string [inttostr](#) (unsigned long t, int base=10, int width=0)
- std::string [booltostr](#) (bool b)
- bool [strtobool](#) (const std::string &s)
- bool [strtobool](#) (const std::string &s, bool &b)
- std::string [lower](#) (const std::string &s)
- std::string [upper](#) (const std::string &s)
- void [tokenize](#) (const std::string &str, std::vector< std::string > &tokens, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes="")
- void [tokenize](#) (const std::string &str, std::list< std::string > &tokens, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes="")
- std::string::size_type [get_token](#) (std::string &token, const std::string &str, std::string::size_type pos, const std::string &delimiters=" ", const std::string &start_quotes="", const std::string &end_quotes="")
- std::string [trim](#) (const std::string &str, const char *sep=NULL)
- std::string [strip](#) (const std::string &str)
- std::string [uri_encode](#) (const std::string &str, bool encode_slash)
- std::string [uri_unencode](#) (const std::string &str)

- `std::string convert_to_rdn` (const `std::string` &dn)
- `std::string escape_chars` (const `std::string` &str, const `std::string` &chars, char esc, bool excl, `escape_type` type=escape_char)
- `std::string unescape_chars` (const `std::string` &str, char esc, `escape_type` type=escape_char)
- `bool CreateThreadFunction` (void(*func)(void *), void *arg, `SimpleCounter` *count=NULL)
- `std::list< URL > ReadURLList` (const `URL` &urllist)
- `std::string toString` (const `ServiceType` st)
- `std::string GetEnv` (const `std::string` &var)
- `std::string GetEnv` (const `std::string` &var, bool &found)
- `bool SetEnv` (const `std::string` &var, const `std::string` &value, bool overwrite=true)
- `void UnsetEnv` (const `std::string` &var)
- `void EnvLockAcquire` (void)
- `void EnvLockRelease` (void)
- `void EnvLockWrap` (bool all=false)
- `void EnvLockUnwrap` (bool all=false)
- `void EnvLockUnwrapComplete` (void)
- `std::string StrError` (int errnum=errno)
- `bool PersistentLibraryInit` (const `std::string` &name)
- `std::ostream & operator<<` (std::ostream &out, const `XMLNode` &node)
- `std::istream & operator>>` (std::istream &in, `XMLNode` &node)
- `bool MatchXMLName` (const `XMLNode` &node1, const `XMLNode` &node2)
- `bool MatchXMLName` (const `XMLNode` &node, const char *name)
- `bool MatchXMLName` (const `XMLNode` &node, const `std::string` &name)
- `bool MatchXMLNamespace` (const `XMLNode` &node1, const `XMLNode` &node2)
- `bool MatchXMLNamespace` (const `XMLNode` &node, const char *uri)
- `bool MatchXMLNamespace` (const `XMLNode` &node, const `std::string` &uri)
- `bool createVOMSAC` (`std::string` &codedac, `Credential` &issuer_cred, `Credential` &holder_cred, `std::vector< std::string >` &fqan, `std::vector< std::string >` &targets, `std::vector< std::string >` &attributes, `std::string` &voname, `std::string` &uri, int lifetime)
- `bool addVOMSAC` (`ArcCredential::AC` **&aclist, `std::string` &acorder, `std::string` &decodedac)
- `bool parseVOMSAC` (X509 *holder, const `std::string` &ca_cert_dir, const `std::string` &ca_cert_file, const `std::string` &vomkdir, `VOMSTrustList` &vomscert_trust_dn, `std::vector< VOMSACInfo >` &output, bool verify=true, bool reportall=false)
- `bool parseVOMSAC` (const `Credential` &holder_cred, const `std::string` &ca_cert_dir, const `std::string` &ca_cert_file, const `std::string` &vomkdir, `VOMSTrustList` &vomscert_trust_dn, `std::vector< VOMSACInfo >` &output, bool verify=true, bool reportall=false)
- `bool parseVOMSAC` (const `std::string` &cert_str, const `std::string` &ca_cert_dir, const `std::string` &ca_cert_file, const `std::string` &vomkdir, `VOMSTrustList` &vomscert_trust_dn, `std::vector< VOMSACInfo >` &output, bool verify=true, bool reportall=false)
- `char * VOMSDecode` (const char *data, int size, int *j)
- `char * VOMSEncode` (const char *data, int size, int *j)
- `std::string getCredentialProperty` (const `Arc::Credential` &u, const `std::string` &property, const `std::string` &ca_cert_dir=std::string(""), const `std::string` &ca_cert_file=std::string(""), const `std::string` &vomkdir=std::string(""), const `std::vector< std::string >` &vom_trust_list=std::vector< std::string >())
- `bool VOMSACSeqEncode` (const `std::string` &ac_seq, `std::string` &asn1)
- `bool VOMSACSeqEncode` (const `std::list< std::string >` acs, `std::string` &asn1)
- `bool OpenSSLInit` (void)
- `void HandleOpenSSLSError` (void)
- `void HandleOpenSSLSError` (int code)
- `int globus_error_to_errno` (const `std::string` &msg, int errno)

- `std::string` [string](#) ([StatusKind](#) kind)
- `const char *` [ContentFromPayload](#) (`const MessagePayload &payload`)
- `void` [WSAFaultAssign](#) (`SOAPEnvelope &message`, [WSAFault](#) fid)
- [WSAFault](#) [WSAFaultExtract](#) (`SOAPEnvelope &message`)
- `int` [passphrase_callback](#) (`char *buf`, `int size`, `int rwflag`, `void *`)
- `bool` [init_xmlsec](#) (`void`)
- `bool` [final_xmlsec](#) (`void`)
- `std::string` [get_cert_str](#) (`const char *certfile`)
- `xmlSecKey *` [get_key_from_keystr](#) (`const std::string &value`)
- `xmlSecKey *` [get_key_from_keyfile](#) (`const char *keyfile`)
- `std::string` [get_key_from_certfile](#) (`const char *certfile`)
- `xmlSecKey *` [get_key_from_certstr](#) (`const std::string &value`)
- `xmlSecKeysMngrPtr` [load_key_from_keyfile](#) (`xmlSecKeysMngrPtr *keys_manager`, `const char *keyfile`)
- `xmlSecKeysMngrPtr` [load_key_from_certfile](#) (`xmlSecKeysMngrPtr *keys_manager`, `const char *certfile`)
- `xmlSecKeysMngrPtr` [load_key_from_certstr](#) (`xmlSecKeysMngrPtr *keys_manager`, `const std::string &certstr`)
- `xmlSecKeysMngrPtr` [load_trusted_cert_file](#) (`xmlSecKeysMngrPtr *keys_manager`, `const char *cert_file`)
- `xmlSecKeysMngrPtr` [load_trusted_cert_str](#) (`xmlSecKeysMngrPtr *keys_manager`, `const std::string &cert_str`)
- `xmlSecKeysMngrPtr` [load_trusted_certs](#) (`xmlSecKeysMngrPtr *keys_manager`, `const char *cafile`, `const char *capath`)
- [XMLNode](#) [get_node](#) ([XMLNode](#) &parent, `const char *name`)

Variables

- `const` [ArcVersion](#) [Version](#)
- `const` `Glib::TimeVal` [ETERNAL](#)
- `const` `Glib::TimeVal` [HISTORIC](#)
- [Logger](#) [CredentialLogger](#)
- `const char *` [plugins_table_name](#)

9.1.1 Detailed Description

[Arc](#) namespace contains all core ARC classes.

9.1.2 Typedef Documentation

9.1.2.1 `typedef AttrMap::const_iterator` `Arc::AttrConstIter`

A typedef of a `const_iterator` for `AttrMap`. This typedef is used as a shorthand for a `const_iterator` for `AttrMap`. It is used extensively within the [MessageAttributes](#) class as well as the `AttributesIterator` class, but is not visible externally.

9.1.2.2 typedef AttrMap::iterator Arc::AttrIter

A typedef of an (non-const) iterator for AttrMap. This typedef is used as a shorthand for a (non-const) iterator for AttrMap. It is used in one method within the [MessageAttributes](#) class, but is not visible externally.

9.1.2.3 typedef std::multimap<std::string,std::string> Arc::AttrMap

A typedef of a multimap for storage of message attributes. This typedef is used as a shorthand for a multimap that uses strings for keys as well as values. It is used within the MessageAttributes class for internal storage of message attributes, but is not visible externally.

9.1.2.4 typedef Plugin*(* Arc::get_plugin_instance)(PluginArgument *arg)

Constructor function of ARC loadable component. This function is called with plugin-specific argument and should produce and return valid instance of plugin. If plugin can't be produced by any reason (for example because passed argument is not applicable) then NULL is returned. No exceptions should be raised.

9.1.3 Enumeration Type Documentation

9.1.3.1 enum Arc::escape_type

Type of escaping or encoding to use.

Enumerator:

- escape_char* place the escape character before the character being escaped
- escape_octal* octal encoding of the character
- escape_hex* hex encoding of the character

9.1.3.2 enum Arc::LogFormat

Output formats. Defines prefix for each message.

Enumerator:

- LongFormat* all information about message is printed
- ShortFormat* only message level is printed
- DebugFormat* message time (microsecond precision) and time difference from previous message are printed. This format is mostly meant for profiling.
- EmptyFormat* only message is printed

9.1.3.3 enum Arc::LogLevel

Logging levels for tagging and filtering log messages.

Enumerator:

- DEBUG* DEBUG level designates finer-grained informational events which should only be used for debugging purposes.

VERBOSE VERBOSE level designates fine-grained informational events that will give additional information about the application.

INFO INFO level designates informational messages that highlight the progress of the application at coarse-grained level.

WARNING WARNING level designates potentially harmful situations.

ERROR ERROR level designates error events that might still allow the application to continue running.

FATAL FATAL level designates very severe error events that will presumably lead the application to abort.

9.1.3.4 enum Arc::PeriodBase

Base to use when constructing a new [Period](#).

Enumerator:

PeriodNanoseconds Nanoseconds.

PeriodMicroseconds Microseconds.

PeriodMilliseconds Milliseconds.

PeriodSeconds Seconds.

PeriodMinutes Minutes.

PeriodHours Hours.

PeriodDays Days.

PeriodWeeks Weeks.

9.1.3.5 enum Arc::ServiceType

Type of service.

Enumerator:

COMPUTING A service that processes jobs.

INDEX A service that provides information.

9.1.3.6 enum Arc::StatusKind

Status kinds (types). This enum defines a set of possible status kinds.

Enumerator:

STATUS_OK Default status - undefined error.

GENERIC_ERROR No error.

PARSING_ERROR Error does not fit any class.

PROTOCOL_RECOGNIZED_ERROR Error detected while parsing request/response.

UNKNOWN_SERVICE_ERROR [Message](#) does not fit into expected protocol.

BUSY_ERROR There is no destination configured for this message.

SESSION_CLOSE [Message](#) can't be processed now.

9.1.3.7 enum Arc::TimeFormat

An enumeration that contains the possible textual time formats.

Enumerator:

MDSTime YYYYMMDDHHMMSSZ.
ASCTime Day Mon DD HH:MM:SS YYYY.
UserTime YYYY-MM-DD HH:MM:SS.
ISOTime YYYY-MM-DDTHH:MM:SS+HH:MM.
UTCTime YYYY-MM-DDTHH:MM:SSZ.
RFC1123Time Day, DD Mon YYYY HH:MM:SS GMT.
EpochTime 1234567890

9.1.3.8 enum Arc::WSAFault

WS-Addressing possible faults.

Enumerator:

WSAFaultUnknown This is not a fault
WSAFaultInvalidAddressingHeader This is not a WS-Addressing fault

9.1.4 Function Documentation

9.1.4.1 bool Arc::addVOMSAC (ArcCredential::AC **& *aclist*, std::string & *acorder*, std::string & *decodedac*)

Add decoded AC string into a list of AC objects

Parameters:

aclist The list of AC objects (output)
acorder The order of AC objects (output)
decodedac The AC string that is decoded from the string returned from voms server (input)

9.1.4.2 bool Arc::CanonicalDir (std::string & *name*, bool *leading_slash* = **true**)

Removes `./` from `'name'`. If `leading_slash=true` `'/'` will be added at the beginning of `'name'` if missing. Otherwise it will be removed. The directory separator used here depends on the platform.

Returns:

false if it is not possible to remove all the `./`

9.1.4.3 const char* Arc::ContentFromPayload (const MessagePayload & *payload*)

Returns pointer to main memory chunk of [Message](#) payload. If no buffer is present or if payload is not of [PayloadRawInterface](#) type NULL is returned.

9.1.4.4 **bool Arc::CreateThreadFunction (void(*) (void *) *func*, void * *arg*, SimpleCounter * *count* = NULL)**

Helper function to create simple thread. It takes care of all the peculiarities of the Glib::Thread API. It runs function 'func' with argument 'arg' in a separate thread. If count parameter is not NULL then count will be incremented before this function returns and then decremented when thread finishes.

Returns:

true on success.

9.1.4.5 **bool Arc::createVOMSAC (std::string & *codedac*, Credential & *issuer_cred*, Credential & *holder_cred*, std::vector< std::string > & *fqan*, std::vector< std::string > & *targets*, std::vector< std::string > & *attributes*, std::string & *voname*, std::string & *uri*, int *lifetime*)**

Create AC(Attribute Certificate) with voms specific format.

Parameters:

codedac The coded AC as output of this method

issuer_cred The issuer credential which is used to sign the AC

holder_cred The holder credential, the holder certificate is the one which carries AC The rest arguments are the same as the above method

9.1.4.6 **bool Arc::DirCreate (const std::string & *path*, uid_t *uid*, gid_t *gid*, mode_t *mode*, bool *with_parents* = false)**

Create a new directory using the specified uid and gid. Specified uid and gid are used for accessing filesystem.

9.1.4.7 **bool Arc::DirDelete (const std::string & *path*, bool *recursive*, uid_t *uid*, gid_t *gid*)**

Delete a directory, and its content if recursive is true. If the directory is not empty and recursive is false DirDelete will fail. Specified uid and gid are used for accessing filesystem.

9.1.4.8 **bool Arc::DirDelete (const std::string & *path*, bool *recursive* = true)**

Delete a directory, and its content if recursive is true. If the directory is not empty and recursive is false DirDelete will fail.

9.1.4.9 **void Arc::EnvLockAcquire (void)**

Obtain lock on environment. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment.

9.1.4.10 **void Arc::EnvLockRelease (void)**

Release lock on environment. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment.

9.1.4.11 void Arc::EnvLockUnwrap (bool *all* = false)

End code which is using setenv/getenv. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment.

Parameters:

all must be same as in corresponding EnvLockWrap.

Referenced by Arc::EnvLockWrapper::~~EnvLockWrapper().

9.1.4.12 void Arc::EnvLockUnwrapComplete (void)

Use after fork() to reset all internal variables and release all locks. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment.

9.1.4.13 void Arc::EnvLockWrap (bool *all* = false)

Start code which is using setenv/getenv. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment. Must always have corresponding EnvLockUnwrap.

Parameters:

all set to true for setenv and false for getenv.

Referenced by Arc::EnvLockWrapper::EnvLockWrapper().

9.1.4.14 std::string Arc::escape_chars (const std::string & *str*, const std::string & *chars*, char *esc*, bool *excl*, escape_type *type* = escape_char)

Escape or encode the given chars in str using the escape character esc. If excl is true then escape all characters not in chars.

9.1.4.15 bool Arc::FileCopy (const std::string & *source_path*, const std::string & *destination_path*, uid_t *uid*, gid_t *gid*)

Copy file *source_path* to file *destination_path*. Specified uid and gid are used for accessing filesystem.

9.1.4.16 bool Arc::FileCreate (const std::string & *filename*, const std::string & *data*, uid_t *uid* = 0, gid_t *gid* = 0, mode_t *mode* = 0)

Simple method to create a new file containing given data. Specified uid and gid are used for accessing filesystem. An existing file is overwritten with the new data. Permissions of the created file are determined using the current umask. If protected access is required, [FileLock](#) should be used in addition to FileRead. If uid/gid are zero then no real switch of uid/gid is done.

9.1.4.17 bool Arc::FileDelete (const std::string & *path*, uid_t *uid*, gid_t *gid*)

Deletes file at *path* using the specified uid and gid. Specified uid and gid are used for accessing filesystem.

9.1.4.18 **bool Arc::FileLink (const std::string & *oldpath*, const std::string & *newpath*, uid_t *uid*, gid_t *gid*, bool *symbolic*)**

Make symbolic or hard link of file using the specified uid and gid. Specified uid and gid are used for accessing filesystem.

9.1.4.19 **bool Arc::FileRead (const std::string & *filename*, std::string & *data*, uid_t *uid* = 0, gid_t *gid* = 0)**

Simple method to read whole file content from filename. Specified uid and gid are used for accessing filesystem.

9.1.4.20 **bool Arc::FileRead (const std::string & *filename*, std::list< std::string > & *data*, uid_t *uid* = 0, gid_t *gid* = 0)**

Simple method to read file content from filename. Specified uid and gid are used for accessing filesystem. The content is split into lines with the new line character removed, and the lines are returned in the data list. If protected access is required, [FileLock](#) should be used in addition to FileRead.

9.1.4.21 **std::string Arc::FileReadLink (const std::string & *path*, uid_t *uid*, gid_t *gid*)**

Returns path at which symbolic link is pointing using the specified uid and gid. Specified uid and gid are used for accessing filesystem.

9.1.4.22 **bool Arc::FileStat (const std::string & *path*, struct stat * *st*, uid_t *uid*, gid_t *gid*, bool *follow_symlinks*)**

Stat a file using the specified uid and gid and put info into the st struct. Specified uid and gid are used for accessing filesystem.

9.1.4.23 **bool Arc::final_xmlsec (void)**

Finalize the xml security library

9.1.4.24 **std::string Arc::get_cert_str (const char * *certfile*)**

Get certificate in string format from certificate file

9.1.4.25 **std::string Arc::get_key_from_certfile (const char * *certfile*)**

Get public key in string format from certificate file

9.1.4.26 **xmlSecKey* Arc::get_key_from_certstr (const std::string & *value*)**

Get public key in xmlSecKey structure from certificate string (the string under "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----")

9.1.4.27 xmlSecKey* Arc::get_key_from_keyfile (const char * *keyfile*)

Get key in xmlSecKey structure from key file

9.1.4.28 xmlSecKey* Arc::get_key_from_keysttr (const std::string & *value*)

Get key in xmlSecKey structure from key in string format

9.1.4.29 XMLNode Arc::get_node (XMLNode & *parent*, const char * *name*)

Generate a new child [XMLNode](#) with specified name

9.1.4.30 std::string Arc::getCredentialProperty (const Arc::Credential & *u*, const std::string & *property*, const std::string & *ca_cert_dir* = std::string(""), const std::string & *ca_cert_file* = std::string(""), const std::string & *vomsdir* = std::string(""), const std::vector< std::string > & *voms_trust_list* = std::vector< std::string >())

Extract the needed field from the certificate.

Parameters:

u The proxy certificate which includes the voms specific formatted AC.

property The property that caller would get, including: dn, voms:vo, voms:role, voms:group

ca_cert_dir

ca_cert_file

vomsdir

voms_trust_list the dn chain that is trusted when parsing voms AC

9.1.4.31 void Arc::GUID (std::string & *guid*)

Utilities for generating unique identifiers in the form 12345678-90ab-cdef-1234-567890abcdef. Generates a unique identifier using information such as IP address, current time etc.

9.1.4.32 bool Arc::init_xmlsec (void)

Initialize the xml security library, it should be called before the xml security functionality is used.

9.1.4.33 std::string Arc::inttostr (unsigned long *t*, int *base* = 10, int *width* = 0) [inline]

Convert unsigned long integer to textual representation for specied base. The result is left-padded with zeroes to make the string size width.

References inttostr().

9.1.4.34 `std::string Arc::inttostr (signed long t, int base = 10, int width = 0) [inline]`

Convert long integer to textual representation for specified base. The result is left-padded with zeroes to make the string size width.

References `inttostr()`.

9.1.4.35 `std::string Arc::inttostr (unsigned int t, int base = 10, int width = 0) [inline]`

Convert unsigned integer to textual representation for specified base. The result is left-padded with zeroes to make the string size width.

References `inttostr()`.

9.1.4.36 `std::string Arc::inttostr (signed int t, int base = 10, int width = 0) [inline]`

Convert integer to textual representation for specified base. The result is left-padded with zeroes to make the string size width.

References `inttostr()`.

9.1.4.37 `std::string Arc::inttostr (unsigned long long t, int base = 10, int width = 0)`

Convert unsigned long long integer to textual representation for specified base. The result is left-padded with zeroes to make the string size width.

9.1.4.38 `std::string Arc::inttostr (signed long long t, int base = 10, int width = 0)`

Convert long long integer to textual representation for specified base. The result is left-padded with zeroes to make the string size width.

Referenced by `inttostr()`.

9.1.4.39 `bool Arc::istring_to_level (const std::string & lStr, LogLevel & l)`

Case-insensitive parsing of a string to a LogLevel with error response. The method will try to parse (case-insensitive) the argument string to a corresponding LogLevel. If the method succeeds, `true` will be returned and the argument `l` will be set to the parsed LogLevel. If the parsing fails `false` will be returned. The parsing succeeds if `lStr` match (case-insensitively) one of the names of the LogLevel members.

Parameters:

lStr a string which should be parsed to a [Arc::LogLevel](#).

l a [Arc::LogLevel](#) reference which will be set to the matching [Arc::LogLevel](#) upon successful parsing.

Returns:

`true` in case of successful parsing, otherwise `false`.

See also:

[LogLevel](#)

9.1.4.40 `xmlSecKeysMngrPtr Arc::load_key_from_certfile (xmlSecKeysMngrPtr * keys_manager,
const char * certfile)`

Load public key from a certificate file into key manager

9.1.4.41 `xmlSecKeysMngrPtr Arc::load_key_from_certstr (xmlSecKeysMngrPtr * keys_manager,
const std::string & certstr)`

Load public key from a certificate string into key manager

9.1.4.42 `xmlSecKeysMngrPtr Arc::load_key_from_keyfile (xmlSecKeysMngrPtr * keys_manager,
const char * keyfile)`

Load private or public key from a key file into key manager

9.1.4.43 `xmlSecKeysMngrPtr Arc::load_trusted_cert_file (xmlSecKeysMngrPtr * keys_manager,
const char * cert_file)`

Load trusted certificate from certificate file into key manager

9.1.4.44 `xmlSecKeysMngrPtr Arc::load_trusted_cert_str (xmlSecKeysMngrPtr * keys_manager,
const std::string & cert_str)`

Load trusted certificate from certificate string into key manager

9.1.4.45 `xmlSecKeysMngrPtr Arc::load_trusted_certs (xmlSecKeysMngrPtr * keys_manager,
const char * cafile, const char * capath)`

Load trusted certificates from a file or directory into key manager

9.1.4.46 `bool Arc::OpenSSLInit (void)`

This module contains various convenience utilities for using OpenSSL. Application may be linked to this module instead of OpenSSL libraries directly. This function initializes OpenSSL library. It may be called multiple times and makes sure everything is done properly and OpenSSL may be used in multi-threaded environment. Because this function makes use of [ArcLocation](#) it is advisable to call it after [ArcLocation::Init\(\)](#).

9.1.4.47 `std::ostream& Arc::operator<< (std::ostream & os, LogLevel level)`

Printing of LogLevel values to ostreams. Output operator so that LogLevel values can be printed in a nicer way.

9.1.4.48 `bool Arc::parseVOMSAC (const std::string & cert_str, const std::string & ca_cert_dir, const std::string & ca_cert_file, const std::string & vommdir, VOMSTrustList & vomscert_trust_dn, std::vector< VOMSACInfo > & output, bool verify = true, bool reportall = false)`

Parse the certificate in string format.

Parameters:

cert_str one certificate, or a chain of certificate, in string format

9.1.4.49 `bool Arc::parseVOMSAC (const Credential & holder_cred, const std::string & ca_cert_dir, const std::string & ca_cert_file, const std::string & vommdir, VOMSTrustList & vomscert_trust_dn, std::vector< VOMSACInfo > & output, bool verify = true, bool reportall = false)`

Parse the certificate. Similar to above one, but collects information From all certificates in a chain.

9.1.4.50 `bool Arc::parseVOMSAC (X509 * holder, const std::string & ca_cert_dir, const std::string & ca_cert_file, const std::string & vommdir, VOMSTrustList & vomscert_trust_dn, std::vector< VOMSACInfo > & output, bool verify = true, bool reportall = false)`

Parse the certificate, and output the attributes.

Parameters:

holder The proxy certificate which includes the voms specific formatted AC.

ca_cert_dir The trusted certificates which are used to verify the certificate which is used to sign the AC

ca_cert_file The same as ca_cert_dir except it is a file instead of a directory. Only one of them need to be set

vommdir The directory which include *.lsc file for each vo. For instance, a vo called "knowarc.eu" should have file vommdir/knowarc/voms.knowarc.eu.lsc which contains on the first line the DN of the VOMS server, and on the second line the corresponding CA DN: /O=Grid/O=NorduGrid/OU=KnowARC/CN=voms.knowarc.eu /O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority See more in : <https://twiki.cern.ch/twiki/bin/view/LCG/VomsFAQforServiceManagers>

output The parsed attributes (Role and Generic Attribute) . Each attribute is stored in element of a vector as a string. It is up to the consumer to understand the meaning of the attribute. There are two types of attributes stored in VOMS AC: AC_IETFATTR, AC_FULL_ATTRIBUTES. The AC_IETFATTR will be like /Role=Employee/Group=Tester/Capability=NULL The AC_FULL_ATTRIBUTES will be like knowarc:Degree=PhD (qualifier::name=value) In order to make the output attribute values be identical, the voms server information is added as prefix of the original attributes in AC. for AC_FULL_ATTRIBUTES, the voname + hostname is added: /voname=knowarc.eu/hostname=arthur.hep.lu.se:15001//knowarc.eu/coredev:attribute1=1 for AC_IETFATTR, the 'VO' (voname) is added: /VO=knowarc.eu/Group=coredev/Role=NULL/Capability=NULL /VO=knowarc.eu/Group=testers/Role=NULL/Capability=NULL

some other redundant attributes is provided: voname=knowarc.eu/hostname=arthur.hep.lu.se:15001

Parameters:

verify true: Verify the voms certificate is trusted based on the `ca_cert_dir/ca_cert_file` which specifies the CA certificates, and the `vomscert_trust_dn` which specifies the trusted DN chain from voms server certificate to CA certificate. false: Not verify, which means the issuer of AC (voms server certificate is supposed to be trusted by default). In this case the parameters `'ca_cert_dir'`, `'ca_cert_file'` and `'vomscert_trust_dn'` will not effect, and may be left empty. This case is specifically used by `'arcproxy --info'` to list all of the attributes in AC, and not to need to verify if the AC's issuer is trusted.

reportall If set to true fills output with all attributes including those which failed passing test procedures. Validity of attributes can be checked through status members of output items. Combination of `verify=true` and `reportall=true` provides most information.

9.1.4.51 int Arc::passphrase_callback (char * buf, int size, int rwflag, void *)

callback method for inputing passphrase of key file

9.1.4.52 std::string Arc::string (StatusKind kind)

Conversion to string. Conversion from StatusKind to string.

Parameters:

kind The StatusKind to convert.

9.1.4.53 bool Arc::strtobool (const std::string & s, bool & b) [inline]

Convert string to bool. Checks whether string is equal to one of "true", "false", "1" or "0", and if not returns false. If equal, true is returned and the bool reference is set to true, if string equals "true" or "1", otherwise it is set to false.

9.1.4.54 bool Arc::strtoint (const std::string & s, unsigned long long & t, int base = 10)

Convert string to unsigned long long integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.55 bool Arc::strtoint (const std::string & s, signed long long & t, int base = 10)

Convert string to long long integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.56 bool Arc::strtoint (const std::string & *s*, unsigned long & *t*, int *base* = 10)

Convert string to unsigned long integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.57 bool Arc::strtoint (const std::string & *s*, signed long & *t*, int *base* = 10)

Convert string to long integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.58 bool Arc::strtoint (const std::string & *s*, unsigned int & *t*, int *base* = 10)

Convert string to unsigned integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.59 bool Arc::strtoint (const std::string & *s*, signed int & *t*, int *base* = 10)

Convert string to integer with specified base.

Returns:

false if any argument is wrong.

9.1.4.60 bool Arc::TmpDirCreate (std::string & *path*)

Create a temporary directory under the system defined temp location, and return its path. Uses mkdtemp if available, and a combination of random parameters if not. This latter method is not as safe as mkdtemp.

9.1.4.61 bool Arc::TmpFileCreate (std::string & *filename*, const std::string & *data*, uid_t *uid* = 0, gid_t *gid* = 0, mode_t *mode* = 0)

Simple method to create a temporary file containing given data. Specified uid and gid are used for accessing filesystem. Permissions of the created file are determined using the current umask. If uid/gid are zero then no real switch of uid/gid is done. Input value of filename argument is ignored. On output it contains path to created file. Content of data argument is written into created file.

9.1.4.62 std::string Arc::uri_encode (const std::string & *str*, bool *encode_slash*)

This method -encodes characters in URI *str*. Characters which are not unreserved according to RFC 3986 are encoded. If *encode_slash* is true forward slashes will also be encoded. It is useful to set *encode_slash* to false when encoding full paths.

9.1.4.63 bool Arc::VOMSACSeqEncode (const std::list< std::string > *acs*, std::string & *asn1*)

Encode the VOMS AC list into ASN1, so that the result can be used to insert into X509 as extension.

Parameters:

acs The input list includes a list of AC

asn1 The encoded value as output

9.1.4.64 bool Arc::VOMSACSeqEncode (const std::string & *ac_seq*, std::string & *asn1*)

Encode the VOMS AC list into ASN1, so that the result can be used to insert into X509 as extension.

Parameters:

ac_seq The input string includes a list of AC with VOMS_AC_HEADER and VOMS_AC_TRAILER as separator

asn1 The encoded value as output

9.1.4.65 char* Arc::VOMSDecode (const char * *data*, int *size*, int * *j*)

Decode the data which is encoded by voms server. Since voms code uses some specific coding method (not base64 encoding), we simply copy the method from voms code to here

9.1.4.66 char* Arc::VOMSEncode (const char * *data*, int *size*, int * *j*)

Encode the data with base64 encoding

9.1.4.67 void Arc::WSAFaultAssign (SOAPEnvelope & *message*, WSAFault *fid*)

Makes WS-Addressing fault. It fills SOAP Fault message with WS-Addressing fault related information.

9.1.4.68 WSAFault Arc::WSAFaultExtract (SOAPEnvelope & *message*)

Gets WS-addressing fault. Analyzes SOAP Fault message and returns WS-Addressing fault it represents.

9.1.5 Variable Documentation**9.1.5.1 Logger Arc::CredentialLogger**

[Logger](#) to be used by all modules of credentials library

9.1.5.2 const char* Arc::plugins_table_name

Name of symbol referring to table of plugins. This C null terminated string specifies name of symbol which shared library should export to give an access to an array of [PluginDescriptor](#) elements. The array is terminated by element with all components set to NULL.

9.1.5.3 `const ArcVersion Arc::Version`

Use this object to obtain current ARC HED version at runtime.

9.2 ArcCredential Namespace Reference

Data Structures

- struct [cert_verify_context](#)
- struct [PROXYPOLICY_st](#)
- struct [PROXYCERTINFO_st](#)
- struct [ACDIGEST](#)
- struct [ACIS](#)
- struct [ACFORM](#)
- struct [ACACI](#)
- struct [ACHOLDER](#)
- struct [ACVAL](#)
- struct [ACIETFATTR](#)
- struct [ACTARGET](#)
- struct [ACTARGETS](#)
- struct [ACATTR](#)
- struct [ACINFO](#)
- struct [ACC](#)
- struct [ACSEQ](#)
- struct [ACCERTS](#)
- struct [ACATTRIBUTE](#)
- struct [ACATTHOLDER](#)
- struct [ACFULLATTRIBUTES](#)

Enumerations

- enum [certType](#) {
[CERT_TYPE_EEC](#), [CERT_TYPE_CA](#), [CERT_TYPE_GSI_3_IMPERSONATION_PROXY](#),
[CERT_TYPE_GSI_3_INDEPENDENT_PROXY](#),
[CERT_TYPE_GSI_3_LIMITED_PROXY](#), [CERT_TYPE_GSI_3_RESTRICTED_PROXY](#),
[CERT_TYPE_GSI_2_PROXY](#), [CERT_TYPE_GSI_2_LIMITED_PROXY](#),
[CERT_TYPE_RFC_IMPERSONATION_PROXY](#), [CERT_TYPE_RFC_INDEPENDENT_PROXY](#),
[CERT_TYPE_RFC_LIMITED_PROXY](#), [CERT_TYPE_RFC_RESTRICTED_PROXY](#),
[CERT_TYPE_RFC_ANYLANGUAGE_PROXY](#) }

9.2.1 Detailed Description

Functions and constants for maintaining proxy certificates The code is derived from globus gsi, voms, and openssl-0.9.8e. The existing code for maintaining proxy certificates in OpenSSL only covers standard proxies and does not cover old Globus proxies, so here the Globus code is introduced.

9.2.2 Enumeration Type Documentation

9.2.2.1 enum ArcCredential::certType

Enumerator:

CERT_TYPE_EEC A end entity certificate

CERT_TYPE_CA A CA certificate

CERT_TYPE_GSI_3_IMPERSONATION_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant impersonation proxy

CERT_TYPE_GSI_3_INDEPENDENT_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant independent proxy

CERT_TYPE_GSI_3_LIMITED_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant limited proxy

CERT_TYPE_GSI_3_RESTRICTED_PROXY A X.509 Proxy Certificate Profile (pre-RFC) compliant restricted proxy

CERT_TYPE_GSI_2_PROXY A legacy Globus impersonation proxy

CERT_TYPE_GSI_2_LIMITED_PROXY A legacy Globus limited impersonation proxy

CERT_TYPE_RFC_IMPERSONATION_PROXY A X.509 Proxy Certificate Profile RFC compliant impersonation proxy; RFC inheritAll proxy

CERT_TYPE_RFC_INDEPENDENT_PROXY A X.509 Proxy Certificate Profile RFC compliant independent proxy; RFC independent proxy

CERT_TYPE_RFC_LIMITED_PROXY A X.509 Proxy Certificate Profile RFC compliant limited proxy

CERT_TYPE_RFC_RESTRICTED_PROXY A X.509 Proxy Certificate Profile RFC compliant restricted proxy

CERT_TYPE_RFC_ANYLANGUAGE_PROXY RFC anyLanguage proxy

9.3 DataStaging Namespace Reference

[DataStaging](#) contains all components for data transfer scheduling and execution.

Data Structures

- class [DataDelivery](#)
DataDelivery transfers data between specified physical locations.
- class [DataDeliveryComm](#)
This class provides an abstract interface for the Delivery layer.
- class [DataDeliveryCommHandler](#)
Singleton class handling all active DataDeliveryComm objects.
- class [DataDeliveryLocalComm](#)
This class starts, monitors and controls a local Delivery process.
- class [DataDeliveryRemoteComm](#)
This class contacts a remote service to make a Delivery request.
- class [TransferParameters](#)
Represents limits and properties of a DTR transfer. These generally apply to all DTRs.
- class [DTRCacheParameters](#)
The configured cache directories.
- class [DTRCallback](#)
The base class from which all callback-enabled classes should be derived.
- class [DTR](#)
Data Transfer Request.
- class [DTRLList](#)
Global list of all active DTRs in the system.
- class [DTRStatus](#)
Class representing the status of a DTR.
- class [DTRErrorStatus](#)
A class to represent error states reported by various components.
- class [Processor](#)
The Processor performs pre- and post-transfer operations.
- class [Scheduler](#)
The Scheduler is the control centre of the data staging framework.
- class [TransferSharesConf](#)

TransferSharesConf describes the configuration of *TransferShares*.

- class [TransferShares](#)

TransferShares is used to implement fair-sharing and priorities.

Typedefs

- typedef [Arc::ThreadedPointer](#)< [DTR](#) > [DTR_ptr](#)
- typedef [Arc::ThreadedPointer](#)< [Arc::Logger](#) > [DTRLogger](#)

Enumerations

- enum [StagingProcesses](#) {
 [GENERATOR](#), [SCHEDULER](#), [PRE_PROCESSOR](#), [DELIVERY](#),
 [POST_PROCESSOR](#) }
- enum [ProcessState](#) { [INITIATED](#), [RUNNING](#), [TO_STOP](#), [STOPPED](#) }
- enum [CacheState](#) {
 [CACHEABLE](#), [NON_CACHEABLE](#), [CACHE_ALREADY_PRESENT](#), [CACHE_-](#)
 [DOWNLOADED](#),
 [CACHE_LOCKED](#), [CACHE_SKIP](#), [CACHE_NOT_USED](#) }

Functions

- [DTR_ptr](#) [createDTRPtr](#) (const std::string &source, const std::string &destination, const [Arc::UserConfig](#) &usercfg, const std::string &jobid, const uid_t &uid, [DTRLogger](#) log)
- [DTRLogger](#) [createDTRLogger](#) ([Arc::Logger](#) &parent, const std::string &subdomain)

9.3.1 Detailed Description

[DataStaging](#) contains all components for data transfer scheduling and execution.

Chapter 10

Data Structure Documentation

10.1 ArcCredential::ACACI Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.2 ArcCredential::ACATTHOLDER Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.3 ArcCredential::ACATTR Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.4 ArcCredential::ACATTRIBUTE Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.5 ArcCredential::ACC Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.6 ArcCredential::ACCERTS Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.7 ArcCredential::ACDIGEST Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.8 ArcCredential::ACFORM Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.9 ArcCredential::ACFULLATTRIBUTES Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.10 ArcCredential::ACHOLDER Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.11 ArcCredential::ACIETFATTR Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.12 ArcCredential::ACINFO Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.13 ArcCredential::ACIS Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.14 ArcCredential::ACSEQ Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.15 ArcCredential::ACTARGET Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.16 ArcCredential::ACTARGETS Struct Reference

The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.17 ArcCredential::ACVAL Struct Reference

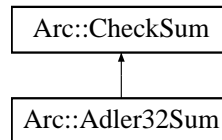
The documentation for this struct was generated from the following file:

- VOMSAttribute.h

10.18 Arc::Adler32Sum Class Reference

Implementation of Adler32 checksum.

#include <arc/Checksum.h> Inheritance diagram for Arc::Adler32Sum::



Public Member Functions

- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *)
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

10.18.1 Detailed Description

Implementation of Adler32 checksum. This class is a specialized class of the [Checksum](#) class. It provides an implementation of the Adler-32 checksum algorithm.

10.18.2 Member Function Documentation

10.18.2.1 virtual void Arc::Adler32Sum::add (void * *buf*, unsigned long long int *len*) [[inline](#), [virtual](#)]

Add data to be checksummed. This method calculates the checksum of the passed data chunk, taking into account the previous state of this object.

Parameters:

- buf* pointer to data chunk to be checksummed.
- len* size of the data chunk.

Implements [Arc::Checksum](#).

10.18.2.2 virtual void Arc::Adler32Sum::end (void) [[inline](#), [virtual](#)]

Finalize the checksumming. This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::Checksum](#).

10.18.2.3 virtual int Arc::Adler32Sum::print (char * *buf*, int *len*) const [inline, virtual]

Retrieve result of checksum into a string. The passed string *buf* is filled with result of checksum algorithm in base 16. At most *len* characters are filled into buffer *buf*. The hexadecimal value is prepended with "algorithm:", where algorithm is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters:

buf pointer to buffer which should be filled with checksum result.
len max number of character filled into buffer.

Returns:

0 on success

Reimplemented from [Arc::Checksum](#).

10.18.2.4 virtual void Arc::Adler32Sum::scan (const char * *buf*) [inline, virtual]

Set internal checksum state. This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [Checksum::print](#) method.

Parameters:

buf string containing textual representation of checksum

See also:

[Checksum::print](#)

Implements [Arc::Checksum](#).

10.18.2.5 virtual void Arc::Adler32Sum::start (void) [inline, virtual]

Initiate the checksum algorithm. This method must be called before starting a new checksum calculation.

Implements [Arc::Checksum](#).

The documentation for this class was generated from the following file:

- CheckSum.h

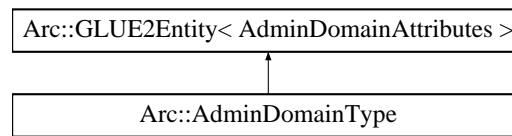
10.19 Arc::AdminDomainAttributes Class Reference

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.20 Arc::AdminDomainType Class Reference

Inheritance diagram for Arc::AdminDomainType::



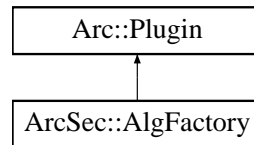
The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.21 ArcSec::AlgFactory Class Reference

Interface for algorithm factory class.

#include <AlgFactory.h>Inheritance diagram for ArcSec::AlgFactory::



Public Member Functions

- virtual [CombiningAlg](#) * [createAlg](#) (const std::string &type)=0

10.21.1 Detailed Description

Interface for algorithm factory class. [AlgFactory](#) is in charge of creating [CombiningAlg](#) according to the algorithm type given as argument of method [createAlg](#). This class can be inherited for implementing a factory class which can create some specific combining algorithm objects.

10.21.2 Member Function Documentation

10.21.2.1 virtual [CombiningAlg](#)* ArcSec::AlgFactory::createAlg (const std::string & type) [pure virtual]

creat algorithm object based on the type algorithm type

Parameters:

type The type of combining algorithm

Returns:

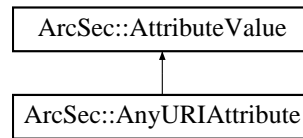
The object of [CombiningAlg](#)

The documentation for this class was generated from the following file:

- AlgFactory.h

10.22 ArcSec::AnyURIAttribute Class Reference

Inheritance diagram for ArcSec::AnyURIAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- std::string [getId](#) ()
- virtual std::string [getType](#) ()

10.22.1 Member Function Documentation

10.22.1.1 virtual std::string ArcSec::AnyURIAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.22.1.2 std::string ArcSec::AnyURIAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.22.1.3 virtual std::string ArcSec::AnyURIAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- AnyURIAttribute.h

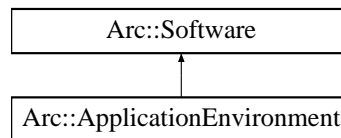
10.23 Arc::ApplicationEnvironment Class Reference

[ApplicationEnvironment](#).

```
#include <arc/compute/ExecutionTarget.h>Inheritance
Arc::ApplicationEnvironment::
```

diagram

for



10.23.1 Detailed Description

[ApplicationEnvironment](#). The ApplicationEnvironment is closely related to the definition given in [GLUE2](#). By extending the [Software](#) class the two [GLUE2](#) attributes AppName and AppVersion are mapped to two private members. However these can be obtained through the inherited member methods getName and getVersion.

[GLUE2](#) description: A description of installed application software or software environment characteristics available within one or more Execution Environments.

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.24 Arc::ApplicationType Class Reference

Data Fields

- [ExecutableType Executable](#)
- std::string [Input](#)
- std::string [Output](#)
- std::string [Error](#)
- std::list< [ExecutableType](#) > [PreExecutable](#)
- std::list< [ExecutableType](#) > [PostExecutable](#)
- std::string [LogDir](#)
- std::list< [RemoteLoggingType](#) > [RemoteLogging](#)

10.24.1 Field Documentation

10.24.1.1 std::string Arc::ApplicationType::Error

Standard error. The Error string specifies the relative path to the job session directory of the file which standard error of the job should be written to.

10.24.1.2 ExecutableType Arc::ApplicationType::Executable

Main executable to be run. The Executable object specifies the main executable which should be run by the job created by the job description enclosing this object. Note that in some job description languages specifying a main executable is not essential.

10.24.1.3 std::string Arc::ApplicationType::Input

Standard input. The Input string specifies the relative path to the job session directory of the file to be used for standard input for the job.

10.24.1.4 std::string Arc::ApplicationType::LogDir

Name of logging directory. The LogDir string specifies the name of the logging directory at the execution service which should be used to access log files for the job.

10.24.1.5 std::string Arc::ApplicationType::Output

Standard output. The Output string specifies the relative path to the job session directory of the file which standard output of the job should be written to.

10.24.1.6 std::list<ExecutableType> Arc::ApplicationType::PostExecutable

Executables to be run after the main executable. The PostExecutable object specifies a number of executables which should be executed after invoking the main application, where the main application is either the main executable (Executable) or the specified run time environment (RunTimeEnvironment in the [ResourcesType](#) class).

10.24.1.7 `std::list<ExecutableType>` `Arc::ApplicationType::PreExecutable`

Executables to be run before the main executable. The `PreExecutable` object specifies a number of executables which should be executed before invoking the main application, where the main application is either the main executable (`Executable`) or the specified run time environment (`RunTimeEnvironment` in the [ResourcesType](#) class).

10.24.1.8 `std::list<RemoteLoggingType>` `Arc::ApplicationType::RemoteLogging`

Remote logging services. The `RemoteLogging` list specifies the services to use for logging job information. See the [RemoteLoggingType](#) class for more details.

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.25 Arc::ArcLocation Class Reference

Determines ARC installation location.

```
#include <ArcLocation.h>
```

Static Public Member Functions

- static void [Init](#) (std::string path)
- static const std::string & [Get](#) ()
- static std::list< std::string > [GetPlugins](#) ()
- static std::string [GetDataDir](#) ()
- static std::string [GetToolsDir](#) ()

10.25.1 Detailed Description

Determines ARC installation location.

10.25.2 Member Function Documentation

10.25.2.1 static std::list<std::string> Arc::ArcLocation::GetPlugins () [static]

Returns ARC plugins directory location. Main source is value of variable ARC_PLUGIN_PATH, otherwise path is derived from installation location.

10.25.2.2 static void Arc::ArcLocation::Init (std::string *path*) [static]

Initializes location information. Main source is value of variable ARC_LOCATION, otherwise path to executable provided in path is used. If nothing works then warning message is sent to logger and initial installation prefix is used.

The documentation for this class was generated from the following file:

- ArcLocation.h

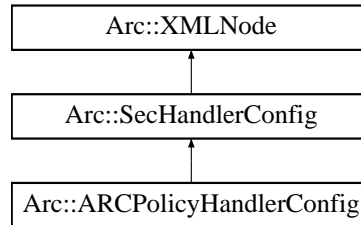
10.26 ArcSec::ArcPeriod Struct Reference

The documentation for this struct was generated from the following file:

- DateTimeAttribute.h

10.27 Arc::ARCPolicyHandlerConfig Class Reference

Inheritance diagram for Arc::ARCPolicyHandlerConfig::



The documentation for this class was generated from the following file:

- ClientInterface.h

10.28 Arc::ArcVersion Class Reference

Determines ARC HED libraries version at runtime.

```
#include <arc/ArcVersion.h>
```

Public Member Functions

- [ArcVersion](#) (const char *ver)

Data Fields

- const unsigned int [Major](#)
- const unsigned int [Minor](#)
- const unsigned int [Patch](#)

10.28.1 Detailed Description

Determines ARC HED libraries version at runtime.

The documentation for this class was generated from the following file:

- ArcVersion.h

10.29 ArcSec::Attr Struct Reference

[Attr](#) contains a tuple of attribute type and value.

```
#include <Request.h>
```

10.29.1 Detailed Description

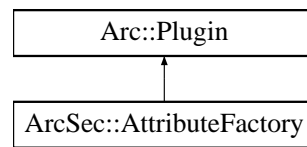
[Attr](#) contains a tuple of attribute type and value.

The documentation for this struct was generated from the following file:

- Request.h

10.30 ArcSec::AttributeFactory Class Reference

`#include <AttributeFactory.h>`Inheritance diagram for ArcSec::AttributeFactory::



10.30.1 Detailed Description

Base attribute factory class

The documentation for this class was generated from the following file:

- AttributeFactory.h

10.31 Arc::AttributeIterator Class Reference

A const iterator class for accessing multiple values of an attribute.

```
#include <MessageAttributes.h>
```

Public Member Functions

- [AttributeIterator](#) ()
- const std::string & [operator*](#) () const
- const std::string * [operator->](#) () const
- const std::string & [key](#) (void) const
- const [AttributeIterator](#) & [operator++](#) ()
- [AttributeIterator](#) [operator++](#) (int)
- bool [hasMore](#) () const

Protected Member Functions

- [AttributeIterator](#) ([AttrConstIter](#) begin, [AttrConstIter](#) end)

Protected Attributes

- [AttrConstIter](#) [current_](#)
- [AttrConstIter](#) [end_](#)

Friends

- class [MessageAttributes](#)

10.31.1 Detailed Description

A const iterator class for accessing multiple values of an attribute. This is an iterator class that is used when accessing multiple values of an attribute. The `getAll()` method of the [MessageAttributes](#) class returns an [AttributeIterator](#) object that can be used to access the values of the attribute.

Typical usage is:

```
MessageAttributes attributes;  
...  
for (AttributeIterator iterator=attributes.getAll("Foo:Bar");  
     iterator.hasMore(); ++iterator)  
    std::cout << *iterator << std::endl;
```

10.31.2 Constructor & Destructor Documentation

10.31.2.1 Arc::AttributeIterator::AttributeIterator ()

Default constructor. The default constructor. Does nothing since all attributes are instances of well-behaving STL classes.

10.31.2.2 **Arc::AttributeIterator::AttributeIterator (AttrConstIter *begin*, AttrConstIter *end*) [protected]**

Protected constructor used by the [MessageAttributes](#) class. This constructor is used to create an iterator for iteration over all values of an attribute. It is not supposed to be visible externally, but is only used from within the `getAll()` method of [MessageAttributes](#) class.

Parameters:

begin A `const_iterator` pointing to the first matching key-value pair in the internal multimap of the [MessageAttributes](#) class.

end A `const_iterator` pointing to the first key-value pair in the internal multimap of the [MessageAttributes](#) class where the key is larger than the key searched for.

10.31.3 Member Function Documentation

10.31.3.1 **bool Arc::AttributeIterator::hasMore () const**

Predicate method for iteration termination. This method determines whether there are more values for the iterator to refer to.

Returns:

Returns true if there are more values, otherwise false.

10.31.3.2 **const std::string& Arc::AttributeIterator::key (void) const**

The key of attribute. This method returns reference to key of attribute to which iterator refers.

10.31.3.3 **const std::string& Arc::AttributeIterator::operator* () const**

The dereference operator. This operator is used to access the current value referred to by the iterator.

Returns:

A (constant reference to a) string representation of the current value.

10.31.3.4 **AttributeIterator Arc::AttributeIterator::operator++ (int)**

The postfix advance operator. Advances the iterator to the next value. Works intuitively.

Returns:

An iterator referring to the value referred to by this iterator before the advance.

10.31.3.5 **const AttributeIterator& Arc::AttributeIterator::operator++ ()**

The prefix advance operator. Advances the iterator to the next value. Works intuitively.

Returns:

A const reference to this iterator.

10.31.3.6 `const std::string* Arc::AttributeIterator::operator-> () const`

The arrow operator. Used to call methods for value objects (strings) conveniently.

10.31.4 Friends And Related Function Documentation

10.31.4.1 `friend class MessageAttributes [friend]`

The [MessageAttributes](#) class is a friend. The constructor that creates an [AttributeIterator](#) that is connected to the internal multimap of the [MessageAttributes](#) class should not be exposed to the outside, but it still needs to be accessible from the `getAll()` method of the [MessageAttributes](#) class. Therefore, that class is a friend.

10.31.5 Field Documentation

10.31.5.1 `AttrConstIter Arc::AttributeIterator::current_ [protected]`

A `const_iterator` pointing to the current key-value pair. This iterator is the internal representation of the current value. It points to the corresponding key-value pair in the internal multimap of the [MessageAttributes](#) class.

10.31.5.2 `AttrConstIter Arc::AttributeIterator::end_ [protected]`

A `const_iterator` pointing beyond the last key-value pair. A `const_iterator` pointing to the first key-value pair in the internal multimap of the [MessageAttributes](#) class where the key is larger than the key searched for.

The documentation for this class was generated from the following file:

- `MessageAttributes.h`

10.32 ArcSec::AttributeProxy Class Reference

Interface for creating the [AttributeValue](#) object, it will be used by [AttributeFactory](#).

```
#include <AttributeProxy.h>
```

Public Member Functions

- virtual [AttributeValue](#) * [getAttribute](#) (const [Arc::XMLNode](#) &node)=0

10.32.1 Detailed Description

Interface for creating the [AttributeValue](#) object, it will be used by [AttributeFactory](#). The [AttributeProxy](#) object will be insert into [AttributeFactory](#); and the [getAttribute\(node\)](#) method will be called inside [AttributeFactory.createvalue\(node\)](#), in order to create a specific [AttributeValue](#)

10.32.2 Member Function Documentation

10.32.2.1 virtual [AttributeValue](#)* [ArcSec::AttributeProxy::getAttribute](#) (const [Arc::XMLNode](#) &*node*) [**pure virtual**]

Create a [AttributeValue](#) object according to the information inside the [XMLNode](#) as parameter.

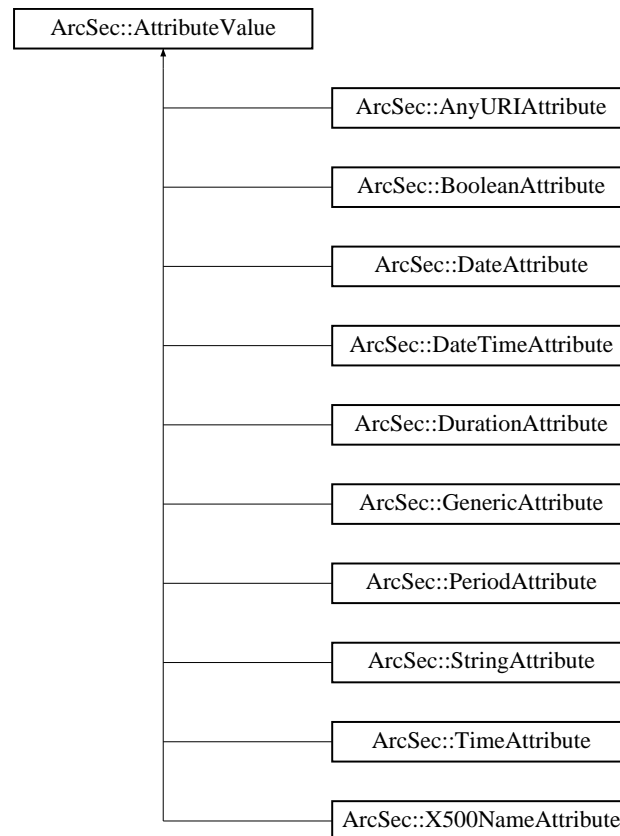
The documentation for this class was generated from the following file:

- [AttributeProxy.h](#)

10.33 ArcSec::AttributeValue Class Reference

Interface for containing different type of <Attribute> node for both policy and request.

#include <AttributeValue.h> Inheritance diagram for ArcSec::AttributeValue::



Public Member Functions

- virtual bool [equal](#) ([AttributeValue](#) *value, bool check_id=true)=0
- virtual std::string [encode](#) ()=0
- virtual std::string [getType](#) ()=0
- virtual std::string [getId](#) ()=0

10.33.1 Detailed Description

Interface for containing different type of <Attribute> node for both policy and request. <Attribute> contains different "Type" definition; Each type of <Attribute> needs different approach to compare the value. Any specific class which is for processing specific "Type" should inherit this class. The "Type" supported so far is: [StringAttribute](#), [DateAttribute](#), [TimeAttribute](#), [DurationAttribute](#), [PeriodAttribute](#), [AnyURIAttribute](#), [X500NameAttribute](#)

10.33.2 Member Function Documentation

10.33.2.1 `virtual std::string ArcSec::AttributeValue::encode () [pure virtual]`

encode the value in a string format

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

10.33.2.2 `virtual bool ArcSec::AttributeValue::equal (AttributeValue * value, bool check_id = true) [pure virtual]`

Evaluate whether "this" equals to the parameter value

10.33.2.3 `virtual std::string ArcSec::AttributeValue::getId () [pure virtual]`

Get the AttributeId of the <Attribute>

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

10.33.2.4 `virtual std::string ArcSec::AttributeValue::getType () [pure virtual]`

Get the DataType of the <Attribute>

Implemented in [ArcSec::AnyURIAttribute](#), [ArcSec::BooleanAttribute](#), [ArcSec::DateTimeAttribute](#), [ArcSec::TimeAttribute](#), [ArcSec::DateAttribute](#), [ArcSec::DurationAttribute](#), [ArcSec::PeriodAttribute](#), [ArcSec::GenericAttribute](#), [ArcSec::StringAttribute](#), and [ArcSec::X500NameAttribute](#).

The documentation for this class was generated from the following file:

- AttributeValue.h

10.34 ArcSec::Attrs Class Reference

[Attrs](#) is a container for one or more [Attr](#).

```
#include <Request.h>
```

10.34.1 Detailed Description

[Attrs](#) is a container for one or more [Attr](#). [Attrs](#) includes includes methods for inserting, getting items, and counting size as well

The documentation for this class was generated from the following file:

- Request.h

10.35 ArcSec::AuthzRequest Struct Reference

The documentation for this struct was generated from the following file:

- PDF.h

10.36 ArcSec::AuthzRequestSection Struct Reference

```
#include <PDP.h>
```

10.36.1 Detailed Description

These structure are based on the request schema for [PDP](#), so far it can apply to the ArcPDP's request schema, see `src/hed/pdc/Request.xsd` and `src/hed/pdc/Request.xml`. It could also apply to the XACMLPDP's request schema, since the difference is minor.

Another approach is, the service composes/marshalls the xml structure directly, then the service should use difference code to compose for ArcPDP's request schema and XACMLPDP's schema, which is not so good.

The documentation for this struct was generated from the following file:

- PDP.h

10.37 Arc::AutoPointer< T > Class Template Reference

Wrapper for pointer with automatic destruction.

```
#include <arc/Utils.h>
```

Public Member Functions

- [AutoPointer](#) (void)
- [AutoPointer](#) (T *o)
- [~AutoPointer](#) (void)
- T & [operator*](#) (void) const
- T * [operator->](#) (void) const
- [operator bool](#) (void) const
- bool [operator!](#) (void) const
- T * [Ptr](#) (void) const
- T * [Release](#) (void)

10.37.1 Detailed Description

```
template<typename T> class Arc::AutoPointer< T >
```

Wrapper for pointer with automatic destruction. If ordinary pointer is wrapped in instance of this class it will be automatically destroyed when instance is destroyed. This is useful for maintaining pointers in scope of one function. Only pointers returned by new() are supported.

The documentation for this class was generated from the following file:

- [Utils.h](#)

10.38 Arc::Base64 Class Reference

[Base64](#) encoding and decoding, borrowed from Axis2c project.

```
#include <arc/Base64.h>
```

Public Member Functions

- [Base64](#) ()

Static Public Member Functions

- static int [encode](#) (char *encoded, const char *string, int len)
- static int [decode](#) (char *bufplain, const char *bufcoded)

10.38.1 Detailed Description

[Base64](#) encoding and decoding, borrowed from Axis2c project.

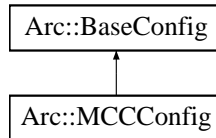
The documentation for this class was generated from the following file:

- Base64.h

10.39 Arc::BaseConfig Class Reference

Configuration for client interface.

#include <arc/ArcConfig.h> Inheritance diagram for Arc::BaseConfig::



Public Member Functions

- [BaseConfig](#) ()
- void [AddPluginsPath](#) (const std::string &path)
- void [AddPrivateKey](#) (const std::string &path)
- void [AddCertificate](#) (const std::string &path)
- void [AddProxy](#) (const std::string &path)
- void [AddCAFile](#) (const std::string &path)
- void [AddCAdir](#) (const std::string &path)
- void [AddOverlay](#) (XMLNode cfg)
- void [GetOverlay](#) (std::string fname)
- virtual XMLNode [MakeConfig](#) (XMLNode cfg) const

Data Fields

- std::string [key](#)
- std::string [cert](#)
- std::string [proxy](#)
- std::string [cafile](#)
- std::string [cadir](#)
- XMLNode [overlay](#)

Protected Attributes

- std::list< std::string > [plugin_paths](#)

10.39.1 Detailed Description

Configuration for client interface. It contains information which can't be expressed in class constructor arguments. Most probably common things like software installation location, identity of user, etc.

10.39.2 Member Function Documentation

10.39.2.1 virtual XMLNode Arc::BaseConfig::MakeConfig (XMLNode *cfg*) const **[virtual]**

Adds plugin configuration into common configuration tree supplied in 'cfg' argument.

Returns:

reference to XML node representing configuration of [ModuleManager](#)

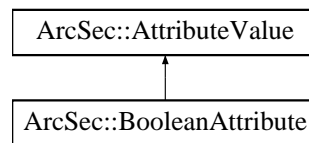
Reimplemented in [Arc::MCCConfig](#).

The documentation for this class was generated from the following file:

- ArcConfig.h

10.40 ArcSec::BooleanAttribute Class Reference

Inheritance diagram for ArcSec::BooleanAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- std::string [getId](#) ()
- std::string [getType](#) ()

10.40.1 Member Function Documentation

10.40.1.1 virtual std::string ArcSec::BooleanAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.40.1.2 std::string ArcSec::BooleanAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.40.1.3 std::string ArcSec::BooleanAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- BooleanAttribute.h

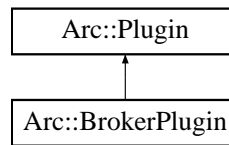
10.41 Arc::Broker Class Reference

The documentation for this class was generated from the following file:

- Broker.h

10.42 Arc::BrokerPlugin Class Reference

Inheritance diagram for Arc::BrokerPlugin::

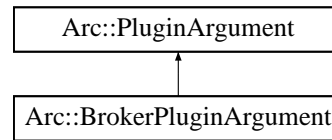


The documentation for this class was generated from the following file:

- [BrokerPlugin.h](#)

10.43 Arc::BrokerPluginArgument Class Reference

Inheritance diagram for Arc::BrokerPluginArgument::

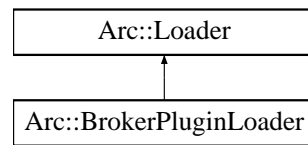


The documentation for this class was generated from the following file:

- [BrokerPlugin.h](#)

10.44 Arc::BrokerPluginLoader Class Reference

Inheritance diagram for Arc::BrokerPluginLoader::



The documentation for this class was generated from the following file:

- [BrokerPlugin.h](#)

10.45 Arc::BrokerPluginTestACControl Class Reference

The documentation for this class was generated from the following file:

- [TestACControl.h](#)

10.46 ArcCredential::cert_verify_context Struct Reference

The documentation for this struct was generated from the following file:

- CertUtil.h

10.47 Arc::CertEnvLocker Class Reference

Class for handling X509* variables in a multi-threaded environment.

```
#include <arc/UserConfig.h>
```

Public Member Functions

- [CertEnvLocker](#) (const [UserConfig](#) &cfg)
- [~CertEnvLocker](#) (void)

10.47.1 Detailed Description

Class for handling X509* variables in a multi-threaded environment. This class is useful when using external libraries which depend on X509* environment variables in a multi-threaded environment. When an instance of this class is created it holds a lock on these variables until the instance is destroyed. Additionally, if the credentials pointed to by the those variables are owned by a different uid from the uid of the current process, a temporary copy is made owned by the uid of the current process and the X509 variable points there instead. This is to comply with some restrictions in third-party libraries which insist on the credential files being owned by the current uid.

The documentation for this class was generated from the following file:

- UserConfig.h

10.48 AuthN::certInfo Struct Reference

The documentation for this struct was generated from the following file:

- NSSUtil.h

10.49 Arc::ChainContext Class Reference

Interface to chain specific functionality.

```
#include <MCCLoader.h>
```

Public Member Functions

- [operator PluginsFactory * \(\)](#)

10.49.1 Detailed Description

Interface to chain specific functionality. Object of this class is associated with every [MCCLoader](#) object. It is accessible for [MCC](#) and [Service](#) components and provides an interface to manipulate chains stored in [Loader](#). This makes it possible to modify chains dynamically - like deploying new services on demand.

10.49.2 Member Function Documentation

10.49.2.1 Arc::ChainContext::operator PluginsFactory * () `[inline]`

Returns associated [PluginsFactory](#) object

References [Arc::Loader::factory_](#).

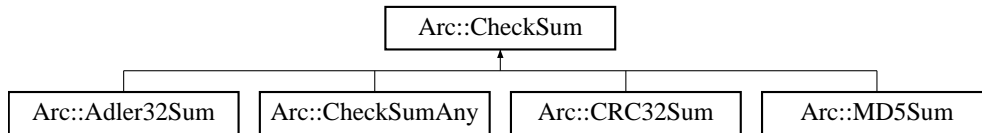
The documentation for this class was generated from the following file:

- [MCCLoader.h](#)

10.50 Arc::Checksum Class Reference

Interface for checksum manipulations.

#include <arc/Checksum.h> Inheritance diagram for Arc::Checksum::



Public Member Functions

- [Checksum](#) (void)
- virtual void [start](#) (void)=0
- virtual void [add](#) (void *buf, unsigned long long int len)=0
- virtual void [end](#) (void)=0
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const =0
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *buf)=0
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

10.50.1 Detailed Description

Interface for checksum manipulations. This class is an interface and is extended in the specialized classes [CRC32Sum](#), [MD5Sum](#) and [Adler32Sum](#). The interface is among others used during data transfers through [DataBuffer](#) class. The helper class [ChecksumAny](#) can be used as an easier way of handling automatically the different checksum types.

See also:

[ChecksumAny](#)
[CRC32Sum](#)
[MD5Sum](#)
[Adler32Sum](#)

10.50.2 Member Function Documentation

10.50.2.1 virtual void Arc::Checksum::add (void * buf, unsigned long long int len) [pure virtual]

Add data to be checksummed. This method calculates the checksum of the passed data chunk, taking into account the previous state of this object.

Parameters:

buf pointer to data chunk to be checksummed.
len size of the data chunk.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::ChecksumAny](#).

Referenced by [Arc::ChecksumAny::add\(\)](#).

10.50.2.2 virtual void Arc::Checksum::end (void) [pure virtual]

Finalize the checksumming. This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::ChecksumAny](#).

Referenced by [Arc::ChecksumAny::end\(\)](#).

10.50.2.3 virtual int Arc::Checksum::print (char * buf, int len) const [inline, virtual]

Retrieve result of checksum into a string. The passed string buf is filled with result of checksum algorithm in base 16. At most len characters are filled into buffer buf. The hexadecimal value is prepended with "algorithm:", where algorithm is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and [Adler32](#) classes.

Parameters:

buf pointer to buffer which should be filled with checksum result.

len max number of character filled into buffer.

Returns:

0 on success

Reimplemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::ChecksumAny](#).

Referenced by [Arc::ChecksumAny::print\(\)](#).

10.50.2.4 virtual void Arc::Checksum::scan (const char * buf) [pure virtual]

Set internal checksum state. This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [Checksum::print](#) method.

Parameters:

buf string containing textual representation of checksum

See also:

[Checksum::print](#)

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::ChecksumAny](#).

Referenced by [Arc::ChecksumAny::scan\(\)](#).

10.50.2.5 virtual void Arc::Checksum::start (void) [pure virtual]

Initiate the checksum algorithm. This method must be called before starting a new checksum calculation.

Implemented in [Arc::CRC32Sum](#), [Arc::MD5Sum](#), [Arc::Adler32Sum](#), and [Arc::ChecksumAny](#).

Referenced by `Arc::ChecksumAny::start()`.

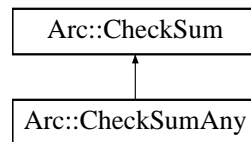
The documentation for this class was generated from the following file:

- `Checksum.h`

10.51 Arc::ChecksumAny Class Reference

Wrapper for [Checksum](#) class.

`#include <arc/Checksum.h>`Inheritance diagram for Arc::ChecksumAny::



Public Types

- enum [type](#) {
[none](#), [unknown](#), [undefined](#), [cksum](#),
[md5](#), [adler32](#) }

Public Member Functions

- [ChecksumAny](#) ([Checksum](#) *c=NULL)
- [ChecksumAny](#) ([type](#) type)
- [ChecksumAny](#) (const char *type)
- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *buf)
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

Static Public Member Functions

- static std::string [FileChecksum](#) (const std::string &filepath, [type](#) tp=md5, bool decimalbase=false)

10.51.1 Detailed Description

Wrapper for [Checksum](#) class. To be used for manipulation of any supported checksum type in a transparent way.

10.51.2 Member Enumeration Documentation

10.51.2.1 enum Arc::ChecksumAny::type

Type of checksum.

Enumerator:

none No checksum.
unknown Unknown checksum.
undefined Undefined checksum.
cksum CRC32 checksum.
md5 MD5 checksum.
adler32 ADLER32 checksum.

10.51.3 Member Function Documentation

10.51.3.1 virtual void Arc::ChecksumAny::add (void * *buf*, unsigned long long int *len*) [inline, virtual]

Add data to be checksummed. This method calculates the checksum of the passed data chunk, taking into account the previous state of this object.

Parameters:

buf pointer to data chunk to be checksummed.
len size of the data chunk.

Implements [Arc::Checksum](#).

References [Arc::Checksum::add\(\)](#).

10.51.3.2 virtual void Arc::ChecksumAny::end (void) [inline, virtual]

Finalize the checksumming. This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::Checksum](#).

References [Arc::Checksum::end\(\)](#).

10.51.3.3 static std::string Arc::ChecksumAny::FileChecksum (const std::string & *filepath*, type *tp* = md5, bool *decimalbase* = false) [static]

Get checksum of a file. This method provides an easy way to get the checksum of a file, by only specifying the path to the file. Optionally the checksum type can be specified, if not the MD5 algorithm will be used.

Parameters:

filepath path to file of which checksum should be calculated
tp type of checksum algorithm to use, default is md5.
decimalbase specifies whether output should be in base 10 or 16

Returns:

a string containing the calculated checksum is returned.

10.51.3.4 `virtual int Arc::ChecksumAny::print (char * buf, int len) const [inline, virtual]`

Retrieve result of checksum into a string. The passed string *buf* is filled with result of checksum algorithm in base 16. At most *len* characters are filled into buffer *buf*. The hexadecimal value is prepended with "algorithm:", where algorithm is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters:

buf pointer to buffer which should be filled with checksum result.

len max number of character filled into buffer.

Returns:

0 on success

Reimplemented from [Arc::Checksum](#).

References [Arc::Checksum::print\(\)](#).

10.51.3.5 `virtual void Arc::ChecksumAny::scan (const char * buf) [inline, virtual]`

Set internal checksum state. This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [Checksum::print](#) method.

Parameters:

buf string containing textual representation of checksum

See also:

[Checksum::print](#)

Implements [Arc::Checksum](#).

References [Arc::Checksum::scan\(\)](#).

10.51.3.6 `virtual void Arc::ChecksumAny::start (void) [inline, virtual]`

Initiate the checksum algorithm. This method must be called before starting a new checksum calculation.

Implements [Arc::Checksum](#).

References [Arc::Checksum::start\(\)](#).

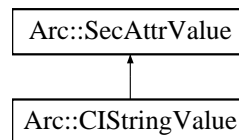
The documentation for this class was generated from the following file:

- [Checksum.h](#)

10.52 Arc::CStringValue Class Reference

This class implements case insensitive strings as security attributes.

`#include <CStringValue.h>`Inheritance diagram for Arc::CStringValue::



Public Member Functions

- [CStringValue](#) ()
- [CStringValue](#) (const char *ss)
- [CStringValue](#) (const std::string &ss)
- virtual [operator bool](#) ()

Protected Member Functions

- virtual bool [equal](#) ([SecAttrValue](#) &b)

10.52.1 Detailed Description

This class implements case insensitive strings as security attributes. This is an example of how to inherit [SecAttrValue](#). The class is meant to implement security attributes that are case insensitive strings.

10.52.2 Constructor & Destructor Documentation

10.52.2.1 Arc::CStringValue::CStringValue ()

Default constructor

10.52.2.2 Arc::CStringValue::CStringValue (const char * ss)

This is a constructor that takes a string littoral.

10.52.2.3 Arc::CStringValue::CStringValue (const std::string & ss)

This is a constructor that takes a string object.

10.52.3 Member Function Documentation

10.52.3.1 virtual bool Arc::CStringValue::equal (SecAttrValue & b) [protected, virtual]

This function returns true if two strings are the same apart from letter case

10.52.3.2 virtual Arc::CStringValue::operator bool () [virtual]

This function returns false if the string is empty or uninitialized

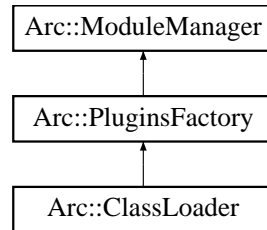
Reimplemented from [Arc::SecAttrValue](#).

The documentation for this class was generated from the following file:

- CStringValue.h

10.53 Arc::ClassLoader Class Reference

Inheritance diagram for Arc::ClassLoader::

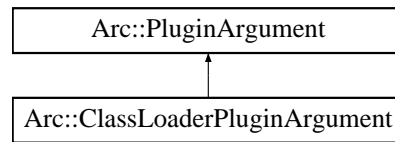


The documentation for this class was generated from the following file:

- ClassLoader.h

10.54 Arc::ClassLoaderPluginArgument Class Reference

Inheritance diagram for Arc::ClassLoaderPluginArgument::



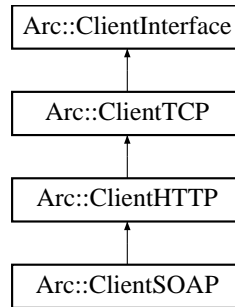
The documentation for this class was generated from the following file:

- ClassLoader.h

10.55 Arc::ClientHTTP Class Reference

Class for setting up a [MCC](#) chain for HTTP communication.

#include <ClientInterface.h> Inheritance diagram for Arc::ClientHTTP::



Public Member Functions

- [MCC](#) * [GetEntry](#) ()
- virtual [MCC_Status Load](#) ()

10.55.1 Detailed Description

Class for setting up a [MCC](#) chain for HTTP communication. The [ClientHTTP](#) class inherits from the [ClientTCP](#) class and adds an HTTP [MCC](#) to the chain.

10.55.2 Member Function Documentation

10.55.2.1 [MCC](#)* Arc::ClientHTTP::GetEntry () [inline]

Returns entry point to HTTP [MCC](#) in configured chain. To initialize entry point [Load\(\)](#) method must be called.

Reimplemented from [Arc::ClientTCP](#).

Reimplemented in [Arc::ClientSOAP](#).

10.55.2.2 virtual [MCC_Status](#) Arc::ClientHTTP::Load () [virtual]

Initializes communication chain for this object. Call to this method in derived class is not needed if [process\(\)](#) methods are used. It is only needed if [GetEntry\(\)](#) is used before [process\(\)](#) is called.

Reimplemented from [Arc::ClientTCP](#).

Reimplemented in [Arc::ClientSOAP](#).

The documentation for this class was generated from the following file:

- [ClientInterface.h](#)

10.56 Arc::ClientHTTPAttributes Class Reference

Proxy class for handling request parameters.

```
#include <ClientInterface.h>
```

10.56.1 Detailed Description

Proxy class for handling request parameters. The purpose of this class is to reduce number of methods in [ClientHTTP](#) class. Use only for temporary variables.

The documentation for this class was generated from the following file:

- ClientInterface.h

10.57 Arc::ClientHTTPwithSAML2SSO Class Reference

Public Member Functions

- [ClientHTTPwithSAML2SSO](#) ()
- [MCC_Status process](#) (const std::string &method, [PayloadRawInterface](#) *request, [HTTPClientInfo](#) *info, [PayloadRawInterface](#) **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false)

10.57.1 Constructor & Destructor Documentation

10.57.1.1 Arc::ClientHTTPwithSAML2SSO::ClientHTTPwithSAML2SSO () [inline]

Constructor creates [MCC](#) chain and connects to server.

10.57.2 Member Function Documentation

10.57.2.1 MCC_Status Arc::ClientHTTPwithSAML2SSO::process (const std::string & *method*, [PayloadRawInterface](#) * *request*, [HTTPClientInfo](#) * *info*, [PayloadRawInterface](#) ** *response*, const std::string & *idp_name*, const std::string & *username*, const std::string & *password*, const bool *reuse_authn* = *false*)

Send HTTP request and receive response.

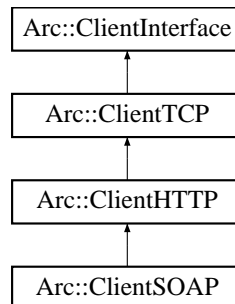
The documentation for this class was generated from the following file:

- [ClientSAML2SSO.h](#)

10.58 Arc::ClientInterface Class Reference

Utility base class for [MCC](#).

`#include <ClientInterface.h>`Inheritance diagram for Arc::ClientInterface::



Public Member Functions

- virtual [MCC_Status Load \(\)](#)

10.58.1 Detailed Description

Utility base class for [MCC](#). The [ClientInterface](#) class is a utility base class used for configuring a client side [Message Chain Component \(MCC\)](#) chain and loading it into memory. It has several specializations of increasing complexity of the [MCC](#) chains. This class is not supposed to be used directly. Instead its descendants like [ClientTCP](#), [ClientHTTP](#), etc. must be used.

10.58.2 Member Function Documentation

10.58.2.1 virtual MCC_Status Arc::ClientInterface::Load () [virtual]

Initializes communication chain for this object. Call to this method in derived class is not needed if `process()` methods are used. It is only needed if `GetEntry()` is used before `process()` is called.

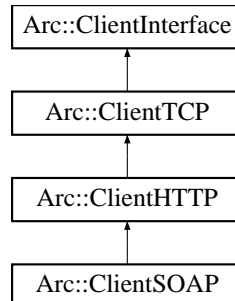
Reimplemented in [Arc::ClientTCP](#), [Arc::ClientHTTP](#), and [Arc::ClientSOAP](#).

The documentation for this class was generated from the following file:

- [ClientInterface.h](#)

10.59 Arc::ClientSOAP Class Reference

#include <ClientInterface.h> Inheritance diagram for Arc::ClientSOAP::



Public Member Functions

- [ClientSOAP \(\)](#)
- [MCC_Status process \(PayloadSOAP *request, PayloadSOAP **response\)](#)
- [MCC_Status process \(const std::string &action, PayloadSOAP *request, PayloadSOAP **response\)](#)
- [MCC * GetEntry \(\)](#)
- void [AddSecHandler \(XMLNode handlercfg, const std::string &libanme="", const std::string &libpath=""\)](#)
- virtual [MCC_Status Load \(\)](#)

10.59.1 Detailed Description

Class with easy interface for sending/receiving SOAP messages over HTTP(S/G). It takes care of configuring [MCC](#) chain and making an entry point.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 Arc::ClientSOAP::ClientSOAP () [inline]

Constructor creates [MCC](#) chain and connects to server.

10.59.3 Member Function Documentation

10.59.3.1 void Arc::ClientSOAP::AddSecHandler (XMLNode handlercfg, const std::string &libanme = "", const std::string &libpath = "")

Adds security handler to configuration of SOAP [MCC](#)

Reimplemented from [Arc::ClientHTTP](#).

10.59.3.2 MCC* Arc::ClientSOAP::GetEntry () [inline]

Returns entry point to SOAP [MCC](#) in configured chain. To initialize entry point [Load\(\)](#) method must be called.

Reimplemented from [Arc::ClientHTTP](#).

10.59.3.3 virtual MCC_Status Arc::ClientSOAP::Load () [virtual]

Instantiates pluggable elements according to generated configuration

Reimplemented from [Arc::ClientHTTP](#).

10.59.3.4 MCC_Status Arc::ClientSOAP::process (const std::string & *action*, PayloadSOAP * *request*, PayloadSOAP ** *response*)

Send SOAP request with specified SOAP action and receive response.

10.59.3.5 MCC_Status Arc::ClientSOAP::process (PayloadSOAP * *request*, PayloadSOAP ** *response*)

Send SOAP request and receive response.

The documentation for this class was generated from the following file:

- ClientInterface.h

10.60 Arc::ClientSOAPwithSAML2SSO Class Reference

Public Member Functions

- [ClientSOAPwithSAML2SSO](#) ()
- [MCC_Status process](#) ([PayloadSOAP](#) *request, [PayloadSOAP](#) **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false)
- [MCC_Status process](#) (const std::string &action, [PayloadSOAP](#) *request, [PayloadSOAP](#) **response, const std::string &idp_name, const std::string &username, const std::string &password, const bool reuse_authn=false)

10.60.1 Constructor & Destructor Documentation

10.60.1.1 Arc::ClientSOAPwithSAML2SSO::ClientSOAPwithSAML2SSO () [inline]

Constructor creates [MCC](#) chain and connects to server.

10.60.2 Member Function Documentation

10.60.2.1 MCC_Status Arc::ClientSOAPwithSAML2SSO::process (const std::string & action, PayloadSOAP * request, PayloadSOAP ** response, const std::string & idp_name, const std::string & username, const std::string & password, const bool reuse_authn = false)

Send SOAP request with specified SOAP action and receive response.

10.60.2.2 MCC_Status Arc::ClientSOAPwithSAML2SSO::process (PayloadSOAP * request, PayloadSOAP ** response, const std::string & idp_name, const std::string & username, const std::string & password, const bool reuse_authn = false)

Send SOAP request and receive response.

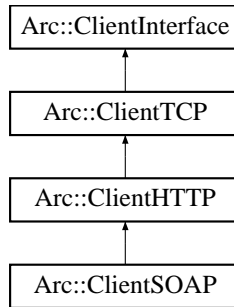
The documentation for this class was generated from the following file:

- ClientSAML2SSO.h

10.61 Arc::ClientTCP Class Reference

Class for setting up a [MCC](#) chain for TCP communication.

#include <ClientInterface.h> Inheritance diagram for Arc::ClientTCP::



Public Member Functions

- [MCC](#) * [GetEntry](#) ()
- virtual [MCC_Status Load](#) ()

10.61.1 Detailed Description

Class for setting up a [MCC](#) chain for TCP communication. The [ClientTCP](#) class is a specialization of the [ClientInterface](#) which sets up a client [MCC](#) chain for TCP communication, and optionally with a security layer on top which can be either TLS, GSI or SSL3.

10.61.2 Member Function Documentation

10.61.2.1 [MCC](#)* Arc::ClientTCP::GetEntry () [inline]

Returns entry point to TCP or TLS [MCC](#) in configured chain. To initialize entry point [Load\(\)](#) method must be called.

Reimplemented in [Arc::ClientHTTP](#), and [Arc::ClientSOAP](#).

10.61.2.2 virtual [MCC_Status](#) Arc::ClientTCP::Load () [virtual]

Initializes communication chain for this object. Call to this method in derived class is not needed if [process\(\)](#) methods are used. It is only needed if [GetEntry\(\)](#) is used before [process\(\)](#) is called.

Reimplemented from [Arc::ClientInterface](#).

Reimplemented in [Arc::ClientHTTP](#), and [Arc::ClientSOAP](#).

The documentation for this class was generated from the following file:

- [ClientInterface.h](#)

10.62 Arc::ClientX509Delegation Class Reference

Public Member Functions

- [ClientX509Delegation](#) ()
- bool [createDelegation](#) (DelegationType deleg, std::string &delegation_id)
- bool [acquireDelegation](#) (DelegationType deleg, std::string &delegation_cred, std::string &delegation_id, const std::string cred_identity="", const std::string cred_delegator_ip="", const std::string username="", const std::string password="")

10.62.1 Constructor & Destructor Documentation

10.62.1.1 Arc::ClientX509Delegation::ClientX509Delegation () [inline]

Constructor creates [MCC](#) chain and connects to server.

10.62.2 Member Function Documentation

10.62.2.1 bool Arc::ClientX509Delegation::acquireDelegation (DelegationType *deleg*, std::string & *delegation_cred*, std::string & *delegation_id*, const std::string *cred_identity* = "", const std::string *cred_delegator_ip* = "", const std::string *username* = "", const std::string *password* = "")

Acquire delegation credential from delegation service. This method should be called by intermediate service ('n+1' service as explained on above) in order to use this delegation credential on behalf of the EEC's holder.

Parameters:

deleg Delegation type

delegation_id delegation ID which is used to look up the credential by delegation service

cred_identity the identity (in case of x509 credential, it is the DN of EEC credential).

cred_delegator_ip the IP address of the credential delegator. Regard of delegation, an intermediate service should accomplish three tasks: 1. Acquire 'n' level delegation credential (which is delegated by 'n-1' level delegator) from delegation service; 1. Create 'n+1' level delegation credential to delegation service; 2. Use 'n' level delegation credential to act on behalf of the EEC's holder. In case of absense of delegation_id, the 'n-1' level delegator's IP address and credential's identity are supposed to be used for look up the delegation credential from delegation service.

10.62.2.2 bool Arc::ClientX509Delegation::createDelegation (DelegationType *deleg*, std::string & *delegation_id*)

Create the delegation credential according to the different remote delegation service. This method should be called by holder of EEC(end entity credential) which would delegate its EEC credential, or by holder of delegated credential(normally, the holder is intermediate service) which would further delegate the credential (on behalf of the original EEC's holder) (for instance, the 'n' intermediate service creates a delegation credential, then the 'n+1' intermediate service acquires this delegation credential from the delegation service and also acts on behalf of the EEC's holder by using this delegation credential).

Parameters:

deleg Delegation type

delegation_id For gridsite delegation service, the delegation_id is supposed to be created by client side, and sent to service side; for ARC delegation service, the delegation_id is supposed to be created by service side, and returned back. So for gridsite delegation service, this parameter is treated as input, while for ARC delegation service, it is treated as output.

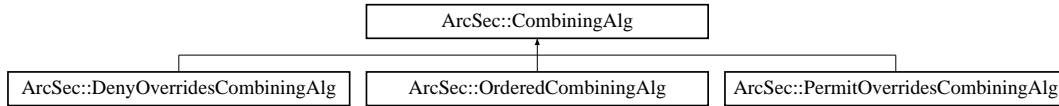
The documentation for this class was generated from the following file:

- ClientX509Delegation.h

10.63 ArcSec::CombiningAlg Class Reference

Interface for combining algrithm.

#include <CombiningAlg.h> Inheritance diagram for ArcSec::CombiningAlg::



Public Member Functions

- virtual Result [combine](#) (EvaluationCtx *ctx, std::list< [Policy](#) * > policies)=0
- virtual const std::string & [getalgId](#) (void) const =0

10.63.1 Detailed Description

Interface for combining algorithm. This class is used to implement a specific combining algorithm for combining policies.

10.63.2 Member Function Documentation

10.63.2.1 virtual Result ArcSec::CombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [pure virtual]

Evaluate request against policy, and if there are more than one policies, combine the evaluation results according to the combing algorithm implemented inside in the method combine(ctx, policies) itself.

Parameters:

- ctx** The information about request is included
- policies** The "match" and "eval" method inside each policy will be called, and then those results from each policy will be combined according to the combining algorithm inside CombiningAlg class.

Implemented in [ArcSec::DenyOverridesCombiningAlg](#), and [ArcSec::PermitOverridesCombiningAlg](#).

10.63.2.2 virtual const std::string& ArcSec::CombiningAlg::getalgId (void) const [pure virtual]

Get the identifier of the combining algorithm class

Returns:

- The identity of the algorithm

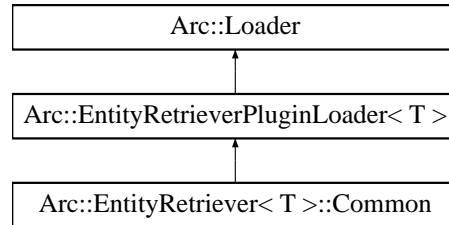
Implemented in [ArcSec::DenyOverridesCombiningAlg](#), and [ArcSec::PermitOverridesCombiningAlg](#).

The documentation for this class was generated from the following file:

- CombiningAlg.h

10.64 Arc::EntityRetriever< T >::Common Class Reference

Inheritance diagram for Arc::EntityRetriever< T >::Common::



template<typename T> class Arc::EntityRetriever< T >::Common

The documentation for this class was generated from the following file:

- EntityRetriever.h

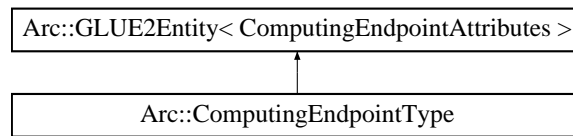
10.65 Arc::ComputingEndpointAttributes Class Reference

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.66 Arc::ComputingEndpointType Class Reference

Inheritance diagram for Arc::ComputingEndpointType::



The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

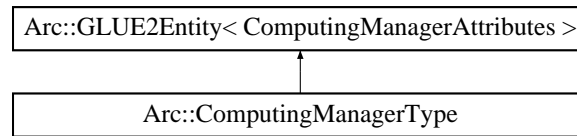
10.67 Arc::ComputingManagerAttributes Class Reference

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.68 Arc::ComputingManagerType Class Reference

Inheritance diagram for Arc::ComputingManagerType::



Data Fields

- [CountedPointer](#)< std::list< [ApplicationEnvironment](#) > > [ApplicationEnvironments](#)

10.68.1 Field Documentation

10.68.1.1 [CountedPointer](#)< std::list<[ApplicationEnvironment](#)> > [Arc::ComputingManagerType::ApplicationEnvironments](#)

[ApplicationEnvironments](#). The [ApplicationEnvironments](#) member is a list of [ApplicationEnvironment](#)'s, defined in section 6.7 [GLUE2](#).

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.69 Arc::ComputingServiceAttributes Class Reference

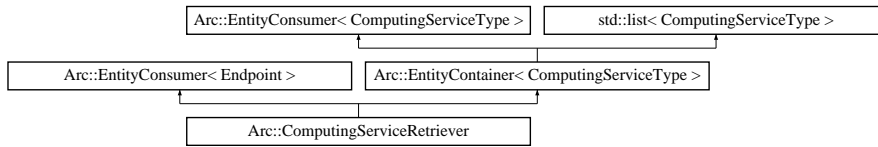
The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.70 Arc::ComputingServiceRetriever Class Reference

Retrieves information about computing elements by querying service registries and CE information systems.

#include <arc/compute/ComputingServiceRetriever.h> Inheritance diagram for Arc::ComputingServiceRetriever::



Public Member Functions

- **ComputingServiceRetriever** (const **UserConfig** &uc, const std::list< **Endpoint** > &services=std::list< **Endpoint** >(), const std::list< std::string > &rejectedServices=std::list< std::string >(), const std::set< std::string > &preferredInterfaceNames=std::set< std::string >(), const std::list< std::string > &capabilityFilter=std::list< std::string >(1, Endpoint::GetStringForCapability(Arc::Endpoint::COMPUTINGINFO)))
- void **wait** ()
- void **addEndpoint** (const **Endpoint** &service)
- void **addEntity** (const **Endpoint** &service)
- void **addConsumer** (EntityConsumer< ComputingServiceType > &c)
- void **removeConsumer** (const EntityConsumer< ComputingServiceType > &c)
- void **GetExecutionTargets** (std::list< **ExecutionTarget** > &etList)
- **EndpointStatusMap** **getAllStatuses** () const

10.70.1 Detailed Description

Retrieves information about computing elements by querying service registries and CE information systems. The **ComputingServiceRetriever** queries service registries and local information systems of computing elements, creates **ComputingServiceType** objects from the retrieved information and besides storing those objects also sends them to all the registered consumers.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 Arc::ComputingServiceRetriever::ComputingServiceRetriever (const **UserConfig** &uc, const std::list< **Endpoint** > &services = std::list< **Endpoint** >(), const std::list< std::string > &rejectedServices = std::list< std::string >(), const std::set< std::string > &preferredInterfaceNames = std::set< std::string >(), const std::list< std::string > &capabilityFilter = std::list< std::string >(1, Endpoint::GetStringForCapability(Arc::Endpoint::

Creates a **ComputingServiceRetriever** with a list of services to query.

Parameters:

← **uc** the **UserConfig** object containing the credentials to use for connecting services

- ← *services* a list of [Endpoint](#) objects containing the services (registries or CEs) to query
- ← *rejectedServices* if the [URL](#) of a service matches an element in this list, the service will not be queried
- ← *preferredInterfaceNames* when an [Endpoint](#) does not have it's GLUE2 interface name specified the class will try interfaces specified here first, and if they return no results, then all the other possible interfaces are tried
- ← *capabilityFilter* only those [ComputingServiceType](#) objects will be sent to the consumer which has at least one of the capabilities provided here

10.70.3 Member Function Documentation

10.70.3.1 void Arc::ComputingServiceRetriever::addConsumer (EntityConsumer< ComputingServiceType > & c) [inline]

Add a consumer to the [ComputingServiceRetriever](#) which will get the results. All the consumers will receive all the retrieved [ComputingServiceType](#) objects one by one.

Parameters:

- ← *c* one consumer of the type [EntityConsumer<ComputingServiceType>](#) capable of accepting [ComputingServiceType](#) objects

References [Arc::EntityRetriever< T >::addConsumer\(\)](#).

10.70.3.2 void Arc::ComputingServiceRetriever::addEndpoint (const Endpoint & service)

Adds a new service (registry or computing element) to query. Depending on the type of the service, it will be added to the internal [ServiceEndpointRetriever](#) (if it's a registry) or the internal [TargetInformationRetriever](#) (if it's a computing element).

Parameters:

- ← *service* an [Endpoint](#) referring to a service to query

Referenced by [addEntity\(\)](#).

10.70.3.3 void Arc::ComputingServiceRetriever::addEntity (const Endpoint & service) [inline, virtual]

Adds a new service to query (used by the internal [ServiceEndpointRetriever](#)). The internal [ServiceEndpointRetriever](#) queries the service registries and feeds the results back to the [ComputingServiceRetriever](#) through this method, so the [ComputingServiceRetriever](#) can recursively query the found resources.

Parameters:

- ← *service* an [Endpoint](#) referring to a service to query

Implements [Arc::EntityConsumer< Endpoint >](#).

References [addEndpoint\(\)](#).

10.70.3.4 EndpointStatusMap Arc::ComputingServiceRetriever::getAllStatuses () const [inline]

Get status of all the queried [Endpoint](#) objects. This method returns a copy of the internal status map, and thus is only a snapshot. If you want the final status map, make sure to invoke the [ComputingServiceRetriever::wait](#) method before this one.

Returns:

a map with [Endpoint](#) objects as keys and status objects as values.

References Arc::EntityRetriever< T >::getAllStatuses().

10.70.3.5 void Arc::ComputingServiceRetriever::GetExecutionTargets (std::list< ExecutionTarget > & etList) [inline]

Convenience method to generate ExecutionTarget objects from the resulted [ComputingServiceType](#) objects. Calls the class method ExecutionTarget::GetExecutionTargets with the list of retrieved ComputerServiceType objects.

Parameters:

→ *etList* the generated ExecutionTargets will be put into this list

10.70.3.6 void Arc::ComputingServiceRetriever::removeConsumer (const EntityConsumer< ComputingServiceType > & c) [inline]

Remove a previously added consumer from this [ComputingServiceRetriever](#). The removed consumer will not get any more result objects

Parameters:

← *c* the consumer to be removed

References Arc::EntityRetriever< T >::removeConsumer().

10.70.3.7 void Arc::ComputingServiceRetriever::wait (void) [inline]

Waits for all the results to arrive. This method call will only return when all the results have arrived..

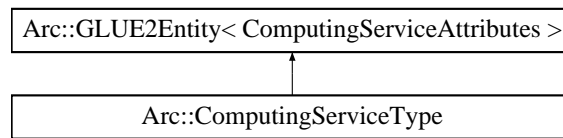
References Arc::EntityRetriever< T >::wait().

The documentation for this class was generated from the following file:

- ComputingServiceRetriever.h

10.71 Arc::ComputingServiceType Class Reference

Inheritance diagram for Arc::ComputingServiceType::

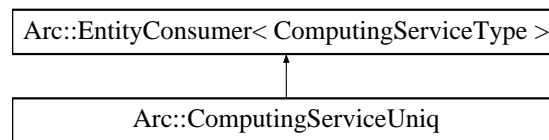


The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.72 Arc::ComputingServiceUniq Class Reference

Inheritance diagram for Arc::ComputingServiceUniq::



Public Member Functions

- void [addEntity](#) (const [ComputingServiceType](#) &service)

10.72.1 Member Function Documentation

10.72.1.1 void Arc::ComputingServiceUniq::addEntity (const ComputingServiceType &) [virtual]

Send an entity to this consumer. This is the method which will be called by the retrievers when a new result is available.

Implements [Arc::EntityConsumer< ComputingServiceType >](#).

The documentation for this class was generated from the following file:

- ComputingServiceRetriever.h

10.73 Arc::ComputingShareAttributes Class Reference

Data Fields

- std::string [Name](#)
- int [MaxMainMemory](#)
- int [MaxVirtualMemory](#)
- int [MaxDiskSpace](#)
- std::map< [Period](#), int > [FreeSlotsWithDuration](#)

10.73.1 Field Documentation

10.73.1.1 std::map<Period, int> Arc::ComputingShareAttributes::FreeSlotsWithDuration

FreeSlotsWithDuration std::map<Period, int>. This attribute express the number of free slots with their time limits. The keys in the std::map are the time limit ([Period](#)) for the number of free slots stored as the value (int). If no time limit has been specified for a set of free slots then the key will equal Period(LONG_MAX).

10.73.1.2 int Arc::ComputingShareAttributes::MaxDiskSpace

MaxDiskSpace UInt64 0..1 GB. The maximum disk space that a job is allowed use in the working; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

10.73.1.3 int Arc::ComputingShareAttributes::MaxMainMemory

MaxMainMemory UInt64 0..1 MB. The maximum physical RAM that a job is allowed to use; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

10.73.1.4 int Arc::ComputingShareAttributes::MaxVirtualMemory

MaxVirtualMemory UInt64 0..1 MB. The maximum total memory size (RAM plus swap) that a job is allowed to use; if the limit is hit, then the LRMS MAY kill the job. A negative value specifies that this member is undefined.

10.73.1.5 std::string Arc::ComputingShareAttributes::Name

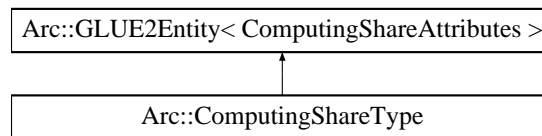
Name String 0..1. Human-readable name. This variable represents the ComputingShare.Name attribute of [GLUE2](#).

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.74 Arc::ComputingShareType Class Reference

Inheritance diagram for Arc::ComputingShareType::



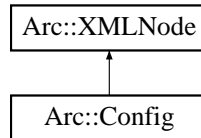
The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.75 Arc::Config Class Reference

Configuration element - represents (sub)tree of ARC XML configuration.

`#include <arc/ArcConfig.h>` Inheritance diagram for Arc::Config::



Public Member Functions

- [Config](#) ()
- [Config](#) (const [NS](#) &ns)
- [Config](#) (const char *filename)
- [Config](#) (const std::string &xml_str)
- [Config](#) ([XMLNode](#) xml)
- [Config](#) ([XMLNode](#) xml, const std::string &filename)
- [Config](#) (long cfg_ptr_addr)
- [Config](#) (const [Config](#) &cfg)
- void [print](#) (void)
- bool [parse](#) (const char *filename)
- const std::string & [getFileName](#) (void) const
- void [setFileName](#) (const std::string &filename)
- void [save](#) (const char *filename)

10.75.1 Detailed Description

Configuration element - represents (sub)tree of ARC XML configuration. This class is intended to be used to pass configuration details to various parts of HED and external modules. Currently it's just a wrapper over XML tree. But that may change in the future, although the interface should be preserved. Currently it is capable of loading an XML configuration document from a file. In future it will be capable of loading a more user-readable format and processing it into a tree-like structure convenient for machine processing (XML-like). So far there are no schema and/or namespaces assigned.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 Arc::Config::Config (XMLNode *xml*) [[inline](#)]

Acquire existing XML (sub)tree. Content is not copied. Make sure XML tree is not destroyed while in use by this object.

10.75.2.2 Arc::Config::Config (XMLNode *xml*, const std::string &*filename*) [[inline](#)]

Acquire existing XML (sub)tree and set config file. Content is not copied. Make sure XML tree is not destroyed while in use by this object.

10.75.3 Member Function Documentation

10.75.3.1 void Arc::Config::print (void)

Print structure of document for debugging purposes. Printed content is not an XML document.

The documentation for this class was generated from the following file:

- ArcConfig.h

10.76 Arc::ConfigEndpoint Class Reference

Represents the endpoint of service with a given type and [GLUE2](#) InterfaceName.

```
#include <arc/UserConfig.h>
```

Public Types

- enum [Type](#) { [REGISTRY](#), [COMPUTINGINFO](#), [ANY](#) }

Public Member Functions

- [ConfigEndpoint](#) (const std::string &[URLString](#)="", const std::string &[InterfaceName](#)="", [ConfigEndpoint::Type](#) type=[ConfigEndpoint::ANY](#))
- [operator bool](#) () const
- bool [operator!](#) () const
- bool [operator==](#) ([ConfigEndpoint](#) c) const

Data Fields

- [Type](#) type
- std::string [URLString](#)
- std::string [InterfaceName](#)
- std::string [RequestedSubmissionInterfaceName](#)

10.76.1 Detailed Description

Represents the endpoint of service with a given type and [GLUE2](#) InterfaceName. A [ConfigEndpoint](#) can be a service registry or a local information system of a computing element. It has a [URL](#), and optionally [GLUE2](#) InterfaceName and a RequestedSubmissionInterfaceName, which will be used to filter the possible job submission interfaces on a computing element.

10.76.2 Member Enumeration Documentation

10.76.2.1 enum Arc::ConfigEndpoint::Type

Types of ComputingEndpoint objects.

Enumerator:

REGISTRY a service registry

COMPUTINGINFO a local information system of a computing element

ANY both, only used for filtering, when both types are accepted

10.76.3 Constructor & Destructor Documentation

10.76.3.1 `Arc::ConfigEndpoint::ConfigEndpoint (const std::string & URLString = "", const std::string & InterfaceName = "", ConfigEndpoint::Type type = ConfigEndpoint::ANY) [inline]`

Creates a [ConfigEndpoint](#) from a [URL](#) an [InterfaceName](#) and a [Type](#).

Parameters:

- ← *URLString* is a string containing the [URL](#) of the [ConfigEndpoint](#)
- ← *InterfaceName* is a string containing the type of the interface based on the [InterfaceName](#) attribute in the [GLUE2](#) specification
- ← *type* is either [ConfigEndpoint::REGISTRY](#) or [ConfigEndpoint::COMPUTINGINFO](#)

10.76.4 Field Documentation

10.76.4.1 `std::string Arc::ConfigEndpoint::RequestedSubmissionInterfaceName`

A [GLUE2](#) [InterfaceName](#) requesting a job submission interface. This will be used when collecting information about the computing element. Only those job submission interfaces will be considered which has this requested [InterfaceName](#).

Referenced by operator==().

The documentation for this class was generated from the following file:

- [UserConfig.h](#)

10.77 Arc::ConfusaCertHandler Class Reference

```
#include <ConfusaCertHandler.h>
```

Public Member Functions

- [ConfusaCertHandler](#) (int keysize, const std::string dn)
- std::string [getCertRequestB64](#) ()
- bool [createCertRequest](#) (std::string password="", std::string storedir=".")

10.77.1 Detailed Description

Wrapper around [Credential](#) handling the Confusa specifics.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 Arc::ConfusaCertHandler::ConfusaCertHandler (int *keysizes*, const std::string *dn*)

Create a new [ConfusaCertHandler](#) for DN dn and given keysize Basically Confusa cert handler wraps around [Credential](#)

10.77.3 Member Function Documentation

10.77.3.1 bool Arc::ConfusaCertHandler::createCertRequest (std::string *password* = "", std::string *storedir* = " . /")

Create a new end entity certificate, with a private key encrypted with password password. Private key and certificate will be stored in directory storedir.

10.77.3.2 std::string Arc::ConfusaCertHandler::getCertRequestB64 ()

Get the certificate request managed by this confusa cert handler in base 64 encoding

The documentation for this class was generated from the following file:

- ConfusaCertHandler.h

10.78 Arc::ConfusaParserUtils Class Reference

```
#include <ConfusaParserUtils.h>
```

Static Public Member Functions

- static std::string [urlencode](#) (const std::string url)
- static std::string [urlencode_params](#) (const std::string url)
- static xmlDocPtr [get_doc](#) (const std::string xml_file)
- static void [destroy_doc](#) (xmlDocPtr doc)
- static std::string [extract_body_information](#) (const std::string html_string)
- static std::string [handle_redirect_step](#) (Arc::MCCConfig cfg, const std::string remote_url, std::string *cookies=NULL, std::multimap< std::string, std::string > *httpAttributes=NULL)
- static std::string [evaluate_path](#) (xmlDocPtr doc, const std::string xpathExpr, std::list< std::string > *contentList=NULL)

10.78.1 Detailed Description

Methods often needed in evaluation web pages from the Confusa WebSSO workflow

10.78.2 Member Function Documentation

10.78.2.1 static void Arc::ConfusaParserUtils::destroy_doc (xmlDocPtr *doc*) [static]

Destroy a libxml2 doc representation

10.78.2.2 static std::string Arc::ConfusaParserUtils::evaluate_path (xmlDocPtr *doc*, const std::string *xpathExpr*, std::list< std::string > * *contentList* = NULL) [static]

Evaluate the given xPathExpr on the document ptr. Return a string with the FIRST result if contentList is NULL. Return a string with the first result and all results, including the first one, in contentList if contentList is not null.

10.78.2.3 static std::string Arc::ConfusaParserUtils::extract_body_information (const std::string *html_string*) [static]

Get the part only within <body> and </body> in a HTML string For parsing, usually only this part is interesting.

10.78.2.4 static xmlDocPtr Arc::ConfusaParserUtils::get_doc (const std::string *xml_file*) [static]

Construct a libxml2 doc representation from the xml file

10.78.2.5 `static std::string Arc::ConfusaParserUtils::handle_redirect_step (Arc::MCCConfig cfg,
const std::string remote_url, std::string * cookies = NULL, std::multimap< std::string,
std::string > * httpAttributes = NULL) [static]`

Handle a single redirect step from the SAML2 WebSSO profile. Store the received cookie in **cookie* and pass the given *httpAttributes* to the site during redirect.

10.78.2.6 `static std::string Arc::ConfusaParserUtils::urlencode (const std::string url) [static]`

urlencode the passed string

10.78.2.7 `static std::string Arc::ConfusaParserUtils::urlencode_params (const std::string url)
[static]`

Urlencode the passed string with respect to the parameters. The difference to `urlencode` is that the parameters will keep their separators, i.e. the `?` and `&` separating parameters will be preserved.

The documentation for this class was generated from the following file:

- `ConfusaParserUtils.h`

10.79 Arc::CountedPointer< T > Class Template Reference

Wrapper for pointer with automatic destruction and multiple references.

```
#include <arc/Utils.h>
```

Data Structures

- class **Base**

Public Member Functions

- T & **operator*** (void) const
- T * **operator->** (void) const
- **operator bool** (void) const
- bool **operator!** (void) const
- bool **operator==** (const **CountedPointer** &p) const
- bool **operator!=** (const **CountedPointer** &p) const
- bool **operator<** (const **CountedPointer** &p) const
- T * **Ptr** (void) const
- T * **Release** (void)

10.79.1 Detailed Description

template<typename T> class Arc::CountedPointer< T >

Wrapper for pointer with automatic destruction and multiple references. If ordinary pointer is wrapped in instance of this class it will be automatically destroyed when all instances referring to it are destroyed. This is useful for maintaining pointers referred from multiple structures with automatic destruction of original object when last reference is destroyed. It is similar to Java approach with a difference that destruction time is strictly defined. Only pointers returned by new() are supported. This class is not thread-safe.

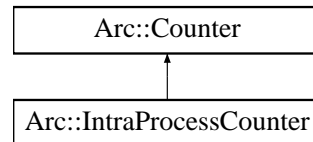
The documentation for this class was generated from the following file:

- **Utils.h**

10.80 Arc::Counter Class Reference

A class defining a common interface for counters.

`#include <arc/Counter.h>`Inheritance diagram for Arc::Counter::



Public Member Functions

- virtual `~Counter ()`
- virtual int `getLimit ()=0`
- virtual int `setLimit (int newLimit)=0`
- virtual int `changeLimit (int amount)=0`
- virtual int `getExcess ()=0`
- virtual int `setExcess (int newExcess)=0`
- virtual int `changeExcess (int amount)=0`
- virtual int `getValue ()=0`
- virtual `CounterTicket reserve (int amount=1, Glib::TimeVal duration=ETERNAL, bool prioritized=false, Glib::TimeVal timeOut=ETERNAL)=0`

Protected Types

- typedef unsigned long long int `IDType`

Protected Member Functions

- `Counter ()`
- virtual void `cancel (IDType reservationID)=0`
- virtual void `extend (IDType &reservationID, Glib::TimeVal &expiryTime, Glib::TimeVal duration=ETERNAL)=0`
- Glib::TimeVal `getCurrentTime ()`
- Glib::TimeVal `getExpiryTime (Glib::TimeVal duration)`
- `CounterTicket getCounterTicket (Counter::IDType reservationID, Glib::TimeVal expiryTime, Counter *counter)`
- `ExpirationReminder getExpirationReminder (Glib::TimeVal expTime, Counter::IDType resID)`

Friends

- class `CounterTicket`
- class `ExpirationReminder`

10.80.1 Detailed Description

A class defining a common interface for counters. This class defines a common interface for counters as well as some common functionality.

The purpose of a counter is to provide housekeeping some resource such as e.g. disk space, memory or network bandwidth. The counter itself will not be aware of what kind of resource it limits the use of. Neither will it be aware of what unit is being used to measure that resource. Counters are thus very similar to semaphores. Furthermore, counters are designed to handle concurrent operations from multiple threads/processes in a consistent manner.

Every counter has a limit, an excess limit and a value. The limit is a number that specify how many units are available for reservation. The value is the number of units that are currently available for reservation, i.e. has not already been reserved. The excess limit specify how many extra units can be reserved for high priority needs even if there are no normal units available for reservation. The excess limit is similar to the credit limit of e.g. a VISA card.

The users of the resource must thus first call the counter in order to make a reservation of an appropriate amount of the resource, then allocate and use the resource and finally call the counter again to cancel the reservation.

Typical usage is:

```
// Declare a counter. Replace XYZ by some appropriate kind of
// counter and provide required parameters. Unit is MB.
XYZCounter memory(...);
...
// Make a reservation of memory for 2000000 doubles.
CounterTicket tick = memory.reserve(2*sizeof(double));
// Use the memory.
double* A=new double[2000000];
doSomething(A);
delete[] A;
// Cancel the reservation.
tick.cancel();
```

There are also alternative ways to make reservations, including self-expiring reservations, prioritized reservations and reservations that fail if they cannot be made fast enough.

For self expiring reservations, a duration is provided in the reserve call:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0));
```

A self-expiring reservation can be cancelled explicitly before it expires, but if it is not cancelled it will expire automatically when the duration has passed. The default value for the duration is ETERNAL, which means that the reservation will not be cancelled automatically.

Prioritized reservations may use the excess limit and succeed immediately even if there are no normal units available for reservation. The value of the counter will in this case become negative. A prioritized reservation looks like this:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0), true);
```

Finally, a time out option can be provided for a reservation. If some task should be performed within two seconds or not at all, the reservation can look like this:

```
tick = memory.reserve(2*sizeof(double), Glib::TimeVal(1,0),
                    true, Glib::TimeVal(2,0));
if (tick.isValid())
    doSomething(...);
```

10.80.2 Member Typedef Documentation

10.80.2.1 typedef unsigned long long int Arc::Counter::IDType [protected]

A typedef of identification numbers for reservation. This is a type that is used as identification numbers (keys) for referencing of reservations. It is used internally in counters for book keeping of reservations as well as in the [CounterTicket](#) class in order to be able to cancel and extend reservations.

10.80.3 Constructor & Destructor Documentation

10.80.3.1 Arc::Counter::Counter () [protected]

Default constructor. This is the default constructor. Since [Counter](#) is an abstract class, it should only be used by subclasses. Therefore it is protected. Furthermore, since the [Counter](#) class has no attributes, nothing needs to be initialized and thus this constructor is empty.

10.80.3.2 virtual Arc::Counter::~~Counter () [virtual]

The destructor. This is the destructor of the [Counter](#) class. Since the [Counter](#) class has no attributes, nothing needs to be cleaned up and thus the destructor is empty.

10.80.4 Member Function Documentation

10.80.4.1 virtual void Arc::Counter::cancel (IDType *reservationID*) [protected, pure virtual]

Cancellation of a reservation. This method cancels a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters:

reservationID The identity number (key) of the reservation to cancel.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.2 virtual int Arc::Counter::changeExcess (int *amount*) [pure virtual]

Changes the excess limit of the counter. Changes the excess limit of the counter by adding a certain amount to the current excess limit.

Parameters:

amount The amount by which to change the excess limit.

Returns:

The new excess limit.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.3 virtual int Arc::Counter::changeLimit (int *amount*) [pure virtual]

Changes the limit of the counter. Changes the limit of the counter by adding a certain amount to the current limit.

Parameters:

amount The amount by which to change the limit.

Returns:

The new limit.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.4 virtual void Arc::Counter::extend (IDType & *reservationID*, Glib::TimeVal & *expiryTime*, Glib::TimeVal *duration* = ETERNAL) [protected, pure virtual]

Extension of a reservation. This method extends a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters:

reservationID Used for input as well as output. Contains the identification number of the original reservation on entry and the new identification number of the extended reservation on exit.

expiryTime Used for input as well as output. Contains the expiry time of the original reservation on entry and the new expiry time of the extended reservation on exit.

duration The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.5 CounterTicket Arc::Counter::getCounterTicket (Counter::IDType *reservationID*, Glib::TimeVal *expiryTime*, Counter * *counter*) [protected]

A "relay method" for a constructor of the [CounterTicket](#) class. This method acts as a relay for one of the constructors of the [CounterTicket](#) class. That constructor is private, but needs to be accessible from the subclasses of [Counter](#) (but not from anywhere else). In order not to have to declare every possible subclass of [Counter](#) as a friend of [CounterTicket](#), only the base class [Counter](#) is a friend and its subclasses access the constructor through this method. (If C++ had supported "package access", as Java does, this trick would not have been necessary.)

Parameters:

reservationID The identity number of the reservation corresponding to the [CounterTicket](#).

expiryTime the expiry time of the reservation corresponding to the [CounterTicket](#).

counter The [Counter](#) from which the reservation has been made.

Returns:

The counter ticket that has been created.

10.80.4.6 Glib::TimeVal Arc::Counter::getCurrentTime () [protected]

Get the current time. Returns the current time. An "adapter method" for the assign_current_time() method in the Glib::TimeVal class. return The current time.

10.80.4.7 virtual int Arc::Counter::getExcess () [pure virtual]

Returns the excess limit of the counter. Returns the excess limit of the counter, i.e. by how much the usual limit may be exceeded by prioritized reservations.

Returns:

The excess limit.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.8 ExpirationReminder Arc::Counter::getExpirationReminder (Glib::TimeVal *expTime*, Counter::IDType *resID*) [protected]

A "relay method" for the constructor of [ExpirationReminder](#). This method acts as a relay for one of the constructors of the [ExpirationReminder](#) class. That constructor is private, but needs to be accessible from the subclasses of [Counter](#) (but not from anywhere else). In order not to have to declare every possible subclass of [Counter](#) as a friend of [ExpirationReminder](#), only the base class [Counter](#) is a friend and its subclasses access the constructor through this method. (If C++ had supported "package access", as Java does, this trick would not have been necessary.)

Parameters:

expTime the expiry time of the reservation corresponding to the [ExpirationReminder](#).

resID The identity number of the reservation corresponding to the [ExpirationReminder](#).

Returns:

The [ExpirationReminder](#) that has been created.

10.80.4.9 Glib::TimeVal Arc::Counter::getExpiryTime (Glib::TimeVal *duration*) [protected]

Computes an expiry time. This method computes an expiry time by adding a duration to the current time.

Parameters:

duration The duration.

Returns:

The expiry time.

10.80.4.10 virtual int Arc::Counter::getLimit () [pure virtual]

Returns the current limit of the counter. This method returns the current limit of the counter, i.e. how many units can be reserved simultaneously by different threads without claiming high priority.

Returns:

The current limit of the counter.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.11 virtual int Arc::Counter::getValue () [pure virtual]

Returns the current value of the counter. Returns the current value of the counter, i.e. the number of unreserved units. Initially, the value is equal to the limit of the counter. When a reservation is made, the the value is decreased. Normally, the value should never be negative, but this may happen if there are prioritized reservations. It can also happen if the limit is decreased after some reservations have been made, since reservations are never revoked.

Returns:

The current value of the counter.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.12 virtual CounterTicket Arc::Counter::reserve (int amount = 1, Glib::TimeVal duration = ETERNAL, bool prioritized = false, Glib::TimeVal timeOut = ETERNAL) [pure virtual]

Makes a reservation from the counter. This method makes a reservation from the counter. If the current value of the counter is too low to allow for the reservation, the method blocks until the reservation is possible or times out.

Parameters:

amount The amount to reserve, default value is 1.

duration The duration of a self expiring reservation, default is that it lasts forever.

prioritized Whether this reservation is prioritized and thus allowed to use the excess limit.

timeOut The maximum time to block if the value of the counter is too low, default is to allow "eternal" blocking.

Returns:

A [CounterTicket](#) that can be queried about the status of the reservation as well as for cancellations and extensions.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.13 virtual int Arc::Counter::setExcess (int newExcess) [pure virtual]

Sets the excess limit of the counter. This method sets a new excess limit for the counter.

Parameters:

newExcess The new excess limit, an absolute number.

Returns:

The new excess limit.

Implemented in [Arc::IntraProcessCounter](#).

10.80.4.14 `virtual int Arc::Counter::setLimit (int newLimit) [pure virtual]`

Sets the limit of the counter. This method sets a new limit for the counter.

Parameters:

newLimit The new limit, an absolute number.

Returns:

The new limit.

Implemented in [Arc::IntraProcessCounter](#).

The documentation for this class was generated from the following file:

- Counter.h

10.81 Arc::CounterTicket Class Reference

A class for "tickets" that correspond to counter reservations.

```
#include <arc/Counter.h>
```

Public Member Functions

- [CounterTicket](#) ()
- bool [isValid](#) ()
- void [extend](#) (Glib::TimeVal duration)
- void [cancel](#) ()

Friends

- class [Counter](#)

10.81.1 Detailed Description

A class for "tickets" that correspond to counter reservations. This is a class for reservation tickets. When a reservation is made from a [Counter](#), a ReservationTicket is returned. This ticket can then be queried about the validity of a reservation. It can also be used for cancellation and extension of reservations.

Typical usage is:

```
// Declare a counter. Replace XYZ by some appropriate kind of
// counter and provide required parameters. Unit is MB.
XYZCounter memory (...);
...
// Make a reservation of memory for 2000000 doubles.
CounterTicket tick = memory.reserve(2*sizeof(double));
// Use the memory.
double* A=new double[2000000];
doSomething(A);
delete[] A;
// Cancel the reservation.
tick.cancel();
```

10.81.2 Constructor & Destructor Documentation

10.81.2.1 Arc::CounterTicket::CounterTicket ()

The default constructor. This is the default constructor. It creates a [CounterTicket](#) that is not valid. The ticket object that is created can later be assigned a ticket that is returned by the reserve() method of a [Counter](#).

10.81.3 Member Function Documentation

10.81.3.1 void Arc::CounterTicket::cancel ()

Cancels a reservation. This method is called to cancel a reservation. It may be called also for self-expiring reservations, which will then be cancelled before they were originally planned to expire.

10.81.3.2 void Arc::CounterTicket::extend (Glib::TimeVal *duration*)

Extends a reservation. Extends a self-expiring reservation. In order to succeed the extension should be made before the previous reservation expires.

Parameters:

duration The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.

10.81.3.3 bool Arc::CounterTicket::isValid ()

Returns the validity of a [CounterTicket](#). This method checks whether a [CounterTicket](#) is valid. The ticket was probably returned earlier by the `reserve()` method of a [Counter](#) but the corresponding reservation may have expired.

Returns:

The validity of the ticket.

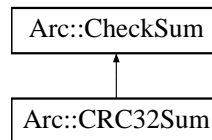
The documentation for this class was generated from the following file:

- Counter.h

10.82 Arc::CRC32Sum Class Reference

Implementation of CRC32 checksum.

`#include <arc/CheckSum.h>`Inheritance diagram for Arc::CRC32Sum::



Public Member Functions

- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *buf)
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

10.82.1 Detailed Description

Implementation of CRC32 checksum. This class is a specialized class of the [CheckSum](#) class. It provides an implementation for the CRC-32 IEEE 802.3 standard.

10.82.2 Member Function Documentation

10.82.2.1 virtual void Arc::CRC32Sum::add (void * *buf*, unsigned long long int *len*) [**virtual**]

Add data to be checksummed. This method calculates the checksum of the passed data chunk, taking into account the previous state of this object.

Parameters:

- buf* pointer to data chunk to be checksummed.
len size of the data chunk.

Implements [Arc::CheckSum](#).

10.82.2.2 virtual void Arc::CRC32Sum::end (void) [**virtual**]

Finalize the checksumming. This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::CheckSum](#).

10.82.2.3 `virtual int Arc::CRC32Sum::print (char * buf, int len) const` `[virtual]`

Retrieve result of checksum into a string. The passed string *buf* is filled with result of checksum algorithm in base 16. At most *len* characters are filled into buffer *buf*. The hexadecimal value is prepended with "algorithm:", where algorithm is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters:

buf pointer to buffer which should be filled with checksum result.
len max number of character filled into buffer.

Returns:

0 on success

Reimplemented from [Arc::Checksum](#).

10.82.2.4 `virtual void Arc::CRC32Sum::scan (const char * buf)` `[virtual]`

Set internal checksum state. This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [Checksum::print](#) method.

Parameters:

buf string containing textual representation of checksum

See also:

[Checksum::print](#)

Implements [Arc::Checksum](#).

10.82.2.5 `virtual void Arc::CRC32Sum::start (void)` `[virtual]`

Initiate the checksum algorithm. This method must be called before starting a new checksum calculation.

Implements [Arc::Checksum](#).

The documentation for this class was generated from the following file:

- CheckSum.h

10.83 Arc::Credential Class Reference

```
#include <Credential.h>
```

Public Member Functions

- [Credential](#) ()
- [Credential](#) (int keybits)
- [Credential](#) (const std::string &CAfile, const std::string &CAkey, const std::string &CAserial, const std::string &extfile, const std::string &extsect, const std::string &passphrase4key)
- [Credential](#) (Time start, Period lifetime=[Period](#)("PT12H"), int keybits=1024, std::string proxyversion="rfc", std::string policylang="inheritAll", std::string policy="", int pathlength=-1)
- [Credential](#) (const std::string &cert, const std::string &key, const std::string &cadir, const std::string &cacfile, const std::string &passphrase4key="", const bool is_file=true)
- [Credential](#) (const [UserConfig](#) &usercfg, const std::string &passphrase4key="")
- void [AddCertExtObj](#) (std::string &sn, std::string &oid)
- void [LogError](#) (void) const
- bool [GetVerification](#) (void) const
- EVP_PKEY * [GetPrivKey](#) (void) const
- EVP_PKEY * [GetPubKey](#) (void) const
- X509 * [GetCert](#) (void) const
- X509_REQ * [GetCertReq](#) (void) const
- STACK_OF (X509) * [GetCertChain](#) (void) const
- int [GetCertNumofChain](#) (void) const
- Credformat [getFormat_BIO](#) (BIO *in, const bool is_file=true) const
- std::string [GetDN](#) (void) const
- std::string [GetIdentityName](#) (void) const
- ArcCredential::certType [GetType](#) (void) const
- std::string [GetIssuerName](#) (void) const
- std::string [GetCAName](#) (void) const
- std::string [GetProxyPolicy](#) (void) const
- void [SetProxyPolicy](#) (const std::string &proxyversion, const std::string &policylang, const std::string &policy, int pathlength)
- bool [OutputPrivatekey](#) (std::string &content, bool encryption=false, const std::string &passphrase="")
- bool [OutputPublickey](#) (std::string &content)
- bool [OutputCertificate](#) (std::string &content, bool is_der=false)
- bool [OutputCertificateChain](#) (std::string &content, bool is_der=false)
- Period [GetLifeTime](#) (void) const
- Time [GetStartTime](#) () const
- Time [GetEndTime](#) () const
- void [SetLifeTime](#) (const Period &period)
- void [SetStartTime](#) (const Time &start_time)
- bool [IsValid](#) (void)
- bool [AddExtension](#) (const std::string &name, const std::string &data, bool crit=false)
- bool [AddExtension](#) (const std::string &name, char **binary)
- std::string [GetExtension](#) (const std::string &name)
- bool [GenerateEECRequest](#) (BIO *reqbio, BIO *keybio, const std::string &dn="")
- bool [GenerateEECRequest](#) (std::string &reqcontent, std::string &keycontent, const std::string &dn="")

- bool [GenerateEECRequest](#) (const char *request_filename, const char *key_filename, const std::string &dn="")
- bool [GenerateRequest](#) (BIO *bio, bool if_der=false)
- bool [GenerateRequest](#) (std::string &content, bool if_der=false)
- bool [GenerateRequest](#) (const char *filename, bool if_der=false)
- bool [InquireRequest](#) (BIO *reqbio, bool if_eec=false, bool if_der=false)
- bool [InquireRequest](#) (std::string &content, bool if_eec=false, bool if_der=false)
- bool [InquireRequest](#) (const char *filename, bool if_eec=false, bool if_der=false)
- bool [SignRequest](#) ([Credential](#) *proxy, BIO *outputbio, bool if_der=false)
- bool [SignRequest](#) ([Credential](#) *proxy, std::string &content, bool if_der=false)
- bool [SignRequest](#) ([Credential](#) *proxy, const char *filename, bool foamat=false)
- bool [SelfSignEECRequest](#) (const std::string &dn, const char *extfile, const std::string &extsect, const char *certfile)
- bool [SignEECRequest](#) ([Credential](#) *eec, const std::string &dn, BIO *outputbio)
- bool [SignEECRequest](#) ([Credential](#) *eec, const std::string &dn, std::string &content)
- bool [SignEECRequest](#) ([Credential](#) *eec, const std::string &dn, const char *filename)

Static Public Member Functions

- static void [InitProxyCertInfo](#) (void)
- static bool [IsCredentialsValid](#) (const [UserConfig](#) &usercfg)

10.83.1 Detailed Description

[Credential](#) class covers the functionality about general processing about certificate/key files, including:

1. certificate/key parsing, information extracting (such as subject name, issuer name, lifetime, etc.), chain verifying, extension processing about proxy certinfo, extension processing about other general certificate extension (such as voms attributes, it should be the extension-specific code itself to create, parse and verify the extension, not the [Credential](#) class. For voms, it is some code about writing and parsing voms-implementing Attribute Certificate/ RFC3281, the voms-attribute is then be looked as a binary part and embeded into extension of X509 certificate/proxy certificate);
2. certificate request, extension emeding and certificate signing, for both proxy certificate and EEC (end entity certificate) certificate

The [Credential](#) class support PEM, DER PKCS12 credential.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 [Arc::Credential::Credential \(\)](#)

Default constructor, only acts as a container for inquiring certificate request, is meaningless for any other use.

10.83.2.2 [Arc::Credential::Credential \(int keybits\)](#)

Constructor with user-defined keylength. Needed for creation of EE certs, since some applications will only support keys with a certain minimum length > 1024

10.83.2.3 Arc::Credential::Credential (const std::string & CAfile, const std::string & CAkey, const std::string & CAserial, const std::string & extfile, const std::string & extsect, const std::string & passphrase4key)

Constructor, specific constructor for CA certificate is meaningless for any other use.

10.83.2.4 Arc::Credential::Credential (Time start, Period lifetime = Period ("PT12H"), int keybits = 1024, std::string proxyversion = "rfc", std::string policylang = "inheritAll", std::string policy = "", int pathlength = -1)

Constructor, specific constructor for proxy certificate, only acts as a container for constraining certificate signing and/or generating certificate request(only keybits is useful for creating certificate request), is meaningless for any other use. The proxyversion and policylang is for specifying the proxy certificate type and the policy language inside proxy. The definition of proxyversion and policy language is based on http://dev.globus.org/wiki/Security/ProxyCertTypes#RFC_3820_Proxy_Certificates The code is supposed to support proxy version: GSI2(legacy proxy), GSI3(Proxy draft) and RFC(RFC3820 proxy), and corresponding policy language. GSI2(GSI2, GSI2_LIMITED) GSI3 and RFC (IMPERSONATION_PROXY--1.3.6.1.5.5.7.21.1, INDEPENDENT_PROXY--1.3.6.1.5.5.7.21.2, LIMITED_PROXY--1.3.6.1.4.1.3536.1.1.1.9, RESTRICTED_PROXY--policy language undefined) In openssl>=0.9.8, there are three types of policy languages: id-ppl-inheritAll--1.3.6.1.5.5.7.21.1, id-ppl-independent--1.3.6.1.5.5.7.21.2, and id-ppl-anyLanguage-1.3.6.1.5.5.7.21.0

Parameters:

start, start time of proxy certificate

lifetime, lifetime of proxy certificate

keybits, modulus size for RSA key generation, it should be greater than 1024 if 'this' class is used for generating X509 request; it should be '0' if 'this' class is used for constraining certificate signing.

10.83.2.5 Arc::Credential::Credential (const std::string & cert, const std::string & key, const std::string & cadir, const std::string & cfile, const std::string & passphrase4key = "", const bool is_file = true)

Constructor, specific constructor for usual certificate, constructing from credential files. only acts as a container for parsing the certificate and key files, is meaningless for any other use. this constructor will parse the credential information, and put them into "this" object

Parameters:

passphrase4key, specifies the password for decrypting private key (if needed). If value is empty then password will be asked interactively. To avoid asking for password use value provided by NoPassword() method.

is_file, specifies if the cert/key are from file, otherwise they are supposed to be from string. default is from file

10.83.2.6 Arc::Credential::Credential (const UserConfig & usercfg, const std::string & passphrase4key = "")

Constructor, specific constructor for usual certificate, constructing from information in UserConfig object. Only acts as a container for parsing the certificate and key files, is meaningless for any other use. this constructor will parse the credential * information, and put them into "this" object

Parameters:

is_file,specify if the cert/key are from file, otherwise they are supposed to be from string. default is from file

10.83.3 Member Function Documentation**10.83.3.1 void Arc::Credential::AddCertExtObj (std::string & *sn*, std::string & *oid*)**

General method for adding a new nid into openssl's global const

10.83.3.2 bool Arc::Credential::AddExtension (const std::string & *name*, char ** *binary*)

Add an extension to the extension part of the certificate

Parameters:

binary,the data which will be inserted into certificate extension part as a specific extension there should be specific methods defined inside specific X509V3_EXT_METHOD structure to parse the specific extension format. For example, VOMS attribute certificate is a specific extension to proxy certificate. There is specific X509V3_EXT_METHOD defined in [VOMSAttribute.h](#) and VOMSAttribute.c for parsing attribute certificate. In openssl, the specific X509V3_EXT_METHOD can be got according to the extension name/id, see X509V3_EXT_get_nid(ext_nid)

10.83.3.3 bool Arc::Credential::AddExtension (const std::string & *name*, const std::string & *data*, bool *crit* = false)

Add an extension to the extension part of the certificate

Parameters:

name,the name of the extension, there OID related with the name should be registered into openssl firstly

data,the data which will be inserted into certificate extension

10.83.3.4 bool Arc::Credential::GenerateEECRequest (const char * *request_filename*, const char * *key_filename*, const std::string & *dn* = "")

Generate an EEC request, output the certificate request and the key to a file

10.83.3.5 bool Arc::Credential::GenerateEECRequest (std::string & *reqcontent*, std::string & *keycontent*, const std::string & *dn* = "")

Generate an EEC request, output the certificate request to a string

10.83.3.6 bool Arc::Credential::GenerateEECRequest (BIO * *reqbio*, BIO * *keybio*, const std::string & *dn* = "")

Generate an EEC request, based on the keybits and signing algorithm information inside this object output the certificate request to output BIO

The user will be asked for a private key password

10.83.3.7 `bool Arc::Credential::GenerateRequest (const char * filename, bool if_der = false)`

Generate a proxy request, output the certificate request to a file

10.83.3.8 `bool Arc::Credential::GenerateRequest (std::string & content, bool if_der = false)`

Generate a proxy request, output the certificate request to a string

10.83.3.9 `bool Arc::Credential::GenerateRequest (BIO * bio, bool if_der = false)`

Generate a proxy request, base on the keybits and signing algorithm information inside this object output the certificate request to output BIO

10.83.3.10 `std::string Arc::Credential::GetCAName (void) const`

Get CA of the certificate attached to this object, if the certificate is an EEC, GetCAName get the same value as GetIssuerName

10.83.3.11 `X509* Arc::Credential::GetCert (void) const`

Get the certificate attached to this object

10.83.3.12 `int Arc::Credential::GetCertNumofChain (void) const`

Get the number of certificates in the certificate chain attached to this object

10.83.3.13 `X509_REQ* Arc::Credential::GetCertReq (void) const`

Get the certificate request, if there is any

10.83.3.14 `std::string Arc::Credential::GetDN (void) const`

Get the DN of the certificate attached to this object

10.83.3.15 `Time Arc::Credential::GetEndTime () const`

Returns validity end time of certificate or proxy

10.83.3.16 `std::string Arc::Credential::GetExtension (const std::string & name)`

Get the specific extension (named by the parameter) in a certificate this function is only supposed to be called after certificate and key are loaded by the constructor for usual certificate

Parameters:

name,the name of the extension to get

10.83.3.17 Credformat Arc::Credential::getFormat_BIO (BIO * *in*, const bool *is_file* = `true`) const

Get the certificate format, PEM PKCS12 or DER BIO could be memory or file, they should be processed differently.

10.83.3.18 std::string Arc::Credential::GetIdentityName (void) const

Get the Identity name of the certificate attached to this object, the result will not include proxy CN

10.83.3.19 std::string Arc::Credential::GetIssuerName (void) const

Get issuer of the certificate attached to this object

10.83.3.20 Period Arc::Credential::GetLifeTime (void) const

Returns lifetime of certificate or proxy

10.83.3.21 EVP_PKEY* Arc::Credential::GetPrivKey (void) const

Get the private key attached to this object

10.83.3.22 std::string Arc::Credential::GetProxyPolicy (void) const

Get the proxy policy attached to the "proxy certificate information" extension of the proxy certificate

10.83.3.23 EVP_PKEY* Arc::Credential::GetPubKey (void) const

Get the public key attached to this object

10.83.3.24 Time Arc::Credential::GetStartTime () const

Returns validity start time of certificate or proxy

10.83.3.25 ArcCredential::certType Arc::Credential::GetType (void) const

Get type of the certificate attached to this object

10.83.3.26 bool Arc::Credential::GetVerification (void) const [inline]

Get the verification result about certificate chain checking

10.83.3.27 static void Arc::Credential::InitProxyCertInfo (void) [static]

Initiate nid for proxy certificate extension

10.83.3.28 bool Arc::Credential::InquireRequest (const char * *filename*, bool *if_eec* = false, bool *if_der* = false)

Inquire the certificate request from a file

10.83.3.29 bool Arc::Credential::InquireRequest (std::string & *content*, bool *if_eec* = false, bool *if_der* = false)

Inquire the certificate request from a string

10.83.3.30 bool Arc::Credential::InquireRequest (BIO * *reqbio*, bool *if_eec* = false, bool *if_der* = false)

Inquire the certificate request from BIO, and put the request information to X509_REQ inside this object, and parse the certificate type from the PROXYCERTINFO of request' extension

Parameters:

if_der false for PEM; true for DER

10.83.3.31 static bool Arc::Credential::IsCredentialsValid (const UserConfig & *usercfg*) [static]

Returns true if credentials are valid. Credentials are read from locations specified in [UserConfig](#) object. This method is deprecated. [User](#) per-instance method [IsValid\(\)](#) instead.

10.83.3.32 bool Arc::Credential::IsValid (void)

Returns true if credentials are valid

10.83.3.33 void Arc::Credential::LogError (void) const

Log error information related with openssl

10.83.3.34 bool Arc::Credential::OutputCertificate (std::string & *content*, bool *is_der* = false)

Output the certificate into string

Parameters:

is_der false for PEM, true for DER

10.83.3.35 `bool Arc::Credential::OutputCertificateChain (std::string & content, bool is_der = false)`

Output the certificate chain into string

Parameters:

is_der false for PEM, true for DER

10.83.3.36 `bool Arc::Credential::OutputPrivatekey (std::string & content, bool encryption = false, const std::string & passphrase = "")`

Output the private key into string

Parameters:

encryption,whether encrypt the output private key or not

passphrase,the passphrase to encrypt the output private key

10.83.3.37 `bool Arc::Credential::OutputPublickey (std::string & content)`

Output the public key into string

10.83.3.38 `bool Arc::Credential::SelfSignEECRequest (const std::string & dn, const char * extfile, const std::string & extsect, const char * certfile)`

Self sign a certificate. This functionality is specific for creating a CA credential by using this [Credential](#) class.

Parameters:

dn the DN for the subject

extfile the configuration file which includes the extension information, typically the openssl.cnf file

extsect the section/group name for the extension, e.g. in openssl.cnf, usr_cert and v3_ca

certfile the certificate file, which contains the signed certificate

10.83.3.39 `void Arc::Credential::SetLifeTime (const Period & period)`

Set lifetime of certificate or proxy

10.83.3.40 `void Arc::Credential::SetProxyPolicy (const std::string & proxyversion, const std::string & policylang, const std::string & policy, int pathlength)`

Set the proxy policy attached to the "proxy certificate information" extension of the proxy certificate

10.83.3.41 `void Arc::Credential::SetStartTime (const Time & start_time)`

Set start time of certificate or proxy

10.83.3.42 `bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, const char * filename)`

Sign request and output the signed certificate to a file

10.83.3.43 `bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, std::string & content)`

Sign request and output the signed certificate to a string

10.83.3.44 `bool Arc::Credential::SignEECRequest (Credential * eec, const std::string & dn, BIO * outputbio)`

Sign eec request, and output the signed certificate to output BIO

10.83.3.45 `bool Arc::Credential::SignRequest (Credential * proxy, const char * filename, bool foamat = false)`

Sign request and output the signed certificate to a file

Parameters:

if_der false for PEM, true for DER

10.83.3.46 `bool Arc::Credential::SignRequest (Credential * proxy, std::string & content, bool if_der = false)`

Sign request and output the signed certificate to a string

Parameters:

if_der false for PEM, true for DER

10.83.3.47 `bool Arc::Credential::SignRequest (Credential * proxy, BIO * outputbio, bool if_der = false)`

Sign request based on the information inside proxy, and output the signed certificate to output BIO

Parameters:

if_der false for PEM, true for DER

10.83.3.48 `Arc::Credential::STACK_OF (X509) const`

Get the certificate chain attached to this object

The documentation for this class was generated from the following file:

- Credential.h

10.84 Arc::CredentialError Class Reference

```
#include <Credential.h>
```

Public Member Functions

- [CredentialError](#) (const std::string &what="")

10.84.1 Detailed Description

This is an exception class that is used to handle runtime errors discovered in the [Credential](#) class.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 Arc::CredentialError::CredentialError (const std::string & *what* = "")

This is the constructor of the [CredentialError](#) class.

Parameters:

- what* An explanation of the error.

The documentation for this class was generated from the following file:

- Credential.h

10.85 Arc::CredentialStore Class Reference

```
#include <CredentialStore.h>
```

10.85.1 Detailed Description

This class provides functionality for storing delegated credentials and retrieving them from some store services. This is very preliminary implementation and currently support only one type of credentials - X.509 proxies, and only one type of store service - MyProxy. Later it will be extended to support at least following services: ARC delegation service, VOMS service, local file system.

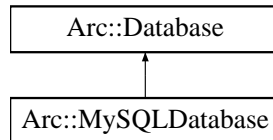
The documentation for this class was generated from the following file:

- CredentialStore.h

10.86 Arc::Database Class Reference

Interface for calling database client library.

`#include <arc/DBInterface.h>` Inheritance diagram for Arc::Database::



Public Member Functions

- [Database](#) ()
- [Database](#) (std::string &server, int port)
- [Database](#) (const [Database](#) &other)
- virtual [~Database](#) ()
- virtual bool [connect](#) (std::string &dbname, std::string &user, std::string &password)=0
- virtual bool [isconnected](#) () const =0
- virtual void [close](#) ()=0
- virtual bool [enable_ssl](#) (const std::string &keyfile="", const std::string &certfile="", const std::string &cafile="", const std::string &capath="")=0
- virtual bool [shutdown](#) ()=0

10.86.1 Detailed Description

Interface for calling database client library. For different types of database client library, different classes should be implemented by implementing this interface.

10.86.2 Member Function Documentation

10.86.2.1 virtual bool Arc::Database::connect (std::string & *dbname*, std::string & *user*, std::string & *password*) [pure virtual]

Do connection with database server.

Parameters:

dbname The database name which will be used.

user The username which will be used to access database.

password The password which will be used to access database.

Implemented in [Arc::MySQLDatabase](#).

10.86.2.2 virtual bool Arc::Database::enable_ssl (const std::string & *keyfile* = "", const std::string & *certfile* = "", const std::string & *cafile* = "", const std::string & *capath* = "") [pure virtual]

Enable ssl communication for the connection.

Parameters:

- keyfile* The location of key file.
- certfile* The location of certificate file.
- cafile* The location of ca file.
- capath* The location of ca directory

Implemented in [Arc::MySQLDatabase](#).

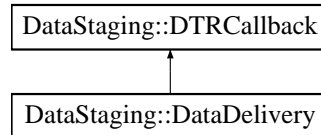
The documentation for this class was generated from the following file:

- DBInterface.h

10.87 DataStaging::DataDelivery Class Reference

[DataDelivery](#) transfers data between specified physical locations.

#include <arc/data-staging/DataDelivery.h> Inheritance diagram for DataStaging::DataDelivery::



Public Member Functions

- [DataDelivery](#) ()
- [~DataDelivery](#) ()
- virtual void [receiveDTR](#) ([DTR_ptr](#) request)
- bool [cancelDTR](#) ([DTR_ptr](#) request)
- bool [start](#) ()
- bool [stop](#) ()
- void [SetTransferParameters](#) (const [TransferParameters](#) ¶ms)

10.87.1 Detailed Description

[DataDelivery](#) transfers data between specified physical locations. [start\(\)](#) must be called to start the delivery thread for processing DTRs and [stop\(\)](#) should be called to stop it (this waits for all data transfers to exit). [stop\(\)](#) is also called in the destructor.

All meta-operations for a [DTR](#) such as resolving replicas must be done before sending to [DataDelivery](#). Calling [receiveDTR\(\)](#) starts a new process which performs data transfer as specified in [DTR](#).

10.87.2 Member Function Documentation

10.87.2.1 virtual void DataStaging::DataDelivery::receiveDTR ([DTR_ptr](#) request) [virtual]

Pass a [DTR](#) to Delivery. This method is called by the scheduler to pass a [DTR](#) to the delivery. The [DataDelivery](#) starts the data transfer either using a local process or by sending a request to a remote delivery service, and then returns. [DataDelivery](#)'s own thread then monitors the transfer.

Implements [DataStaging::DTRCallback](#).

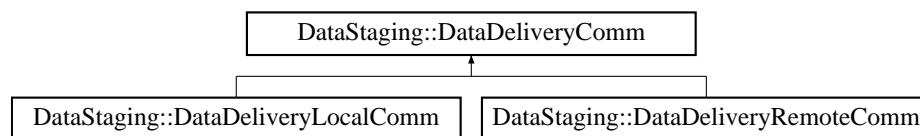
The documentation for this class was generated from the following file:

- [DataDelivery.h](#)

10.88 DataStaging::DataDeliveryComm Class Reference

This class provides an abstract interface for the Delivery layer.

`#include <arc/data-staging/DataDeliveryComm.h>`Inheritance diagram for DataStaging::DataDeliveryComm::



Data Structures

- struct [Status](#)

Plain C struct to pass information from executing process back to main thread.

Public Types

- enum [CommStatusType](#) {
[CommInit](#), [CommNoError](#), [CommTimeout](#), [CommClosed](#),
[CommExited](#), [CommFailed](#) }

Public Member Functions

- virtual [~DataDeliveryComm](#) ()
- [Status](#) [GetStatus](#) () const
- std::string [GetError](#) () const
- virtual [operator bool](#) () const =0
- virtual bool [operator!](#) () const =0

Static Public Member Functions

- static [DataDeliveryComm](#) * [CreateInstance](#) ([DTR_ptr](#) dtr, const [TransferParameters](#) ¶ms)
- static bool [CheckComm](#) ([DTR_ptr](#) dtr, std::vector< std::string > &allowed_dirs)

Protected Member Functions

- virtual void [PullStatus](#) ()=0
- [DataDeliveryComm](#) ([DTR_ptr](#) dtr, const [TransferParameters](#) ¶ms)

Protected Attributes

- [Status](#) `status_`
- [Status](#) `status_buf_`
- unsigned int `status_pos_`
- [Glib::Mutex](#) `lock_`
- [DataDeliveryCommHandler](#) * `handler_`
- `std::string` `dtr_id`
- [TransferParameters](#) `transfer_params`
- [Arc::Time](#) `start_`
- [DTRLogger](#) `logger_`

10.88.1 Detailed Description

This class provides an abstract interface for the Delivery layer. Different implementations provide different ways of providing Delivery functionality. [DataDeliveryLocalComm](#) launches a local process to perform the transfer and [DataDeliveryRemoteComm](#) contacts a remote service which performs the transfer. The implementation is chosen depending on what is set in the [DTR](#), which the [Scheduler](#) should set based on various factors.

[CreateInstance\(\)](#) should be used to get a pointer to the instantiated object. This also starts the transfer. Deleting this object stops the transfer and cleans up any used resources. A singleton instance of [DataDeliveryCommHandler](#) regularly polls all active transfers using [PullStatus\(\)](#) and fills the [Status](#) object with current information, which can be obtained through [GetStatus\(\)](#).

10.88.2 Member Enumeration Documentation

10.88.2.1 enum [DataStaging::DataDeliveryComm::CommStatusType](#)

Communication status with transfer.

Enumerator:

CommInit Initializing/starting transfer, rest of information not valid.

CommNoError Communication going on smoothly.

CommTimeout Communication experienced timeout.

CommClosed Communication channel was closed.

CommExited Transfer exited. Mostly same as *CommClosed* but exit detected before pipe closed.

CommFailed Transfer failed. If we have *CommFailed* and no error code reported that normally means segfault or external kill.

10.88.3 Constructor & Destructor Documentation

10.88.3.1 [DataStaging::DataDeliveryComm::DataDeliveryComm](#) ([DTR_ptr](#) *dtr*, const [TransferParameters](#) & *params*) [[protected](#)]

Start transfer with parameters taken from [DTR](#) and supplied transfer limits. Constructor should not be used directly, [CreateInstance\(\)](#) should be used instead.

10.88.4 Member Function Documentation

10.88.4.1 static bool DataStaging::DataDeliveryComm::CheckComm (DTR_ptr *dtr*, std::vector< std::string > & *allowed_dirs*) [static]

Check the delivery method is available. Calls CheckComm of the appropriate subclass.

Parameters:

dtr [DTR](#) from which credentials are used

allowed_dirs filled with list of dirs that this comm is allowed to read/write

Returns:

true if selected delivery method is available

Reimplemented in [DataStaging::DataDeliveryLocalComm](#), and [DataStaging::DataDeliveryRemoteComm](#).

10.88.4.2 virtual void DataStaging::DataDeliveryComm::PullStatus () [protected, pure virtual]

Check for new state and fill state accordingly. This method is periodically called by the comm handler to obtain status info. It detects communication and delivery failures and delivery termination.

Implemented in [DataStaging::DataDeliveryLocalComm](#), and [DataStaging::DataDeliveryRemoteComm](#).

The documentation for this class was generated from the following file:

- [DataDeliveryComm.h](#)

10.89 DataStaging::DataDeliveryCommHandler Class Reference

Singleton class handling all active [DataDeliveryComm](#) objects.

```
#include <arc/data-staging/DataDeliveryComm.h>
```

Public Member Functions

- void [Add](#) ([DataDeliveryComm](#) *item)
- void [Remove](#) ([DataDeliveryComm](#) *item)

Static Public Member Functions

- static [DataDeliveryCommHandler](#) * [getInstance](#) ()

10.89.1 Detailed Description

Singleton class handling all active [DataDeliveryComm](#) objects.

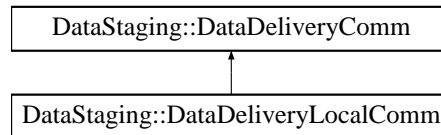
The documentation for this class was generated from the following file:

- [DataDeliveryComm.h](#)

10.90 DataStaging::DataDeliveryLocalComm Class Reference

This class starts, monitors and controls a local Delivery process.

`#include <arc/data-staging/DataDeliveryLocalComm.h>`
Inheritance diagram for DataStaging::DataDeliveryLocalComm::



Public Member Functions

- [DataDeliveryLocalComm](#) ([DTR_ptr](#) dtr, const [TransferParameters](#) ¶ms)
- virtual [~DataDeliveryLocalComm](#) ()
- virtual void [PullStatus](#) ()
- virtual [operator bool](#) () const
- virtual bool [operator!](#) () const

Static Public Member Functions

- static bool [CheckComm](#) ([DTR_ptr](#) dtr, std::vector< std::string > &allowed_dirs)

10.90.1 Detailed Description

This class starts, monitors and controls a local Delivery process.

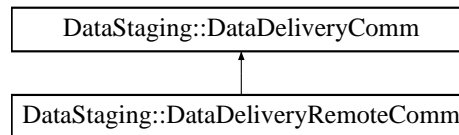
The documentation for this class was generated from the following file:

- DataDeliveryLocalComm.h

10.91 DataStaging::DataDeliveryRemoteComm Class Reference

This class contacts a remote service to make a Delivery request.

#include <arc/data-staging/DataDeliveryRemoteComm.h> Inheritance diagram for DataStaging::DataDeliveryRemoteComm::



Public Member Functions

- [DataDeliveryRemoteComm](#) ([DTR_ptr](#) dtr, const [TransferParameters](#) ¶ms)
- virtual [~DataDeliveryRemoteComm](#) ()
- virtual void [PullStatus](#) ()
- virtual [operator bool](#) () const
- virtual bool [operator!](#) () const

Static Public Member Functions

- static bool [CheckComm](#) ([DTR_ptr](#) dtr, std::vector< std::string > &allowed_dirs)

10.91.1 Detailed Description

This class contacts a remote service to make a Delivery request.

The documentation for this class was generated from the following file:

- DataDeliveryRemoteComm.h

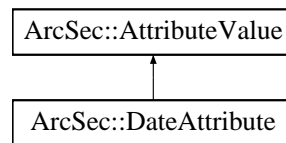
10.92 Arc::DataStagingType Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.93 ArcSec::DateAttribute Class Reference

Inheritance diagram for ArcSec::DateAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.93.1 Member Function Documentation

10.93.1.1 virtual std::string ArcSec::DateAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.93.1.2 virtual std::string ArcSec::DateAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.93.1.3 virtual std::string ArcSec::DateAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

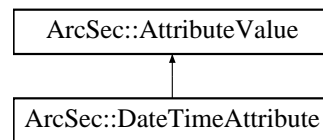
Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- DateTimeAttribute.h

10.94 ArcSec::DateTimeAttribute Class Reference

#include <DateTimeAttribute.h> Inheritance diagram for ArcSec::DateTimeAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.94.1 Detailed Description

Format: YYYYMMDDHHMMSSZ Day Month DD HH:MM:SS YYYY YYYY-MM-DD HH:MM:SS
YYYY-MM-DDTHH:MM:SS+HH:MM YYYY-MM-DDTHH:MM:SSZ

10.94.2 Member Function Documentation

10.94.2.1 virtual std::string ArcSec::DateTimeAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.94.2.2 virtual std::string ArcSec::DateTimeAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.94.2.3 virtual std::string ArcSec::DateTimeAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

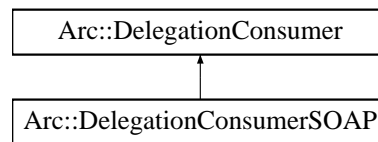
Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- DateTimeAttribute.h

10.95 Arc::DelegationConsumer Class Reference

#include <DelegationInterface.h> Inheritance diagram for Arc::DelegationConsumer::



Public Member Functions

- [DelegationConsumer](#) (void)
- [DelegationConsumer](#) (const std::string &content)
- const std::string & [ID](#) (void)
- bool [Backup](#) (std::string &content)
- bool [Restore](#) (const std::string &content)
- bool [Request](#) (std::string &content)
- bool [Acquire](#) (std::string &content)
- bool [Acquire](#) (std::string &content, std::string &identity)

Protected Member Functions

- bool [Generate](#) (void)
- void [LogError](#) (void)

10.95.1 Detailed Description

A consumer of delegated X509 credentials. During delegation procedure this class acquires delegated credentials aka proxy - certificate, private key and chain of previous certificates. Delegation procedure consists of calling [Request\(\)](#) method for generating certificate request followed by call to [Acquire\(\)](#) method for making complete credentials from certificate chain.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 Arc::DelegationConsumer::DelegationConsumer (void)

Creates object with new private key

10.95.2.2 Arc::DelegationConsumer::DelegationConsumer (const std::string & content)

Creates object with provided private key

10.95.3 Member Function Documentation

10.95.3.1 bool Arc::DelegationConsumer::Acquire (std::string & content, std::string & identity)

Includes the functionality of Acquire(content) plus extracting the credential identity.

10.95.3.2 bool Arc::DelegationConsumer::Acquire (std::string & *content*)

Ads private key into certificates chain in 'content' On exit content contains complete delegated credentials.

10.95.3.3 bool Arc::DelegationConsumer::Backup (std::string & *content*)

Stores content of this object into a string

10.95.3.4 bool Arc::DelegationConsumer::Generate (void) [protected]

Private key

10.95.3.5 const std::string& Arc::DelegationConsumer::ID (void)

Return identifier of this object - not implemented

10.95.3.6 void Arc::DelegationConsumer::LogError (void) [protected]

Creates private key

10.95.3.7 bool Arc::DelegationConsumer::Request (std::string & *content*)

Make X509 certificate request from internal private key

10.95.3.8 bool Arc::DelegationConsumer::Restore (const std::string & *content*)

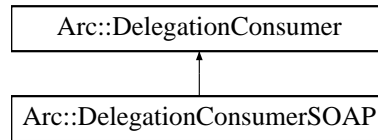
Restores content of object from string

The documentation for this class was generated from the following file:

- DelegationInterface.h

10.96 Arc::DelegationConsumerSOAP Class Reference

#include <DelegationInterface.h> Inheritance diagram for Arc::DelegationConsumerSOAP::



Public Member Functions

- [DelegationConsumerSOAP](#) (void)
- [DelegationConsumerSOAP](#) (const std::string &content)
- bool [DelegateCredentialsInit](#) (const std::string &id, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [UpdateCredentials](#) (std::string &credentials, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [UpdateCredentials](#) (std::string &credentials, std::string &identity, const SOAPEnvelope &in, SOAPEnvelope &out)
- bool [DelegatedToken](#) (std::string &credentials, [XMLNode](#) token)

10.96.1 Detailed Description

This class extends [DelegationConsumer](#) to support SOAP message exchange. Implements WS interface <http://www.nordugrid.org/schemas/delegation> described in delegation.wsdl.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 Arc::DelegationConsumerSOAP::DelegationConsumerSOAP (void)

Creates object with new private key

10.96.2.2 Arc::DelegationConsumerSOAP::DelegationConsumerSOAP (const std::string &content)

Creates object with specified private key

10.96.3 Member Function Documentation

10.96.3.1 bool Arc::DelegationConsumerSOAP::DelegateCredentialsInit (const std::string &id, const SOAPEnvelope &in, SOAPEnvelope &out)

Process SOAP message which starts delegation. Generated message in 'out' is meant to be sent back to DelagationProviderSOAP. Argument 'id' contains identifier of procedure and is used only to produce SOAP message.

10.96.3.2 `bool Arc::DelegationConsumerSOAP::DelegatedToken (std::string & credentials, XMLNode token)`

Similar to UpdateCredentials but takes only DelegatedToken XML element

10.96.3.3 `bool Arc::DelegationConsumerSOAP::UpdateCredentials (std::string & credentials, std::string & identity, const SOAPEnvelope & in, SOAPEnvelope & out)`

Includes the functionality in above UpdateCredentials method; plus extracting the credential identity

10.96.3.4 `bool Arc::DelegationConsumerSOAP::UpdateCredentials (std::string & credentials, const SOAPEnvelope & in, SOAPEnvelope & out)`

Accepts delegated credentials. Process 'in' SOAP message and stores full proxy credentials in 'credentials'. 'out' message is generated for sending to DelagationProviderSOAP.

The documentation for this class was generated from the following file:

- DelegationInterface.h

10.97 Arc::DelegationContainerSOAP Class Reference

```
#include <DelegationInterface.h>
```

Public Member Functions

- bool [DelegatedToken](#) (std::string &credentials, [XMLNode](#) token, const std::string &client="")
- bool [MatchNamespace](#) (const SOAPEnvelope &in)
- std::string [GetFailure](#) (void)

Protected Member Functions

- virtual [DelegationConsumerSOAP](#) * [AddConsumer](#) (std::string &id, const std::string &client)
- virtual [DelegationConsumerSOAP](#) * [FindConsumer](#) (const std::string &id, const std::string &client)
- virtual bool [TouchConsumer](#) ([DelegationConsumerSOAP](#) *c, const std::string &credentials)
- virtual bool [QueryConsumer](#) ([DelegationConsumerSOAP](#) *c, std::string &credentials)
- virtual void [ReleaseConsumer](#) ([DelegationConsumerSOAP](#) *c)
- virtual void [RemoveConsumer](#) ([DelegationConsumerSOAP](#) *c)
- virtual void [CheckConsumers](#) (void)
- bool [DelegateCredentialsInit](#) (const SOAPEnvelope &in, SOAPEnvelope &out, const std::string &client="")
- bool [UpdateCredentials](#) (std::string &credentials, const SOAPEnvelope &in, SOAPEnvelope &out, const std::string &client="")

Protected Attributes

- std::string [failure_](#)
- int [max_size_](#)
- int [max_duration_](#)
- int [max_usage_](#)
- bool [context_lock_](#)

10.97.1 Detailed Description

Manages multiple delegated credentials. Delegation consumers are created automatically with DelegateCredentialsInit method up to max_size_ and assigned unique identifier. It's methods are similar to those of [DelegationConsumerSOAP](#) with identifier included in SOAP message used to route execution to one of managed [DelegationConsumerSOAP](#) instances.

10.97.2 Member Function Documentation

10.97.2.1 virtual [DelegationConsumerSOAP](#)* Arc::DelegationContainerSOAP::AddConsumer (std::string &id, const std::string &client) [protected, virtual]

Creates new consumer object, if empty assigns id and stores in intenal store

10.97.2.2 `virtual void Arc::DelegationContainerSOAP::CheckConsumers (void) [protected, virtual]`

Periodic management of stored consumers

10.97.2.3 `bool Arc::DelegationContainerSOAP::DelegateCredentialsInit (const SOAPEnvelope & in, SOAPEnvelope & out, const std::string & client = "") [protected]`

See [DelegationConsumerSOAP::DelegateCredentialsInit](#) If 'client' is not empty then all subsequent calls involving access to generated credentials must contain same value in their 'client' arguments.

10.97.2.4 `bool Arc::DelegationContainerSOAP::DelegatedToken (std::string & credentials, XMLNode token, const std::string & client = "")`

See [DelegationConsumerSOAP::DelegatedToken](#)

10.97.2.5 `virtual DelegationConsumerSOAP* Arc::DelegationContainerSOAP::FindConsumer (const std::string & id, const std::string & client) [protected, virtual]`

Finds previously created consumer in internal store

10.97.2.6 `std::string Arc::DelegationContainerSOAP::GetFailure (void)`

Returns textual description of last failure.

10.97.2.7 `bool Arc::DelegationContainerSOAP::MatchNamespace (const SOAPEnvelope & in)`

Match namespace of SOAP request against supported interfaces. Returns true if namespace is supported.

10.97.2.8 `virtual bool Arc::DelegationContainerSOAP::QueryConsumer (DelegationConsumerSOAP * c, std::string & credentials) [protected, virtual]`

Obtain stored credentials - not all containers may provide this functionality

10.97.2.9 `virtual void Arc::DelegationContainerSOAP::ReleaseConsumer (DelegationConsumerSOAP * c) [protected, virtual]`

Releases consumer obtained by call to [AddConsumer\(\)](#) or [FindConsumer\(\)](#)

10.97.2.10 `virtual void Arc::DelegationContainerSOAP::RemoveConsumer (DelegationConsumerSOAP * c) [protected, virtual]`

Releases consumer obtained by call to [AddConsumer\(\)](#) or [FindConsumer\(\)](#) and deletes it

10.97.2.11 `virtual bool Arc::DelegationContainerSOAP::TouchConsumer
(DelegationConsumerSOAP * c, const std::string & credentials) [protected,
virtual]`

Marks consumer as recently used and acquire new credentials

10.97.2.12 `bool Arc::DelegationContainerSOAP::UpdateCredentials (std::string & credentials,
const SOAPEnvelope & in, SOAPEnvelope & out, const std::string & client = "")
[protected]`

See [DelegationConsumerSOAP::UpdateCredentials](#)

10.97.3 Field Documentation

10.97.3.1 `bool Arc::DelegationContainerSOAP::context_lock_ [protected]`

If true delegation consumer is deleted when connection context is destroyed

10.97.3.2 `int Arc::DelegationContainerSOAP::max_duration_ [protected]`

Lifetime of unused delegation consumer

10.97.3.3 `int Arc::DelegationContainerSOAP::max_size_ [protected]`

Max. number of delegation consumers

10.97.3.4 `int Arc::DelegationContainerSOAP::max_usage_ [protected]`

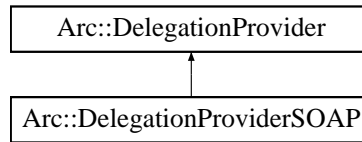
Max. times same delegation consumer may accept credentials

The documentation for this class was generated from the following file:

- DelegationInterface.h

10.98 Arc::DelegationProvider Class Reference

#include <DelegationInterface.h> Inheritance diagram for Arc::DelegationProvider::



Public Member Functions

- [DelegationProvider](#) (const std::string &credentials)
- [DelegationProvider](#) (const std::string &cert_file, const std::string &key_file, std::istream *inpwd=NULL)
- std::string [Delegate](#) (const std::string &request, const DelegationRestrictions &restrictions=DelegationRestrictions())

10.98.1 Detailed Description

A provider of delegated credentials. During delegation procedure this class generates new credential to be used in proxy/delegated credential.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 Arc::DelegationProvider::DelegationProvider (const std::string & *credentials*)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain PEM-encoded certificate, private key and optionally certificates chain.

10.98.2.2 Arc::DelegationProvider::DelegationProvider (const std::string & *cert_file*, const std::string & *key_file*, std::istream * *inpwd* = NULL)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain filesystem path to PEM-encoded certificate and private key. Optionally cert_file may contain certificates chain.

10.98.3 Member Function Documentation

10.98.3.1 std::string Arc::DelegationProvider::Delegate (const std::string & *request*, const DelegationRestrictions & *restrictions* = DelegationRestrictions())

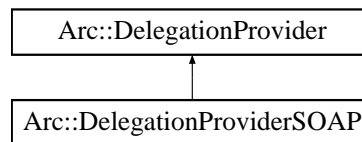
Perform delegation. Takes X509 certificate request and creates proxy credentials excluding private key. Result is then to be fed into [DelegationConsumer::Acquire](#)

The documentation for this class was generated from the following file:

- DelegationInterface.h

10.99 Arc::DelegationProviderSOAP Class Reference

#include <DelegationInterface.h> Inheritance diagram for Arc::DelegationProviderSOAP::



Public Member Functions

- [DelegationProviderSOAP](#) (const std::string &credentials)
- [DelegationProviderSOAP](#) (const std::string &cert_file, const std::string &key_file, std::istream *inpwd=NULL)
- bool [DelegateCredentialsInit](#) ([MCCInterface](#) &mcc_interface, [MessageContext](#) *context, ServiceType stype=ARCDelagation)
- bool [DelegateCredentialsInit](#) ([MCCInterface](#) &mcc_interface, [MessageAttributes](#) *attributes_in, [MessageAttributes](#) *attributes_out, [MessageContext](#) *context, ServiceType stype=ARCDelagation)
- bool [UpdateCredentials](#) ([MCCInterface](#) &mcc_interface, [MessageContext](#) *context, const DelegationRestrictions &restrictions=DelegationRestrictions(), ServiceType stype=ARCDelagation)
- bool [UpdateCredentials](#) ([MCCInterface](#) &mcc_interface, [MessageAttributes](#) *attributes_in, [MessageAttributes](#) *attributes_out, [MessageContext](#) *context, const DelegationRestrictions &restrictions=DelegationRestrictions(), ServiceType stype=ARCDelagation)
- bool [DelegatedToken](#) ([XMLNode](#) parent)
- const std::string & [ID](#) (void)

10.99.1 Detailed Description

Extension of [DelegationProvider](#) with SOAP exchange interface. This class is also a temporary container for intermediate information used during delegation procedure.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 Arc::DelegationProviderSOAP::DelegationProviderSOAP (const std::string &credentials)

Creates instance from provided credentials. Credentials are used to sign delegated credentials.

10.99.2.2 Arc::DelegationProviderSOAP::DelegationProviderSOAP (const std::string & cert_file, const std::string & key_file, std::istream * inpwd = NULL)

Creates instance from provided credentials. Credentials are used to sign delegated credentials. Arguments should contain filesystem path to PEM-encoded certificate and private key. Optionally cert_file may contain certificates chain.

10.99.3 Member Function Documentation

10.99.3.1 `bool Arc::DelegationProviderSOAP::DelegateCredentialsInit (MCCInterface & mcc_interface, MessageAttributes * attributes_in, MessageAttributes * attributes_out, MessageContext * context, ServiceType stype = ARCDelagation)`

Extended version of DelegateCredentialsInit(MCCInterface&,MessageContext*). Additionally takes attributes for request and response message to make fine control on message processing possible.

10.99.3.2 `bool Arc::DelegationProviderSOAP::DelegateCredentialsInit (MCCInterface & mcc_interface, MessageContext * context, ServiceType stype = ARCDelagation)`

Performs DelegateCredentialsInit SOAP operation. As result request for delegated credentials is received by this instance and stored internally. Call to UpdateCredentials should follow.

10.99.3.3 `bool Arc::DelegationProviderSOAP::DelegatedToken (XMLNode parent)`

Generates DelegatedToken element. Element is created as child of provided XML element and contains structure described in delegation.wsdl.

10.99.3.4 `const std::string& Arc::DelegationProviderSOAP::ID (void) [inline]`

Returns the identifier provided by service accepting delegated credentials. This identifier may then be used to refer to credentials stored at service.

10.99.3.5 `bool Arc::DelegationProviderSOAP::UpdateCredentials (MCCInterface & mcc_interface, MessageAttributes * attributes_in, MessageAttributes * attributes_out, MessageContext * context, const DelegationRestrictions & restrictions = DelegationRestrictions (), ServiceType stype = ARCDelagation)`

Extended version of UpdateCredentials(MCCInterface&,MessageContext*). Additionally takes attributes for request and response message to make fine control on message processing possible.

10.99.3.6 `bool Arc::DelegationProviderSOAP::UpdateCredentials (MCCInterface & mcc_interface, MessageContext * context, const DelegationRestrictions & restrictions = DelegationRestrictions (), ServiceType stype = ARCDelagation)`

Performs UpdateCredentials SOAP operation. This concludes delegation procedure and passes delegated credentials to [DelegationConsumerSOAP](#) instance.

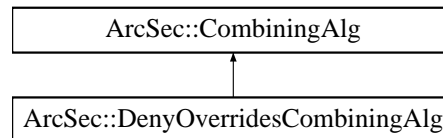
The documentation for this class was generated from the following file:

- DelegationInterface.h

10.100 ArcSec::DenyOverridesCombiningAlg Class Reference

Implement the "Deny-Overrides" algorithm.

```
#include <DenyOverridesAlg.h>Inheritance      diagram      for      Arc-
Sec::DenyOverridesCombiningAlg::
```



Public Member Functions

- virtual Result [combine](#) (EvaluationCtx *ctx, std::list< [Policy](#) * > policies)
- virtual const std::string & [getalgId](#) (void) const

10.100.1 Detailed Description

Implement the "Deny-Overrides" algorithm. Deny-Overrides, scans the policy set which is given as the parameters of "combine" method, if gets "deny" result from any policy, then stops scanning and gives "deny" as result, otherwise gives "permit".

10.100.2 Member Function Documentation

10.100.2.1 virtual Result ArcSec::DenyOverridesCombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [virtual]

If there is one policy which return negative evaluation result, then omit the other policies and return DECISION_DENY

Parameters:

- ctx* This object contains request information which will be used to evaluated against policy.
- policies* This is a container which contains policy objects.

Returns:

The combined result according to the algorithm.

Implements [ArcSec::CombiningAlg](#).

10.100.2.2 virtual const std::string& ArcSec::DenyOverridesCombiningAlg::getalgId (void) const [inline, virtual]

Get the identifier

Implements [ArcSec::CombiningAlg](#).

The documentation for this class was generated from the following file:

- DenyOverridesAlg.h

10.101 Arc::DiskSpaceRequirementType Class Reference

Data Fields

- [Range< int > DiskSpace](#)
- [int CacheDiskSpace](#)
- [int SessionDiskSpace](#)

10.101.1 Field Documentation

10.101.1.1 int Arc::DiskSpaceRequirementType::CacheDiskSpace

Specifies the required size of cache which must be available to the job in mega-bytes (MB). A negative value undefines this attribute

10.101.1.2 Range<int> Arc::DiskSpaceRequirementType::DiskSpace

Specifies the required size of disk space which must be available to the job in mega-bytes (MB). A negative value undefines this attribute

10.101.1.3 int Arc::DiskSpaceRequirementType::SessionDiskSpace

Specifies the required size of job session disk space which must be available to the job in mega-byte (MB). A negative value undefines this attribute.

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

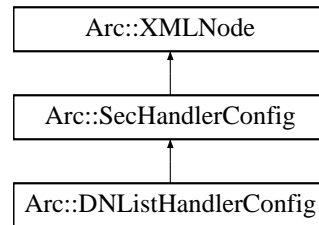
10.102 Arc::PluginsFactory::modules_t::diterator Class Reference

The documentation for this class was generated from the following file:

- Plugin.h

10.103 Arc::DNListHandlerConfig Class Reference

Inheritance diagram for Arc::DNListHandlerConfig::



The documentation for this class was generated from the following file:

- ClientInterface.h

10.104 DataStaging::DTR Class Reference

Data Transfer Request.

```
#include <arc/data-staging/DTR.h>
```

Public Member Functions

- [DTR](#) (const std::string &source, const std::string &destination, const [Arc::UserConfig](#) &usercfg, const std::string &jobid, const uid_t &uid, [DTRLogger](#) log)
- [~DTR](#) ()
- [operator bool](#) () const
- [bool operator!](#) () const
- void [registerCallback](#) ([DTRCallback](#) *cb, [StagingProcesses](#) owner)
- void [reset](#) ()
- void [set_id](#) (const std::string &id)
- std::string [get_id](#) () const
- std::string [get_short_id](#) () const
- [Arc::DataHandle](#) & [get_source](#) ()
- [Arc::DataHandle](#) & [get_destination](#) ()
- std::string [get_source_str](#) () const
- std::string [get_destination_str](#) () const
- const [Arc::UserConfig](#) & [get_usercfg](#) () const
- void [set_timeout](#) (time_t value)
- [Arc::Time](#) [get_timeout](#) () const
- void [set_process_time](#) (const [Arc::Period](#) &process_time)
- [Arc::Time](#) [get_process_time](#) () const
- [Arc::Time](#) [get_creation_time](#) () const
- [Arc::Time](#) [get_modification_time](#) () const
- std::string [get_parent_job_id](#) () const
- void [set_priority](#) (int pri)
- int [get_priority](#) () const
- void [set_rfc_proxy](#) (bool rfc)
- bool [is_rfc_proxy](#) () const
- void [set_transfer_share](#) (const std::string &share_name)
- std::string [get_transfer_share](#) () const
- void [set_sub_share](#) (const std::string &share)
- std::string [get_sub_share](#) () const
- void [set_tries_left](#) (unsigned int tries)
- unsigned int [get_tries_left](#) () const
- unsigned int [get_initial_tries](#) () const
- void [decrease_tries_left](#) ()
- void [set_status](#) ([DTRStatus](#) stat)
- [DTRStatus](#) [get_status](#) ()
- void [set_error_status](#) ([DTRErrorStatus::DTRErrorStatusType](#) error_stat, [DTRErrorStatus::DTRErrorLocation](#) error_loc, const std::string &desc="")
- void [reset_error_status](#) ()
- [DTRErrorStatus](#) [get_error_status](#) ()
- void [set_bytes_transferred](#) (unsigned long long int bytes)
- unsigned long long int [get_bytes_transferred](#) () const

- void [set_cancel_request](#) ()
- bool [cancel_requested](#) () const
- void [set_delivery_endpoint](#) (const [Arc::URL](#) &endpoint)
- const [Arc::URL](#) & [get_delivery_endpoint](#) () const
- void [add_problematic_delivery_service](#) (const [Arc::URL](#) &endpoint)
- const std::vector< [Arc::URL](#) > & [get_problematic_delivery_services](#) () const
- void [host_cert_for_remote_delivery](#) (bool host)
- bool [host_cert_for_remote_delivery](#) () const
- void [set_cache_file](#) (const std::string &filename)
- std::string [get_cache_file](#) () const
- void [set_cache_parameters](#) (const [DTRCacheParameters](#) ¶m)
- const [DTRCacheParameters](#) & [get_cache_parameters](#) () const
- void [set_cache_state](#) ([CacheState](#) state)
- [CacheState](#) [get_cache_state](#) () const
- void [set_mapped_source](#) (const std::string &file="")
- std::string [get_mapped_source](#) () const
- [StagingProcesses](#) [get_owner](#) () const
- [Arc::User](#) [get_local_user](#) () const
- void [set_replication](#) (bool rep)
- bool [is_replication](#) () const
- void [set_force_registration](#) (bool force)
- bool [is_force_registration](#) () const
- void [set_bulk_start](#) (bool value)
- bool [get_bulk_start](#) () const
- void [set_bulk_end](#) (bool value)
- bool [get_bulk_end](#) () const
- bool [bulk_possible](#) ()
- const [DTRLogger](#) & [get_logger](#) () const
- void [connect_logger](#) ()
- void [disconnect_logger](#) ()
- bool [suspend](#) ()
- bool [error](#) () const
- bool [is_destined_for_pre_processor](#) () const
- bool [is_destined_for_post_processor](#) () const
- bool [is_destined_for_delivery](#) () const
- bool [came_from_pre_processor](#) () const
- bool [came_from_post_processor](#) () const
- bool [came_from_delivery](#) () const
- bool [came_from_generator](#) () const
- bool [is_in_final_state](#) () const

Static Public Member Functions

- static void [push](#) ([DTR_ptr](#) dtr, [StagingProcesses](#) new_owner)

Static Public Attributes

- static const [Arc::URL](#) [LOCAL_DELIVERY](#)
- static [Arc::LogLevel](#) [LOG_LEVEL](#)

10.104.1 Detailed Description

Data Transfer Request. [DTR](#) stands for Data Transfer Request and a [DTR](#) describes a data transfer between two endpoints, a source and a destination. There are several parameters and options relating to the transfer contained in a [DTR](#). The normal workflow is for a Generator to create a [DTR](#) and send it to the [Scheduler](#) for processing using `DTR::push(SCHEDULER)`. If the Generator is a subclass of [DTRCallback](#), when the [Scheduler](#) has finished with the [DTR](#) the [DTRCallback::receiveDTR\(\)](#) callback method is called.

DTRs should always be used through the [Arc::ThreadedPointer](#) `DTR_ptr`. This ensures proper memory management when passing DTRs among various threads. To enforce this policy the copy constructor and assignment operator are private.

A lock protects member variables that are likely to be accessed and modified by multiple threads.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 DataStaging::DTR::DTR (const std::string & *source*, const std::string & *destination*, const Arc::UserConfig & *usercfg*, const std::string & *jobid*, const uid_t & *uid*, DTRLogger *log*)

Normal constructor. Construct a new [DTR](#).

Parameters:

source Endpoint from which to read data

destination Endpoint to which to write data

usercfg Provides some user configuration information

jobid ID of the job associated with this data transfer

uid UID to use when accessing local file system if source or destination is a local file. If this is different to the current uid then the current uid must have sufficient privileges to change uid.

log ThreadedPointer containing log object. If NULL the root logger is used.

10.104.3 Member Function Documentation

10.104.3.1 void DataStaging::DTR::add_problematic_delivery_service (const Arc::URL & *endpoint*) [inline]

Add problematic endpoint. Should only be those endpoints where there is a problem with the service itself and not the transfer.

10.104.3.2 void DataStaging::DTR::registerCallback (DTRCallback * *cb*, StagingProcesses *owner*)

Register callback objects to be used during [DTR](#) processing. Objects deriving from [DTRCallback](#) can be registered with this method. The callback method of these objects will then be called when the [DTR](#) is passed to the specified owner. Protected by lock.

10.104.3.3 void DataStaging::DTR::reset ()

Reset information held on this [DTR](#), such as resolved replicas, error state etc. Useful when a failed [DTR](#) is to be retried.

10.104.3.4 `void DataStaging::DTR::set_error_status (DTErrorStatus::DTErrorStatusType error_stat, DTErrorStatus::DTErrorLocation error_loc, const std::string & desc = "")`

Set the error status. The [DTErrorStatus](#) last error state field is set to the current status of the [DTR](#). Protected by lock.

The documentation for this class was generated from the following file:

- [DTR.h](#)

10.105 DataStaging::DTRCacheParameters Class Reference

The configured cache directories.

```
#include <arc/data-staging/DTR.h>
```

Public Member Functions

- [DTRCacheParameters](#) (void)
- [DTRCacheParameters](#) (std::vector< std::string > caches, std::vector< std::string > remote_caches, std::vector< std::string > drain_caches)

Data Fields

- std::vector< std::string > [cache_dirs](#)
- std::vector< std::string > [remote_cache_dirs](#)
- std::vector< std::string > [drain_cache_dirs](#)

10.105.1 Detailed Description

The configured cache directories.

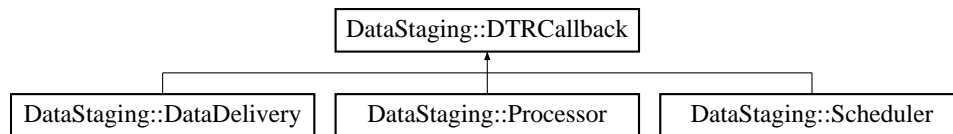
The documentation for this class was generated from the following file:

- DTR.h

10.106 DataStaging::DTRCallback Class Reference

The base class from which all callback-enabled classes should be derived.

#include <arc/data-staging/DTR.h> Inheritance diagram for DataStaging::DTRCallback::



Public Member Functions

- virtual [~DTRCallback](#) ()
- virtual void [receiveDTR](#) ([DTR_ptr](#) dtr)=0

10.106.1 Detailed Description

The base class from which all callback-enabled classes should be derived. This class is a container for a callback method which is called when a [DTR](#) is to be passed to a component. Several components in data staging (eg [Scheduler](#), [Generator](#)) are subclasses of [DTRCallback](#), which allows them to receive DTRs through the callback system.

10.106.2 Member Function Documentation

10.106.2.1 virtual void DataStaging::DTRCallback::receiveDTR ([DTR_ptr](#) dtr) [pure virtual]

Defines the callback method called when a [DTR](#) is pushed to this object. The automatic memory management of [DTR_ptr](#) ensures that the [DTR](#) object is only deleted when the last copy is deleted.

Implemented in [DataStaging::DataDelivery](#), [DataStaging::Processor](#), and [DataStaging::Scheduler](#).

The documentation for this class was generated from the following file:

- [DTR.h](#)

10.107 DataStaging::DTRErrorStatus Class Reference

A class to represent error states reported by various components.

```
#include <arc/data-staging/DTRStatus.h>
```

Public Types

- enum [DTRErrorStatusType](#) {
[NONE_ERROR](#), [INTERNAL_LOGIC_ERROR](#), [INTERNAL_PROCESS_ERROR](#), [SELF_REPLICATION_ERROR](#),
[CACHE_ERROR](#), [TEMPORARY_REMOTE_ERROR](#), [PERMANENT_REMOTE_ERROR](#),
[LOCAL_FILE_ERROR](#),
[TRANSFER_SPEED_ERROR](#), [STAGING_TIMEOUT_ERROR](#) }
- enum [DTRErrorLocation](#) {
[NO_ERROR_LOCATION](#), [ERROR_SOURCE](#), [ERROR_DESTINATION](#), [ERROR_TRANSFER](#),
[ERROR_UNKNOWN](#) }

Public Member Functions

- [DTRErrorStatus](#) ([DTRErrorStatusType](#) status, [DTRStatus::DTRStatusType](#) error_state, [DTRErrorLocation](#) location, const std::string &desc="")
- [DTRErrorStatus](#) ()
- [DTRErrorStatusType](#) [GetErrorStatus](#) () const
- [DTRStatus::DTRStatusType](#) [GetLastErrorState](#) () const
- [DTRErrorLocation](#) [GetErrorLocation](#) () const
- std::string [GetDesc](#) () const
- bool [operator==](#) (const [DTRErrorStatusType](#) &s) const
- bool [operator==](#) (const [DTRErrorStatus](#) &s) const
- bool [operator!=](#) (const [DTRErrorStatusType](#) &s) const
- bool [operator!=](#) (const [DTRErrorStatus](#) &s) const
- [DTRErrorStatus](#) & [operator=](#) (const [DTRErrorStatusType](#) &s)

10.107.1 Detailed Description

A class to represent error states reported by various components.

10.107.2 Member Enumeration Documentation

10.107.2.1 enum DataStaging::DTRErrorStatus::DTRErrorLocation

Describes where the error occurred.

Enumerator:

- [NO_ERROR_LOCATION](#)* No error.
- [ERROR_SOURCE](#)* Error with source.
- [ERROR_DESTINATION](#)* Error with destination.

ERROR_TRANSFER Error during transfer not directly related to source or destination.

ERROR_UNKNOWN Error occurred in an unknown location.

10.107.2.2 enum DataStaging::DTRErrorStatus::DTRErrorStatusType

A list of error types.

Enumerator:

NONE_ERROR No error.

INTERNAL_LOGIC_ERROR Internal error in Data Staging logic.

INTERNAL_PROCESS_ERROR Internal processing error, like losing contact with external process.

SELF_REPLICATION_ERROR Attempt to replicate a file to itself.

CACHE_ERROR Permanent error with cache.

TEMPORARY_REMOTE_ERROR Temporary error with remote service.

PERMANENT_REMOTE_ERROR Permanent error with remote service.

LOCAL_FILE_ERROR Error with local file.

TRANSFER_SPEED_ERROR Transfer rate was too slow.

STAGING_TIMEOUT_ERROR Waited for too long to become staging.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 DataStaging::DTRErrorStatus::DTRErrorStatus (DTRErrorStatusType *status*, DTRStatus::DTRStatusType *error_state*, DTRErrorLocation *location*, const std::string & *desc* = "") [inline]

Create a new [DTRErrorStatus](#) with given error states.

Parameters:

status Type of error

error_state [DTR](#) state in which the error occurred

location Location of error (at source, destination or during transfer)

desc Text description of error

The documentation for this class was generated from the following file:

- DTRStatus.h

10.108 DataStaging::DTRLList Class Reference

Global list of all active DTRs in the system.

```
#include <arc/data-staging/DTRLList.h>
```

Public Member Functions

- bool [add_dtr](#) ([DTR_ptr](#) DTRToAdd)
- bool [delete_dtr](#) ([DTR_ptr](#) DTRToDelete)
- bool [filter_dtrs_by_owner](#) ([StagingProcesses](#) OwnerToFilter, std::list< [DTR_ptr](#) > &FilteredList)
- int [number_of_dtrs_by_owner](#) ([StagingProcesses](#) OwnerToFilter)
- bool [filter_dtrs_by_status](#) ([DTRStatus::DTRStatusType](#) StatusToFilter, std::list< [DTR_ptr](#) > &FilteredList)
- bool [filter_dtrs_by_statuses](#) (const std::vector< [DTRStatus::DTRStatusType](#) > &StatusesToFilter, std::list< [DTR_ptr](#) > &FilteredList)
- bool [filter_dtrs_by_statuses](#) (const std::vector< [DTRStatus::DTRStatusType](#) > &StatusesToFilter, std::map< [DTRStatus::DTRStatusType](#), std::list< [DTR_ptr](#) > > &FilteredList)
- bool [filter_dtrs_by_next_receiver](#) ([StagingProcesses](#) NextReceiver, std::list< [DTR_ptr](#) > &FilteredList)
- bool [filter_pending_dtrs](#) (std::list< [DTR_ptr](#) > &FilteredList)
- bool [filter_dtrs_by_job](#) (const std::string &jobid, std::list< [DTR_ptr](#) > &FilteredList)
- void [caching_started](#) ([DTR_ptr](#) request)
- void [caching_finished](#) ([DTR_ptr](#) request)
- bool [is_being_cached](#) ([DTR_ptr](#) DTRToCheck)
- bool [empty](#) ()
- std::list< std::string > [all_jobs](#) ()
- void [dumpState](#) (const std::string &path)

10.108.1 Detailed Description

Global list of all active DTRs in the system. This class contains several methods for filtering the list by owner, state etc.

10.108.2 Member Function Documentation

10.108.2.1 void DataStaging::DTRLList::dumpState (const std::string & *path*)

Dump state of all current DTRs to a destination, eg file, database, url... Currently only file is supported.

Parameters:

path Path to the file in which to dump state.

10.108.2.2 bool DataStaging::DTRLList::filter_dtrs_by_job (const std::string & *jobid*, std::list< [DTR_ptr](#) > & *FilteredList*)

Get the list of DTRs corresponding to the given job ID.

Parameters:

jobid Job id to filter on

FilteredList This list is filled with filtered DTRs

10.108.2.3 **bool DataStaging::DTRLList::filter_dtrs_by_next_receiver** (StagingProcesses *NextReceiver*, `std::list< DTR_ptr > & FilteredList`)

Select DTRs that are about to go to the specified process. This selection is actually a virtual queue for pre-, post-processor and delivery.

Parameters:

NextReceiver The process to filter on

FilteredList This list is filled with filtered DTRs

10.108.2.4 **bool DataStaging::DTRLList::filter_dtrs_by_owner** (StagingProcesses *OwnerToFilter*, `std::list< DTR_ptr > & FilteredList`)

Filter the queue to select DTRs owned by a specified process.

Parameters:

OwnerToFilter The owner to filter on

FilteredList This list is filled with filtered DTRs

10.108.2.5 **bool DataStaging::DTRLList::filter_dtrs_by_status** (DTRStatus::DTRStatusType *StatusToFilter*, `std::list< DTR_ptr > & FilteredList`)

Filter the queue to select DTRs with particular status. If we have only one common queue for all DTRs, this method is necessary to make virtual queues for the DTRs about to go into the pre-, post-processor or delivery stages.

Parameters:

StatusToFilter [DTR](#) status to filter on

FilteredList This list is filled with filtered DTRs

10.108.2.6 **bool DataStaging::DTRLList::filter_dtrs_by_statuses** (const `std::vector< DTRStatus::DTRStatusType > & StatusesToFilter`, `std::map< DTRStatus::DTRStatusType, std::list< DTR_ptr > > & FilteredList`)

Filter the queue to select DTRs with particular statuses.

Parameters:

StatusesToFilter Vector of [DTR](#) statuses to filter on

FilteredList This map is filled with filtered DTRs, one list per state.

10.108.2.7 `bool DataStaging::DTRLList::filter_dtrs_by_statuses (const std::vector< DTRStatus::DTRStatusType > & StatusesToFilter, std::list< DTR_ptr > & FilteredList)`

Filter the queue to select DTRs with particular statuses.

Parameters:

StatusesToFilter Vector of [DTR](#) statuses to filter on

FilteredList This list is filled with filtered DTRs

10.108.2.8 `bool DataStaging::DTRLList::filter_pending_dtrs (std::list< DTR_ptr > & FilteredList)`

Select DTRs that have just arrived from pre-, post-processor, delivery or generator. These DTRs need some reaction from the scheduler. This selection is actually a virtual queue of DTRs that need to be processed.

Parameters:

FilteredList This list is filled with filtered DTRs

The documentation for this class was generated from the following file:

- DTRLList.h

10.109 DataStaging::DTRStatus Class Reference

Class representing the status of a [DTR](#).

```
#include <arc/data-staging/DTRStatus.h>
```

Public Types

- enum [DTRStatusType](#) {
[NEW](#), [CHECK_CACHE](#), [CHECKING_CACHE](#), [CACHE_WAIT](#),
[CACHE_CHECKED](#), [RESOLVE](#), [RESOLVING](#), [RESOLVED](#),
[QUERY_REPLICA](#), [QUERYING_REPLICA](#), [REPLICA_QUERIED](#), [PRE_CLEAN](#),
[PRE_CLEANNING](#), [PRE_CLEANNED](#), [STAGE_PREPARE](#), [STAGING_PREPARING](#),
[STAGING_PREPARING_WAIT](#), [STAGED_PREPARED](#), [TRANSFER](#), [TRANSFERRING](#),
[TRANSFERRING_CANCEL](#), [TRANSFERRED](#), [RELEASE_REQUEST](#), [RELEASING_-REQUEST](#),
[REQUEST_RELEASED](#), [REGISTER_REPLICA](#), [REGISTERING_REPLICA](#), [REPLICA_-REGISTERED](#),
[PROCESS_CACHE](#), [PROCESSING_CACHE](#), [CACHE_PROCESSED](#), [DONE](#),
[CANCELLED](#), [CANCELLED_FINISHED](#), [ERROR](#), [NULL_STATE](#) }

Public Member Functions

- [DTRStatus](#) (const [DTRStatusType](#) &status, std::string desc="")
- [DTRStatus](#) ()
- bool [operator==](#) (const [DTRStatusType](#) &s) const
- bool [operator==](#) (const [DTRStatus](#) &s) const
- bool [operator!=](#) (const [DTRStatusType](#) &s) const
- bool [operator!=](#) (const [DTRStatus](#) &s) const
- [DTRStatus](#) & [operator=](#) (const [DTRStatusType](#) &s)
- std::string [str](#) () const
- void [SetDesc](#) (const std::string &d)
- std::string [GetDesc](#) () const
- [DTRStatusType](#) [GetStatus](#) () const

Static Public Attributes

- static const std::vector< [DTRStatus::DTRStatusType](#) > [ToProcessStates](#)
- static const std::vector< [DTRStatus::DTRStatusType](#) > [ProcessingStates](#)
- static const std::vector< [DTRStatus::DTRStatusType](#) > [StagedStates](#)

10.109.1 Detailed Description

Class representing the status of a [DTR](#).

10.109.2 Member Enumeration Documentation

10.109.2.1 enum DataStaging::DTRStatus::DTRStatusType

Possible state values.

Enumerator:

NEW Just created.

CHECK_CACHE Check the cache for the file may be already there.

CHECKING_CACHE Checking the cache.

CACHE_WAIT Cache file is locked, waiting for its release.

CACHE_CHECKED Cache check completed.

RESOLVE Resolve a meta-protocol.

RESOLVING Resolving replicas.

RESOLVED Replica resolution completed.

QUERY_REPLICA Query a replica.

QUERYING_REPLICA Replica is being queried.

REPLICA_QUERIED Replica was queried.

PRE_CLEAN The destination should be deleted.

PRE_CLEANNING Deleting the destination.

PRE_CLEANNED The destination file has been deleted.

STAGE_PREPARE Prepare or stage the source and/or destination.

STAGING_PREPARING Making a staging or preparing request.

STAGING_PREPARING_WAIT Wait for the status of the staging/preparing request.

STAGED_PREPARED Staging/preparing request completed.

TRANSFER Transfer ready and can be started.

TRANSFERRING Transfer is going.

TRANSFERRING_CANCEL Transfer is on-going but scheduled for cancellation.

TRANSFERRED Transfer completed.

RELEASE_REQUEST Transfer finished, release requests on the storage.

RELEASING_REQUEST Releasing staging/preparing request.

REQUEST_RELEASED Release of staging/preparing request completed.

REGISTER_REPLICA Register a new replica of the destination.

REGISTERING_REPLICA Registering a replica in an index service.

REPLICA_REGISTERED Replica registration completed.

PROCESS_CACHE Destination is cacheable, process cache.

PROCESSING_CACHE Releasing locks and copying/linking cache files to the session dir.

CACHE_PROCESSED Cache processing completed.

DONE Everything completed successfully.

CANCELLED Cancellation request fulfilled successfully.

CANCELLED_FINISHED Cancellation request fulfilled but [DTR](#) also completed transfer successfully.

ERROR Error occurred.

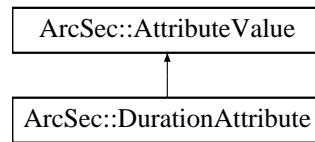
NULL_STATE "Stateless" [DTR](#)

The documentation for this class was generated from the following file:

- DTRStatus.h

10.110 ArcSec::DurationAttribute Class Reference

#include <DateTimeAttribute.h> Inheritance diagram for ArcSec::DurationAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.110.1 Detailed Description

Format: P??Y??M??DT??H??M??S

10.110.2 Member Function Documentation

10.110.2.1 virtual std::string ArcSec::DurationAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.110.2.2 virtual std::string ArcSec::DurationAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.110.2.3 virtual std::string ArcSec::DurationAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- DateTimeAttribute.h

10.111 Arc::Endpoint Class Reference

Represents an endpoint of a service with a given interface type and capabilities.

```
#include <arc/compute/Endpoint.h>
```

Public Types

- enum [CapabilityEnum](#)

Public Member Functions

- [Endpoint](#) (const std::string &URLString="", const std::set< std::string > &Capability=std::set< std::string >(), const std::string &InterfaceName="")
- [Endpoint](#) (const std::string &URLString, const [Endpoint::CapabilityEnum](#) cap, const std::string &InterfaceName="")
- [Endpoint](#) (const [ExecutionTarget](#) &e, const std::string &rsi="")
- [Endpoint](#) (const [ComputingEndpointAttributes](#) &cea, const std::string &rsi="")
- [Endpoint](#) (const [ConfigEndpoint](#) &endpoint)
- bool [HasCapability](#) ([Endpoint::CapabilityEnum](#) cap) const
- bool [HasCapability](#) (const std::string &cap) const
- std::string [str](#) () const
- std::string [getServiceName](#) () const
- bool [operator<](#) (const [Endpoint](#) &other) const
- [Endpoint](#) & [operator=](#) (const [ConfigEndpoint](#) &e)

Static Public Member Functions

- static std::string [GetStringForCapability](#) ([Endpoint::CapabilityEnum](#) cap)

Data Fields

- std::string [URLString](#)
- std::string [InterfaceName](#)
- std::string [HealthState](#)
- std::string [HealthStateInfo](#)
- std::string [QualityLevel](#)
- std::set< std::string > [Capability](#)
- std::string [RequestedSubmissionInterfaceName](#)
- std::string [ServiceID](#)

10.111.1 Detailed Description

Represents an endpoint of a service with a given interface type and capabilities. The type of the interface is described by a string called InterfaceName (from the GLUE2 specification). An [Endpoint](#) object must have a [URL](#), and it is quite useless without capabilities (the system has to know if an [Endpoint](#) is a service registry or a computing element), but the InterfaceName is optional.

The [Endpoint](#) object also contains information about the health state and quality level of the endpoint, and optionally the requested submission interface name, which will be used later if a job will be submitted to a computing element related to this endpoint.

See also:

[CapabilityEnum](#) where the capabilities are listed.

10.111.2 Member Enumeration Documentation

10.111.2.1 enum Arc::Endpoint::CapabilityEnum

The capabilities:

- REGISTRY: service registry capable of returning endpoints
- COMPUTINGINFO: local information system of a computing element capable of returning information about the resource
- JOBLIST: local information system of a computing element capable of returning the list of jobs on the resource
- JOBSUBMIT: interface of a computing element where jobs can be submitted
- JOBCREATION: interface of a computing element where jobs can be created
- UNSPECIFIED: unspecified capability

10.111.3 Constructor & Destructor Documentation

10.111.3.1 Arc::Endpoint::Endpoint (const std::string & *URLString* = "", const std::set< std::string > & *Capability* = std::set<std::string>(), const std::string & *InterfaceName* = "") [inline]

Create a new [Endpoint](#) with a list of capability strings.

Parameters:

- ← *URLString* is a string representing the [URL](#) of the endpoint
- ← *Capability* is a list of capability strings specifying the capabilities of the service
- ← *InterfaceName* is a string specifying the type of the interface of the service

10.111.3.2 Arc::Endpoint::Endpoint (const std::string & *URLString*, const Endpoint::CapabilityEnum *cap*, const std::string & *InterfaceName* = "") [inline]

Create a new [Endpoint](#) with a single capability specified by the [CapabilityEnum](#).

Parameters:

- ← *URLString* is a string representing the [URL](#) of the endpoint
- ← *cap* is a [CapabilityEnum](#) specifying the single capability of the endpoint
- ← *InterfaceName* is an optional string specifying the type of the interface

References [Capability](#), and [GetStringForCapability\(\)](#).

10.111.3.3 Arc::Endpoint::Endpoint (const ExecutionTarget & *e*, const std::string & *rsi* = "")

Create new [Endpoint](#) from [ExecutionTarget](#) object.

Parameters:

e [ExecutionTarget](#) object to create new [Endpoint](#) from.

rsi string specifying the requested submission interface if any. Default value is the empty string.

10.111.3.4 Arc::Endpoint::Endpoint (const ComputingEndpointAttributes & *cea*, const std::string & *rsi* = "")

Create new [Endpoint](#) from [ExecutionTarget](#) object.

Parameters:

cea [ComputingEndpointAttributes](#) object to create new [Endpoint](#) from.

rsi string specifying the requested submission interface if any. Default value is the empty string.

10.111.3.5 Arc::Endpoint::Endpoint (const ConfigEndpoint & *endpoint*) [inline]

Create a new [Endpoint](#) from a [ConfigEndpoint](#). The [URL](#), [InterfaceName](#) and the [RequestedSubmissionInterfaceName](#) will be copied from the [ConfigEndpoint](#), and if the type of the [ConfigEndpoint](#) is [REGISTRY](#) or [COMPUTINGINFO](#), the given capability will be added to the new [Endpoint](#) object.

Parameters:

← *endpoint* is the [ConfigEndpoint](#) object which will be converted to an [Endpoint](#)

This will call [operator=](#).

10.111.4 Member Function Documentation**10.111.4.1 std::string Arc::Endpoint::getServiceName () const**

A string identifying the service exposing this endpoint. It currently extracts the host name from the [URL](#), but this may be refined later.

10.111.4.2 static std::string Arc::Endpoint::GetStringForCapability (Endpoint::CapabilityEnum *cap*) [inline, static]

Get the string representation of the given [CapabilityEnum](#).

Referenced by [Endpoint\(\)](#).

10.111.4.3 bool Arc::Endpoint::HasCapability (const std::string & *cap*) const

Checks if the [Endpoint](#) has the given capability specified by a string

Parameters:

← *cap* is a string specifying a capability

Returns:

true if the [Endpoint](#) has the given capability

10.111.4.4 bool Arc::Endpoint::HasCapability (Endpoint::CapabilityEnum *cap*) const

Checks if the [Endpoint](#) has the given capability specified by a CapabilityEnum

Parameters:

← *cap* is the specified CapabilityEnum

Returns:

true if the [Endpoint](#) has the given capability

10.111.4.5 bool Arc::Endpoint::operator< (const Endpoint & *other*) const

Needed for std::map to be able to sort the keys

10.111.4.6 Endpoint& Arc::Endpoint::operator= (const ConfigEndpoint & *e*)

Copy a [ConfigEndpoint](#) into the [Endpoint](#)

10.111.4.7 std::string Arc::Endpoint::str () const

Returns a string representation of the [Endpoint](#) containing the [URL](#), the main capability and the Interface-Name

10.111.5 Field Documentation**10.111.5.1 std::set<std::string> Arc::Endpoint::Capability**

Set of [GLUE2](#) Capability strings

Referenced by Endpoint().

10.111.5.2 std::string Arc::Endpoint::HealthState

[GLUE2](#) HealthState

10.111.5.3 std::string Arc::Endpoint::HealthStateInfo

[GLUE2](#) HealthStateInfo

10.111.5.4 std::string Arc::Endpoint::InterfaceName

The type of the interface ([GLUE2](#) InterfaceName)

10.111.5.5 std::string Arc::Endpoint::QualityLevel

[GLUE2](#) QualityLevel

10.111.5.6 std::string Arc::Endpoint::RequestedSubmissionInterfaceName

A [GLUE2](#) InterfaceName requesting an InterfaceName used for job submission.

If a user specifies an InterfaceName for submitting jobs, that information will be stored here and will be used when collecting information about the computing element. Only those job submission interfaces will be considered which has this requested InterfaceName.

10.111.5.7 std::string Arc::Endpoint::ServiceID

The ID of the service this [Endpoint](#) belongs to

10.111.5.8 std::string Arc::Endpoint::URLString

The string representation of the [URL](#) of the [Endpoint](#)

The documentation for this class was generated from the following file:

- Endpoint.h

10.112 Arc::EndpointQueryingStatus Class Reference

Represents the status in the [EntityRetriever](#) of the query process of an [Endpoint](#) (service registry, computing element).

```
#include <arc/compute/EndpointQueryingStatus.h>
```

Public Types

- enum [EndpointQueryingStatusType](#) {
[UNKNOWN](#), [SUSPENDED_NOTREQUIRED](#), [STARTED](#), [FAILED](#),
[NOPLUGIN](#), [NOINFORETURNED](#), [SUCCESSFUL](#) }

Public Member Functions

- [EndpointQueryingStatus](#) ([EndpointQueryingStatusType](#) status=[UNKNOWN](#), const std::string &description="")
- bool [operator==](#) ([EndpointQueryingStatusType](#) s) const
- bool [operator==](#) (const [EndpointQueryingStatus](#) &s) const
- bool [operator!=](#) ([EndpointQueryingStatusType](#) s) const
- bool [operator!=](#) (const [EndpointQueryingStatus](#) &s) const
- bool [operator!](#) () const
- [operator bool](#) () const
- [EndpointQueryingStatus](#) & [operator=](#) ([EndpointQueryingStatusType](#) s)
- [EndpointQueryingStatus](#) & [operator=](#) (const [EndpointQueryingStatus](#) &s)
- [EndpointQueryingStatusType](#) [getStatus](#) () const
- const std::string & [getDescription](#) () const
- std::string [str](#) () const

Static Public Member Functions

- static std::string [str](#) ([EndpointQueryingStatusType](#) status)

10.112.1 Detailed Description

Represents the status in the [EntityRetriever](#) of the query process of an [Endpoint](#) (service registry, computing element). An object of this class is returned by the instances of the [EntityRetriever](#) (e.g. [ServiceEndpointRetriever](#), [TargetInformationRetriever](#), [JobListRetriever](#)) representing the state of the process of querying an [Endpoint](#). It contains an [EndpointQueryingStatusType](#) enum ([getStatus](#)), and a description string ([getDescription](#))

10.112.2 Member Enumeration Documentation

10.112.2.1 enum Arc::EndpointQueryingStatus::EndpointQueryingStatusType

The possible states:

Enumerator:

UNKNOWN the state is unknown

SUSPENDED_NOTREQUIRED Querying of the endpoint is suspended since querying it is not required.

STARTED the query process was started

FAILED the query process failed

NOPLUGIN there is no plugin for the given [Endpoint](#) InterfaceName (so the query process was not even started)

NOINFORETURNED query was successful but the response didn't contain entity information

SUCCESSFUL the query process was successful

10.112.3 Constructor & Destructor Documentation

10.112.3.1 `Arc::EndpointQueryingStatus::EndpointQueryingStatus (EndpointQueryingStatusType status = UNKNOWN, const std::string & description = "") [inline]`

A new [EndpointQueryingStatus](#) is created with [UNKNOWN](#) status and with an empty description by default

10.112.4 Member Function Documentation

10.112.4.1 `const std::string& Arc::EndpointQueryingStatus::getDescription () const [inline]`

Return the description string contained within this [EndpointQueryingStatus](#) object

10.112.4.2 `EndpointQueryingStatusType Arc::EndpointQueryingStatus::getStatus () const [inline]`

Return the enum [EndpointQueryingStatusType](#) contained within this [EndpointQueryingStatus](#) object

10.112.4.3 `Arc::EndpointQueryingStatus::operator bool (void) const [inline]`

Returns:

true if the status is successful

References [SUCCESSFUL](#).

10.112.4.4 `bool Arc::EndpointQueryingStatus::operator! (void) const [inline]`

Returns:

true if the status is not successful

References [SUCCESSFUL](#).

10.112.4.5 `bool Arc::EndpointQueryingStatus::operator!= (const EndpointQueryingStatus & s)
const [inline]`

Inequality.

See also:

`operator==(const EndpointQueryingStatus&)`

10.112.4.6 `bool Arc::EndpointQueryingStatus::operator!= (EndpointQueryingStatusType s) const
[inline]`

Inequality.

See also:

`operator==(EndpointQueryingStatusType)`

10.112.4.7 `EndpointQueryingStatus& Arc::EndpointQueryingStatus::operator= (const
EndpointQueryingStatus & s) [inline]`

Copying the [EndpointQueryingStatus](#) object into this one.

Parameters:

← *s* the [EndpointQueryingStatus](#) object whose status and description will be copied into this object

10.112.4.8 `EndpointQueryingStatus& Arc::EndpointQueryingStatus::operator=
(EndpointQueryingStatusType s) [inline]`

Setting the [EndpointQueryingStatus](#) object's state

Parameters:

← *s* the new enum [EndpointQueryingStatusType](#) status

10.112.4.9 `bool Arc::EndpointQueryingStatus::operator== (const EndpointQueryingStatus & s)
const [inline]`

This [EndpointQueryingStatus](#) object equals to another [EndpointQueryingStatus](#) object, if their state equals. The description doesn't matter.

10.112.4.10 `bool Arc::EndpointQueryingStatus::operator== (EndpointQueryingStatusType s)
const [inline]`

This [EndpointQueryingStatus](#) object equals to an enum [EndpointQueryingStatusType](#) if it contains the same state

10.112.4.11 `std::string Arc::EndpointQueryingStatus::str (void) const` `[inline]`

String representation of the [EndpointQueryingStatus](#) object, which is currently simply the string representation of the enum [EndpointQueryingStatusType](#)

References `str()`.

Referenced by `str()`.

10.112.4.12 `static std::string Arc::EndpointQueryingStatus::str (EndpointQueryingStatusType status)` `[static]`

String representation of the states in the enum [EndpointQueryingStatusType](#)

The documentation for this class was generated from the following file:

- `EndpointQueryingStatus.h`

10.113 Arc::EndpointQueryOptions< T > Class Template Reference

Options controlling the query process.

```
#include <EntityRetrieverPlugin.h>
```

Public Member Functions

- [EndpointQueryOptions](#) (const std::set< std::string > &preferredInterfaceNames=std::set< std::string >())

10.113.1 Detailed Description

```
template<typename T> class Arc::EndpointQueryOptions< T >
```

Options controlling the query process.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 `template<typename T> Arc::EndpointQueryOptions< T >::EndpointQueryOptions (const std::set< std::string > & preferredInterfaceNames = std::set<std::string>()) [inline]`

Options for querying [Endpoint](#) objects. When an [Endpoint](#) does not have its interface name specified, all the supported interfaces can be tried. If preferred interface names are provided here, those will be tried first.

Parameters:

← *preferredInterfaceNames* a list of the preferred InterfaceName strings

See also:

[EndpointQueryOptions<Endpoint>](#) the [EntityRetriever<Endpoint>](#) (a.k.a. [ServiceEndpointRetriever](#)) needs different options

The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

10.114 Arc::EndpointQueryOptions< Endpoint > Class Template Reference

The [EntityRetriever<Endpoint>](#) (a.k.a. [ServiceEndpointRetriever](#)) needs different options.

```
#include <EntityRetrieverPlugin.h>
```

Public Member Functions

- [EndpointQueryOptions](#) (bool recursive=false, const std::list< std::string > &capabilityFilter=std::list< std::string >(), const std::list< std::string > &rejectedServices=std::list< std::string >(), const std::set< std::string > &preferredInterfaceNames=std::set< std::string >())

10.114.1 Detailed Description

```
template<> class Arc::EndpointQueryOptions< Endpoint >
```

The [EntityRetriever<Endpoint>](#) (a.k.a. [ServiceEndpointRetriever](#)) needs different options.

10.114.2 Constructor & Destructor Documentation

10.114.2.1 Arc::EndpointQueryOptions< Endpoint >::EndpointQueryOptions (bool recursive = false, const std::list< std::string > &capabilityFilter = std::list<std::string>(), const std::list< std::string > &rejectedServices = std::list<std::string>(), const std::set< std::string > &preferredInterfaceNames = std::set<std::string>()) [inline]

Options for recursivity, filtering of capabilities and rejecting services.

Parameters:

- ← **recursive** Recursive query means that if a service registry is discovered that will be also queried for additional services
- ← **capabilityFilter** Only those services will be discovered which has at least one capability from this list.
- ← **rejectedServices** If a service's [URL](#) contains any item from this list, the services will be not returned among the results.

The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

10.115 Arc::EndpointStatusMap Class Reference

The documentation for this class was generated from the following file:

- Endpoint.h

10.116 Arc::EndpointSubmissionStatus Class Reference

Public Types

- enum [EndpointSubmissionStatusType](#)

Public Member Functions

- [EndpointSubmissionStatus](#) ([EndpointSubmissionStatusType](#) status=UNKNOWN, const std::string &description="")
- bool [operator==](#) ([EndpointSubmissionStatusType](#) s) const
- bool [operator==](#) (const [EndpointSubmissionStatus](#) &s) const
- bool [operator!=](#) ([EndpointSubmissionStatusType](#) s) const
- bool [operator!=](#) (const [EndpointSubmissionStatus](#) &s) const
- bool [operator!](#) () const
- [operator bool](#) () const
- [EndpointSubmissionStatus](#) & [operator=](#) ([EndpointSubmissionStatusType](#) s)
- [EndpointSubmissionStatus](#) & [operator=](#) (const [EndpointSubmissionStatus](#) &s)
- [EndpointSubmissionStatusType](#) [getStatus](#) () const
- const std::string & [getDescription](#) () const
- std::string [str](#) () const

Static Public Member Functions

- static std::string [str](#) ([EndpointSubmissionStatusType](#) status)

10.116.1 Member Enumeration Documentation

10.116.1.1 enum Arc::EndpointSubmissionStatus::EndpointSubmissionStatusType

The possible states:

10.116.2 Constructor & Destructor Documentation

10.116.2.1 Arc::EndpointSubmissionStatus::EndpointSubmissionStatus ([EndpointSubmissionStatusType](#) status = UNKNOWN, const std::string & description = "") [inline]

A new [EndpointSubmissionStatus](#) is created with UNKNOWN status and with an empty description by default

10.116.3 Member Function Documentation

10.116.3.1 const std::string& Arc::EndpointSubmissionStatus::getDescription () const [inline]

Return the description string contained within this [EndpointSubmissionStatus](#) object

10.116.3.2 EndpointSubmissionStatusType Arc::EndpointSubmissionStatus::getStatus () const [inline]

Return the enum [EndpointSubmissionStatusType](#) contained within this [EndpointSubmissionStatus](#) object

10.116.3.3 Arc::EndpointSubmissionStatus::operator bool (void) const [inline]

Returns:

true if the status is successful

10.116.3.4 bool Arc::EndpointSubmissionStatus::operator! (void) const [inline]

Returns:

true if the status is not successful

10.116.3.5 bool Arc::EndpointSubmissionStatus::operator!= (const EndpointSubmissionStatus & s) const [inline]

Inequality.

See also:

operator==(const EndpointSubmissionStatus&)

10.116.3.6 bool Arc::EndpointSubmissionStatus::operator!= (EndpointSubmissionStatusType s) const [inline]

Inequality.

See also:

operator==(EndpointSubmissionStatus)

10.116.3.7 EndpointSubmissionStatus& Arc::EndpointSubmissionStatus::operator= (const EndpointSubmissionStatus & s) [inline]

Copying the [EndpointSubmissionStatus](#) object into this one.

Parameters:

← *s* the [EndpointSubmissionStatus](#) object whose status and description will be copied into this object

10.116.3.8 EndpointSubmissionStatus& Arc::EndpointSubmissionStatus::operator= (EndpointSubmissionStatusType s) [inline]

Setting the [EndpointSubmissionStatus](#) object's state

Parameters:

← *s* the new enum [EndpointSubmissionStatusType](#) status

10.116.3.9 `bool Arc::EndpointSubmissionStatus::operator==(const EndpointSubmissionStatus & s) const [inline]`

This [EndpointSubmissionStatus](#) object equals to another [EndpointQueryingStatus](#) object, if their state equals. The description doesn't matter.

10.116.3.10 `bool Arc::EndpointSubmissionStatus::operator==(EndpointSubmissionStatusType s) const [inline]`

This [EndpointSubmissionStatus](#) object equals to an enum [EndpointSubmissionStatusType](#) if it contains the same state

10.116.3.11 `std::string Arc::EndpointSubmissionStatus::str (void) const [inline]`

String representation of the [EndpointSubmissionStatus](#) object, which is currently simply the string representation of the enum [EndpointSubmissionStatusType](#)

References `str()`.

Referenced by `str()`.

10.116.3.12 `static std::string Arc::EndpointSubmissionStatus::str (EndpointSubmissionStatusType status) [static]`

String representation of the states in the enum [EndpointSubmissionStatusType](#)

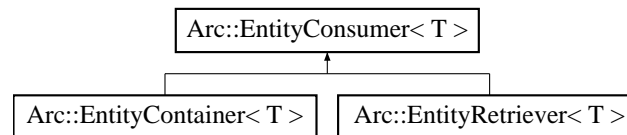
The documentation for this class was generated from the following file:

- Submitter.h

10.117 Arc::EntityConsumer< T > Class Template Reference

A general concept of an object which can consume entities use by the retrievers to return results.

#include <arc/compute/EntityRetriever.h>Inheritance diagram for Arc::EntityConsumer< T >::



Public Member Functions

- virtual void [addEntity](#) (const T &)=0

10.117.1 Detailed Description

template<typename T> class Arc::EntityConsumer< T >

A general concept of an object which can consume entities use by the retrievers to return results. A class which wants to receive results from Retrievers, needs to subclass this class, and implement the [addEntity](#) method.

10.117.2 Member Function Documentation

10.117.2.1 template<typename T> virtual void Arc::EntityConsumer< T >::addEntity (const T &) [pure virtual]

Send an entity to this consumer. This is the method which will be called by the retrievers when a new result is available.

Implemented in [Arc::ExecutionTargetSorter](#), [Arc::ComputingServiceUniq](#), [Arc::ComputingServiceRetriever](#), [Arc::EntityContainer< T >](#), [Arc::EntityRetriever< T >](#), [Arc::JobSupervisor](#), [Arc::EntityContainer< ComputingServiceType >](#), [Arc::EntityRetriever< ComputingServiceType >](#), and [Arc::EntityRetriever< Endpoint >](#).

The documentation for this class was generated from the following file:

- EntityRetriever.h

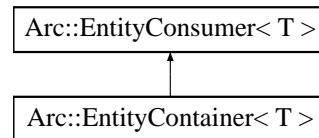
10.118 Arc::EntityContainer< T > Class Template Reference

An entity consumer class storing all the consumed entities in a list.

```
#include <arc/compute/EntityRetriever.h>
Arc::EntityContainer< T >::
```

diagram

for



Public Member Functions

- virtual void [addEntity](#) (const T &t)

10.118.1 Detailed Description

```
template<typename T> class Arc::EntityContainer< T >
```

An entity consumer class storing all the consumed entities in a list. This class is a concrete subclass of the [EntityConsumer](#) abstract class, it also inherits from [std::list](#), and implements the [addEntity](#) method in a way, that it stores all the consumed entities in the list (in itself).

The retrievers return their results through entity consumer objects, so this container object can be used in those places, and then the results can be found in the container, which can be treated as a standard list.

10.118.2 Member Function Documentation

10.118.2.1 `template<typename T> virtual void Arc::EntityContainer< T >::addEntity (const T &t) [inline, virtual]`

All the consumed entities are pushed to the list. Because the [EntityContainer](#) is a standard list, it can push the entities in itself.

Implements [Arc::EntityConsumer< T >](#).

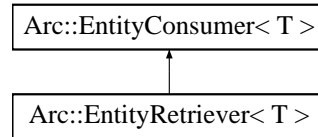
The documentation for this class was generated from the following file:

- EntityRetriever.h

10.119 Arc::EntityRetriever< T > Class Template Reference

Queries [Endpoint](#) objects (using plugins in parallel) and sends the found entities to consumers.

#include <arc/compute/EntityRetriever.h> Inheritance diagram for Arc::EntityRetriever< T >::



Data Structures

- class [Common](#)
- class [Result](#)
- class [ThreadArg](#)

Public Member Functions

- [EntityRetriever](#) (const [UserConfig](#) &, const [EndpointQueryOptions](#)< T > &options=[EndpointQueryOptions](#)< T >())
- void [wait](#) () const
- bool [isDone](#) () const
- void [addConsumer](#) ([EntityConsumer](#)< T > &consumer)
- void [removeConsumer](#) (const [EntityConsumer](#)< T > &consumer)
- [EndpointQueryingStatus](#) [getStatusOfEndpoint](#) (const [Endpoint](#) &endpoint) const
- [EndpointStatusMap](#) [getAllStatuses](#) () const
- bool [setStatusOfEndpoint](#) (const [Endpoint](#) &endpoint, const [EndpointQueryingStatus](#) &status, bool overwrite=true)
- void [getServicesWithStatus](#) (const [EndpointQueryingStatus](#) &status, std::set< std::string > &result)
- void [clearEndpointStatuses](#) ()
- bool [removeEndpoint](#) (const [Endpoint](#) &e)
- virtual void [addEntity](#) (const T &entity)
- virtual void [addEndpoint](#) (const [Endpoint](#) &endpoint)

10.119.1 Detailed Description

template<typename T> class Arc::EntityRetriever< T >

Queries [Endpoint](#) objects (using plugins in parallel) and sends the found entities to consumers. The [EntityRetriever](#) is a template class which queries [Endpoint](#) objects and returns entities of the template type T. The query is done by plugins (capable of retrieving type T objects from [Endpoint](#) objects), and the results are sent to the registered [EntityConsumer](#) objects (capable of consuming type T objects).

When an [Endpoint](#) is added to the [EntityRetriever](#), a new is started which queries the given [Endpoint](#). Each plugin is capable of querying [Endpoint](#) objects with given interfaces (which is indicated with the InterfaceName attribute of the [Endpoint](#)). If the [Endpoint](#) has the InterfaceName specified, then the plugin capable of querying that interface will be selected. If the InterfaceName of the [Endpoint](#) is not specified,

all the available plugins will be considered. If there is a preferred list of interfaces, then first the plugins supporting those interfaces will be tried, and if there are no preferred interfaces, or the preferred ones did not give any result, then all the plugins will be tried. All this happens parallel in separate threads. Currently there are three instance classes:

- the [ServiceEndpointRetriever](#) queries service registries and returns new [Endpoint](#) objects
- the [TargetInformationRetriever](#) queries computing elements and returns [ComputingServiceType](#) objects containing the [BLUE2](#) information about the computing element
- the [JobListRetriever](#) queries computing elements and returns jobs residing on the computing element

To start querying, a new [EntityRetriever](#) needs to be created with the user's credentials in the [UserConfig](#) object, then one or more consumers needs to be added with the [addConsumer](#) method (e.g. an [EntityContainer](#) of the given T type), then the Endpoints need to be added one by one with the [addEndpoint](#) method. Then the [wait](#) method can be called to wait for all the results to arrive, after which we can be sure that all the retrieved entities are passed to the registered consumer objects. If we registered an [EntityContainer](#), then we can get all the results from the container, using it as a standard list.

It is possible to specify options in the constructor, which in case of the [TargetInformationRetriever](#) and the [JobListRetriever](#) classes is an [EndpointQueryOptions](#) object containing a list of preferred InterfaceNames. When an [Endpoint](#) has not InterfaceName specified, these preferred InterfaceNames will be tried first. The [ServiceEndpointRetriever](#) has different options though: the [EndpointQueryOptions<Endpoint>](#) object does not contain a preferred list of InterfaceNames. It has a flag for recursivity instead and string lists for filtering services by capability and rejecting them by [URL](#).

See also:

[ComputingServiceRetriever](#) which combines the [ServiceEndpointRetriever](#) and the [TargetInformationRetriever](#) to query both the service registries and the computing elements

[ServiceEndpointRetriever](#) example:

```
Arc::UserConfig uc;
// create the retriever with no options
Arc::ServiceEndpointRetriever retriever(uc);
// create a container which will store the results
Arc::EntityContainer<Arc::Endpoint> container;
// add the container to the retriever
retriever.addConsumer(container);
// create an endpoint which will be queried
Arc::Endpoint registry("test.nordugrid.org", Arc::Endpoint::REGISTRY);
// start querying the endpoint
retriever.addEndpoint(registry);
// wait for the querying process to finish
retriever.wait();
// get the status of the query
Arc::EndpointQueryingStatus status = retriever.getStatusOfEndpoint(registry);
```

After [wait](#) returns, container contains all the services found in the registry "test.nordugrid.org".

[TargetInformationRetriever](#) example:

```
Arc::UserConfig uc;
// create the retriever with no options
Arc::TargetInformationRetriever retriever(uc);
// create a container which will store the results
Arc::EntityContainer<Arc::ComputingServiceType> container;
// add the container to the retriever
retriever.addConsumer(container);
```

```
// create an endpoint which will be queried
Arc::Endpoint ce("test.nordugrid.org", Arc::Endpoint::COMPUTINGINFO);
// start querying the endpoint
retriever.addEndpoint(ce);
// wait for the querying process to finish
retriever.wait();
// get the status of the query
Arc::EndpointQueryingStatus status = retriever.getStatusOfEndpoint(ce);
```

After `wait` returns, container contains the `ComputingServiceType` object which has the full GLUE2 information about the computing element.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 `template<typename T> Arc::EntityRetriever< T >::EntityRetriever
(const UserConfig &, const EndpointQueryOptions< T > & options =
EndpointQueryOptions< T >())`

Needs the credentials of the user and can have some options. Creating the `EntityRetriever` does not start any querying yet.

Parameters:

`UserConfig` with the user's credentials
`options` contain type T specific querying options

10.119.3 Member Function Documentation

10.119.3.1 `template<typename T> void Arc::EntityRetriever< T >::addConsumer
(EntityConsumer< T > & consumer) [inline]`

Register a new consumer which will receive results from now on.

Parameters:

← `consumer` is a consumer object capable of consuming type T objects

Referenced by `Arc::ComputingServiceRetriever::addConsumer()`.

10.119.3.2 `template<typename T> virtual void Arc::EntityRetriever< T >::addEndpoint (const
Endpoint & endpoint) [virtual]`

Starts querying an `Endpoint`. This method is used to start querying an `Endpoint`. It starts the query process in a separate thread, and returns immediately.

Parameters:

← `endpoint` is the `Endpoint` to query

10.119.3.3 `template<typename T> virtual void Arc::EntityRetriever< T >::addEntity (const T &
entity) [virtual]`

This method should only be used by the plugins when they return their results. This will send the results to all the registered consumers.

In the case of the [ServiceEndpointRetriever](#), the retrieved entities are actually [Endpoint](#) objects, and the [ServiceEndpointRetriever](#) does more work here depending on the options set in [EndpointQueryOptions<Endpoint>](#):

- if the [URL](#) of a retrieved [Endpoint](#) is on the rejected list, the [Endpoint](#) is not sent to the consumers
- if recursivity is turned on, and the retrieved [Endpoint](#) is a service registry, then it is sent to the [addEntity](#) method for querying
- if the retrieved [Endpoint](#) does not have at least one of the capabilities provided in the capability filter, then the [Endpoint](#) is not sent to the consumers

Parameters:

← *entity* is the type T object retrieved from the endpoints

Implements [Arc::EntityConsumer< T >](#).

10.119.3.4 `template<typename T> void Arc::EntityRetriever< T >::clearEndpointStatuses () [inline]`

Clear statuses of registered endpoints. The status map of registered endpoints will be cleared when calling this method. That can be useful if an already registered endpoint need to be queried again.

10.119.3.5 `template<typename T> EndpointStatusMap Arc::EntityRetriever< T >::getAllStatuses () const [inline]`

Get status of all the queried [Endpoint](#) objects. This method returns a copy of the internal status map, and thus is only a snapshot. If you want the final status map, make sure to invoke the [EntityRetriever::wait](#) method before this one.

Returns:

a map with [Endpoint](#) objects as keys and status objects as values.

Referenced by [Arc::ComputingServiceRetriever::getAllStatuses\(\)](#).

10.119.3.6 `template<typename T> void Arc::EntityRetriever< T >::getServicesWithStatus (const EndpointQueryingStatus & status, std::set< std::string > & result)`

Insert into *results* the endpoint.ServiceName() of each endpoint with the given status.

Parameters:

← *status* is the status of the desired endpoints

↔ *result* is a set into which the matching endpoint service names are inserted

10.119.3.7 `template<typename T> EndpointQueryingStatus Arc::EntityRetriever< T >::getStatusOfEndpoint (const Endpoint & endpoint) const`

Get the status of the query process of a given [Endpoint](#).

Parameters:

← *endpoint* is the [Endpoint](#) whose status we want to know.

Returns:

an [EndpointQueryingStatus](#) object containing the status of the query

10.119.3.8 template<typename T> bool Arc::EntityRetriever< T >::isDone () const [inline]

Check if the query is finished.

Returns:

true if the query is finished, all the results were delivered to the consumers.

10.119.3.9 template<typename T> void Arc::EntityRetriever< T >::removeConsumer (const EntityConsumer< T > & *consumer*)

Remove a previously registered consumer

Parameters:

← *consumer* is the consumer object

Referenced by Arc::ComputingServiceRetriever::removeConsumer().

10.119.3.10 template<typename T> bool Arc::EntityRetriever< T >::removeEndpoint (const Endpoint & *e*) [inline]

Remove a particular registered endpoint. The specified endpoint will be removed from the status map of registered endpoints.

Parameters:

e endpoint to remove from status map.

Returns:

true is returned if endpoint is found in the map, otherwise false is returned.

10.119.3.11 template<typename T> bool Arc::EntityRetriever< T >::setStatusOfEndpoint (const Endpoint & *endpoint*, const EndpointQueryingStatus & *status*, bool *overwrite* = true)

Set the status of the query process of a given [Endpoint](#). This method should only be used by the plugins when they finished querying an [Endpoint](#).

Parameters:

← *endpoint* is the [Endpoint](#) whose status we want to set

← *status* is the [EndpointQueryStatus](#) object containing the status

← *overwrite* indicates if a previous status should be overwritten, if not, then in case of an existing status the method returns false

Returns:

true if the new status was set, false if it was not set (e.g. because overwrite was false, and the status was already set previously)

**10.119.3.12 template<typename T> void Arc::EntityRetriever< T >::wait (void) const
[inline]**

This method blocks until all the results arrive.

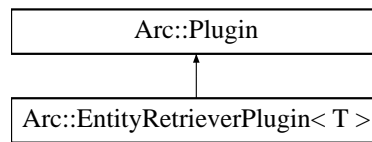
Referenced by Arc::ComputingServiceRetriever::wait().

The documentation for this class was generated from the following file:

- EntityRetriever.h

10.120 Arc::EntityRetrieverPlugin< T > Class Template Reference

Inheritance diagram for Arc::EntityRetrieverPlugin< T >::



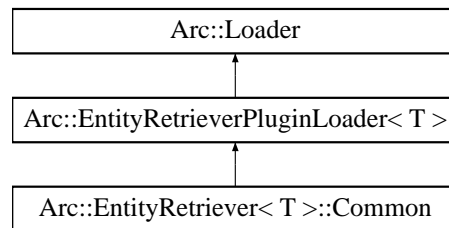
```
template<typename T> class Arc::EntityRetrieverPlugin< T >
```

The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

10.121 Arc::EntityRetrieverPluginLoader< T > Class Template Reference

Inheritance diagram for Arc::EntityRetrieverPluginLoader< T >::



```
template<typename T> class Arc::EntityRetrieverPluginLoader< T >
```

The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

10.122 Arc::EnvLockWrapper Class Reference

Class to provide automatic locking/unlocking of environment on creation/destruction.

```
#include <arc/Utils.h>
```

Public Member Functions

- [EnvLockWrapper](#) (bool all=false)
- [~EnvLockWrapper](#) (void)

10.122.1 Detailed Description

Class to provide automatic locking/unlocking of environment on creation/destruction. For use with external libraries using unprotected setenv/getenv in a multi-threaded environment.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 Arc::EnvLockWrapper::EnvLockWrapper (bool *all* = false) [inline]

Create a new environment lock for using setenv/getenv.

Parameters:

all set to true for setenv and false for getenv.

References [Arc::EnvLockWrap\(\)](#).

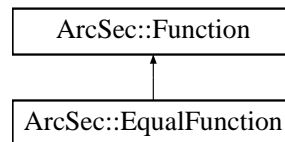
The documentation for this class was generated from the following file:

- [Utils.h](#)

10.123 ArcSec::EqualFunction Class Reference

Evaluate whether the two values are equal.

`#include <EqualFunction.h>` Inheritance diagram for ArcSec::EqualFunction::



Public Member Functions

- virtual [AttributeValue](#) * [evaluate](#) ([AttributeValue](#) *arg0, [AttributeValue](#) *arg1, bool check_id=true)
- virtual std::list< [AttributeValue](#) * > [evaluate](#) (std::list< [AttributeValue](#) * > args, bool check_id=true)

Static Public Member Functions

- static std::string [getFunctionName](#) (std::string datatype)

10.123.1 Detailed Description

Evaluate whether the two values are equal.

10.123.2 Member Function Documentation

10.123.2.1 virtual std::list< [AttributeValue](#)* > ArcSec::EqualFunction::evaluate (std::list< [AttributeValue](#) * > args, bool check_id = true) [virtual]

Evaluate a list of [AttributeValue](#) objects, and return a list of Attribute objects

Implements [ArcSec::Function](#).

10.123.2.2 virtual [AttributeValue](#)* ArcSec::EqualFunction::evaluate ([AttributeValue](#) * arg0, [AttributeValue](#) * arg1, bool check_id = true) [virtual]

Evaluate two [AttributeValue](#) objects, and return one [AttributeValue](#) object

Implements [ArcSec::Function](#).

10.123.2.3 static std::string ArcSec::EqualFunction::getFunctionName (std::string datatype) [static]

help function to get the FunctionName

The documentation for this class was generated from the following file:

- EqualFunction.h

10.124 ArcSec::EvalResult Struct Reference

Struct to record the xml node and effect, which will be used by [Evaluator](#) to get the information about which rule/policy(in xmlnode) is satisfied.

```
#include <Result.h>
```

10.124.1 Detailed Description

Struct to record the xml node and effect, which will be used by [Evaluator](#) to get the information about which rule/policy(in xmlnode) is satisfied.

The documentation for this struct was generated from the following file:

- Result.h

10.125 ArcSec::EvaluationCtx Class Reference

[EvaluationCtx](#), in charge of storing some context information for.

```
#include <EvaluationCtx.h>
```

Public Member Functions

- [EvaluationCtx](#) ([Request](#) *request)

10.125.1 Detailed Description

[EvaluationCtx](#), in charge of storing some context information for.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 ArcSec::EvaluationCtx::EvaluationCtx (Request * *request*) [inline]

Construct a new [EvaluationCtx](#) based on the given request

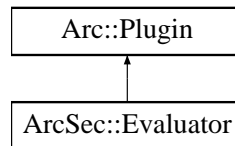
The documentation for this class was generated from the following file:

- [EvaluationCtx.h](#)

10.126 ArcSec::Evaluator Class Reference

Interface for policy evaluation. Execute the policy evaluation, based on the request and policy.

#include <Evaluator.h> Inheritance diagram for ArcSec::Evaluator::



Public Member Functions

- virtual [Response](#) * [evaluate](#) ([Request](#) *request)=0
- virtual [Response](#) * [evaluate](#) (const [Source](#) &request)=0
- virtual [Response](#) * [evaluate](#) ([Request](#) *request, const [Source](#) &policy)=0
- virtual [Response](#) * [evaluate](#) (const [Source](#) &request, const [Source](#) &policy)=0
- virtual [Response](#) * [evaluate](#) ([Request](#) *request, [Policy](#) *policyobj)=0
- virtual [Response](#) * [evaluate](#) (const [Source](#) &request, [Policy](#) *policyobj)=0
- virtual [AttributeFactory](#) * [getAttrFactory](#) ()=0
- virtual [FnFactory](#) * [getFnFactory](#) ()=0
- virtual [AlgFactory](#) * [getAlgFactory](#) ()=0
- virtual void [addPolicy](#) (const [Source](#) &policy, const std::string &id="")=0
- virtual void [addPolicy](#) ([Policy](#) *policy, const std::string &id="")=0
- virtual void [setCombiningAlg](#) ([EvaluatorCombiningAlg](#) alg)=0
- virtual void [setCombiningAlg](#) ([CombiningAlg](#) *alg=NULL)=0
- virtual const char * [getName](#) (void) const =0

Protected Member Functions

- virtual [Response](#) * [evaluate](#) ([EvaluationCtx](#) *ctx)=0

10.126.1 Detailed Description

Interface for policy evaluation. Execute the policy evaluation, based on the request and policy.

10.126.2 Member Function Documentation

10.126.2.1 virtual void ArcSec::Evaluator::addPolicy ([Policy](#) * *policy*, const std::string & *id* = "") [pure virtual]

Add policy to the evaluator. [Policy](#) will be marked with id. The policy object is taken over by this instance and will be destroyed in destructor.

10.126.2.2 virtual void ArcSec::Evaluator::addPolicy (const [Source](#) & *policy*, const std::string & *id* = "") [pure virtual]

Add policy from specified source to the evaluator. [Policy](#) will be marked with id.

10.126.2.3 **virtual Response* ArcSec::Evaluator::evaluate (EvaluationCtx * *ctx*) [protected, pure virtual]**

Evaluate the request by using the [EvaluationCtx](#) object (which includes the information about request). The ctx is destroyed inside this method (why?!?!?).

10.126.2.4 **virtual Response* ArcSec::Evaluator::evaluate (const Source & *request*, Policy * *policyobj*) [pure virtual]**

Evaluate the request from specified source against the specified policy. In some implementations all of the existing policie inside the evaluator may be destroyed by this method.

10.126.2.5 **virtual Response* ArcSec::Evaluator::evaluate (Request * *request*, Policy * *policyobj*) [pure virtual]**

Evaluate the specified request against the specified policy. In some implementations all of the existing policy inside the evaluator may be destroyed by this method.

10.126.2.6 **virtual Response* ArcSec::Evaluator::evaluate (const Source & *request*, const Source & *policy*) [pure virtual]**

Evaluate the request from specified source against the policy from specified source. In some implementations all of the existing policie inside the evaluator may be destroyed by this method.

10.126.2.7 **virtual Response* ArcSec::Evaluator::evaluate (Request * *request*, const Source & *policy*) [pure virtual]**

Evaluate the specified request against the policy from specified source. In some implementations all of the existing policies inside the evaluator may be destroyed by this method.

10.126.2.8 **virtual Response* ArcSec::Evaluator::evaluate (const Source & *request*) [pure virtual]**

Evaluates the request by using a specified source

10.126.2.9 **virtual Response* ArcSec::Evaluator::evaluate (Request * *request*) [pure virtual]**

Evaluates the request by using a [Request](#) object. Evaluation is done till at least one of policies is satisfied.

10.126.2.10 **virtual AlgFactory* ArcSec::Evaluator::getAlgFactory () [pure virtual]**

Get the [AlgFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator AlgFactory *().

10.126.2.11 virtual AttributeFactory* ArcSec::Evaluator::getAttrFactory () [pure virtual]

Get the [AttributeFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator AttributeFactory *().

10.126.2.12 virtual FnFactory* ArcSec::Evaluator::getFnFactory () [pure virtual]

Get the [FnFactory](#) object

Referenced by ArcSec::EvaluatorContext::operator FnFactory *().

10.126.2.13 virtual const char* ArcSec::Evaluator::getName (void) const [pure virtual]

Get the name of this evaluator

10.126.2.14 virtual void ArcSec::Evaluator::setCombiningAlg (CombiningAlg * alg = NULL) [pure virtual]

Specifies loadable combining algorithms. In case of multiple policies their results will be combined using this algorithm. To switch to simple algorithm specify NULL argument.

10.126.2.15 virtual void ArcSec::Evaluator::setCombiningAlg (EvaluatorCombiningAlg alg) [pure virtual]

Specifies one of simple combining algorithms. In case of multiple policies their results will be combined using this algorithm.

The documentation for this class was generated from the following file:

- Evaluator.h

10.127 ArcSec::EvaluatorContext Class Reference

Context for evaluator. It includes the factories which will be used to create related objects.

```
#include <Evaluator.h>
```

Public Member Functions

- [operator AttributeFactory * \(\)](#)
- [operator FnFactory * \(\)](#)
- [operator AlgFactory * \(\)](#)

10.127.1 Detailed Description

Context for evaluator. It includes the factories which will be used to create related objects.

10.127.2 Member Function Documentation

10.127.2.1 ArcSec::EvaluatorContext::operator AlgFactory * () [inline]

Returns associated [AlgFactory](#) object

References [ArcSec::Evaluator::getAlgFactory\(\)](#).

10.127.2.2 ArcSec::EvaluatorContext::operator AttributeFactory * () [inline]

Returns associated [AttributeFactory](#) object

References [ArcSec::Evaluator::getAttrFactory\(\)](#).

10.127.2.3 ArcSec::EvaluatorContext::operator FnFactory * () [inline]

Returns associated [FnFactory](#) object

References [ArcSec::Evaluator::getFnFactory\(\)](#).

The documentation for this class was generated from the following file:

- [Evaluator.h](#)

10.128 ArcSec::EvaluatorLoader Class Reference

[EvaluatorLoader](#) is implemented as a helper class for loading different [Evaluator](#) objects, like ArcEvaluator.

```
#include <EvaluatorLoader.h>
```

Public Member Functions

- [Evaluator](#) * [getEvaluator](#) (const std::string &classname)
- [Evaluator](#) * [getEvaluator](#) (const [Policy](#) *policy)
- [Evaluator](#) * [getEvaluator](#) (const [Request](#) *request)
- [Request](#) * [getRequest](#) (const std::string &classname, const [Source](#) &requestsource)
- [Request](#) * [getRequest](#) (const [Source](#) &requestsource)
- [Policy](#) * [getPolicy](#) (const std::string &classname, const [Source](#) &polycysource)
- [Policy](#) * [getPolicy](#) (const [Source](#) &polycysource)

10.128.1 Detailed Description

[EvaluatorLoader](#) is implemented as a helper class for loading different [Evaluator](#) objects, like ArcEvaluator. The object loading is based on the configuration information about evaluator, including information for factory class, request, policy and evaluator itself

10.128.2 Member Function Documentation

10.128.2.1 [Evaluator](#)* ArcSec::EvaluatorLoader::getEvaluator (const [Request](#) * *request*)

Get evaluator object suitable for presented request

10.128.2.2 [Evaluator](#)* ArcSec::EvaluatorLoader::getEvaluator (const [Policy](#) * *policy*)

Get evaluator object suitable for presented policy

10.128.2.3 [Evaluator](#)* ArcSec::EvaluatorLoader::getEvaluator (const std::string & *classname*)

Get evaluator object according to the class name

10.128.2.4 [Policy](#)* ArcSec::EvaluatorLoader::getPolicy (const [Source](#) & *polycysource*)

Get proper policy object according to the policy source

10.128.2.5 [Policy](#)* ArcSec::EvaluatorLoader::getPolicy (const std::string & *classname*, const [Source](#) & *polycysource*)

Get policy object according to the class name, based on the policy source

10.128.2.6 [Request](#)* ArcSec::EvaluatorLoader::getRequest (const [Source](#) & *requestsource*)

Get request object according to the request source

10.128.2.7 Request* ArcSec::EvaluatorLoader::getRequest (const std::string & *classname*, const Source & *requestsource*)

Get request object according to the class name, based on the request source

The documentation for this class was generated from the following file:

- EvaluatorLoader.h

10.129 Arc::ExecutableType Class Reference

Executable.

```
#include <arc/compute/JobDescription.h>
```

Data Fields

- `std::string` [Path](#)
- `std::list< std::string >` [Argument](#)
- `std::pair< bool, int >` [SuccessExitCode](#)

10.129.1 Detailed Description

Executable. The [ExecutableType](#) class is used to specify path to an executable, arguments to pass to it when invoked and the exit code for successful execution.

Note:

The Name string member has been renamed to Path.

10.129.2 Field Documentation

10.129.2.1 `std::list<std::string>` [Arc::ExecutableType::Argument](#)

List of arguments to executable. The Argument list is used to specify arguments which should be passed to the executable upon invocation.

10.129.2.2 `std::string` [Arc::ExecutableType::Path](#)

Path to executable. The Path string should specify the path to an executable. Note that some implementations might only accept a relative path, while others might also accept a absolute one.

10.129.2.3 `std::pair<bool, int>` [Arc::ExecutableType::SuccessExitCode](#)

Exit code at successful execution. The SuccessExitCode pair is used to specify the exit code returned by the executable in case of successful execution. For some scenarios the exit code returned by the executable should be ignored, which is specified by setting the first member of this object to false. If the exit code should be used for validation at the execution service, the first member of pair must be set to true, while the second member should be the exit code returned at successful execution.

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.130 Arc::ExecutionEnvironmentAttributes Class Reference

Data Fields

- [Software OperatingSystem](#)

10.130.1 Field Documentation

10.130.1.1 Software Arc::ExecutionEnvironmentAttributes::OperatingSystem

OperatingSystem. The OperatingSystem member is not present in [GLUE2](#) but contains the three [GLUE2](#) attributes OSFamily, OSName and OSVersion.

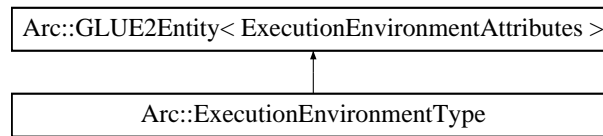
- OSFamily OSFamily_t 1 * The general family to which the Execution Environment operating * system belongs.
- OSName OSName_t 0..1 * The specific name of the operating sytem
- OSVersion String 0..1 * The version of the operating system, as defined by the vendor.

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.131 Arc::ExecutionEnvironmentType Class Reference

Inheritance diagram for Arc::ExecutionEnvironmentType::



The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.132 Arc::ExecutionTarget Class Reference

[ExecutionTarget](#).

```
#include <arc/compute/ExecutionTarget.h>
```

Public Member Functions

- [ExecutionTarget](#) ()
- [ExecutionTarget](#) (const [ExecutionTarget](#) &t)
- [ExecutionTarget](#) (long int addrptr)
- void [RegisterJobSubmission](#) (const [JobDescription](#) &jobdesc) const

Friends

- std::ostream & [operator<<](#) (std::ostream &, const [ExecutionTarget](#) &)

10.132.1 Detailed Description

[ExecutionTarget](#). This class describe a target which accept computing jobs. All of the members contained in this class, with a few exceptions, are directly linked to attributes defined in the GLUE Specification v. 2.0 (GFD-R-P.147).

10.132.2 Constructor & Destructor Documentation

10.132.2.1 Arc::ExecutionTarget::ExecutionTarget () [inline]

Create an [ExecutionTarget](#). Default constructor to create an [ExecutionTarget](#). Takes no arguments.

10.132.2.2 Arc::ExecutionTarget::ExecutionTarget (const ExecutionTarget & t) [inline]

Create an [ExecutionTarget](#). Copy constructor.

Parameters:

target [ExecutionTarget](#) to copy.

10.132.2.3 Arc::ExecutionTarget::ExecutionTarget (long int addrptr) [inline]

Create an [ExecutionTarget](#). Copy constructor? Needed from Python?

Parameters:

addrptr

10.132.3 Member Function Documentation

10.132.3.1 `void Arc::ExecutionTarget::RegisterJobSubmission (const JobDescription & jobdesc) const`

Update [ExecutionTarget](#) after succesful job submission. Method to update the [ExecutionTarget](#) after a job succesfully has been submitted to the computing resource it represents. E.g. if a job is sent to the computing resource and is expected to enter the queue, then the WaitingJobs attribute is incremented with 1.

Parameters:

jobdesc contains all information about the job submitted.

10.132.4 Friends And Related Function Documentation

10.132.4.1 `std::ostream& operator<< (std::ostream &, const ExecutionTarget &) [friend]`

Print the [ExecutionTarget](#) information. Method to print the [ExecutionTarget](#) attributes to a std::ostream object.

Parameters:

out is the std::ostream to print the attributes to.

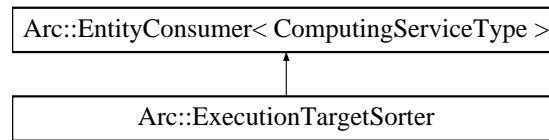
@return the input ostream object is returned.

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.133 Arc::ExecutionTargetSorter Class Reference

Inheritance diagram for Arc::ExecutionTargetSorter::



Public Member Functions

- void [addEntity](#) (const [ComputingServiceType](#) &cs)

10.133.1 Member Function Documentation

10.133.1.1 void Arc::ExecutionTargetSorter::addEntity (const ComputingServiceType &) [virtual]

Send an entity to this consumer. This is the method which will be called by the retrievers when a new result is available.

Implements [Arc::EntityConsumer< ComputingServiceType >](#).

The documentation for this class was generated from the following file:

- Broker.h

10.134 Arc::ExpirationReminder Class Reference

A class intended for internal use within counters.

```
#include <arc/Counter.h>
```

Public Member Functions

- `bool operator< (const ExpirationReminder &other) const`
- `Glib::TimeVal getExpiryTime () const`
- `Counter::IDType getReservationID () const`

Friends

- class [Counter](#)

10.134.1 Detailed Description

A class intended for internal use within counters. This class is used for "reminder objects" that are used for automatic deallocation of self-expiring reservations.

10.134.2 Member Function Documentation

10.134.2.1 Glib::TimeVal Arc::ExpirationReminder::getExpiryTime () const

Returns the expiry time. This method returns the expiry time of the reservation that this [ExpirationReminder](#) is associated with.

Returns:

The expiry time.

10.134.2.2 Counter::IDType Arc::ExpirationReminder::getReservationID () const

Returns the identification number of the reservation. This method returns the identification number of the self-expiring reservation that this [ExpirationReminder](#) is associated with.

Returns:

The identification number.

10.134.2.3 bool Arc::ExpirationReminder::operator< (const [ExpirationReminder](#) & other) const

Less than operator, compares "soonness". This is the less than operator for the [ExpirationReminder](#) class. It compares the priority of such objects with respect to which reservation expires first. It is used when reminder objects are inserted in a priority queue in order to allways place the next reservation to expire at the top.

The documentation for this class was generated from the following file:

- `Counter.h`

10.135 Arc::FileAccess Class Reference

Defines interface for accessing filesystems.

```
#include <arc/FileAccess.h>
```

Data Structures

- struct [header_t](#)

Internal struct used for communication between processes.

Public Member Functions

- [FileAccess](#) (void)
- [~FileAccess](#) (void)
- bool [ping](#) (void)
- bool [fa_setuid](#) (int uid, int gid)
- bool [fa_mkdir](#) (const std::string &path, mode_t mode)
- bool [fa_mkdirp](#) (const std::string &path, mode_t mode)
- bool [fa_link](#) (const std::string &oldpath, const std::string &newpath)
- bool [fa_softlink](#) (const std::string &oldpath, const std::string &newpath)
- bool [fa_copy](#) (const std::string &oldpath, const std::string &newpath, mode_t mode)
- bool [fa_rename](#) (const std::string &oldpath, const std::string &newpath)
- bool [fa_chmod](#) (const std::string &path, mode_t mode)
- bool [fa_stat](#) (const std::string &path, struct stat &st)
- bool [fa_lstat](#) (const std::string &path, struct stat &st)
- bool [fa_fstat](#) (struct stat &st)
- bool [fa_ftruncate](#) (off_t length)
- off_t [fa_fallocate](#) (off_t length)
- bool [fa_readlink](#) (const std::string &path, std::string &linkpath)
- bool [fa_remove](#) (const std::string &path)
- bool [fa_unlink](#) (const std::string &path)
- bool [fa_rmdir](#) (const std::string &path)
- bool [fa_rmdirr](#) (const std::string &path)
- bool [fa_opendir](#) (const std::string &path)
- bool [fa_closedir](#) (void)
- bool [fa_readdir](#) (std::string &name)
- bool [fa_open](#) (const std::string &path, int flags, mode_t mode)
- bool [fa_close](#) (void)
- bool [fa_mkstemp](#) (std::string &path, mode_t mode)
- off_t [fa_lseek](#) (off_t offset, int whence)
- ssize_t [fa_read](#) (void *buf, size_t size)
- ssize_t [fa_write](#) (const void *buf, size_t size)
- ssize_t [fa_pread](#) (void *buf, size_t size, off_t offset)
- ssize_t [fa_pwrite](#) (const void *buf, size_t size, off_t offset)
- int [geterrno](#) ()
- [operator bool](#) (void)
- bool [operator!](#) (void)

Static Public Member Functions

- static [FileAccess](#) * [Acquire](#) (void)
- static void [Release](#) ([FileAccess](#) *fa)
- static void [testtune](#) (void)

10.135.1 Detailed Description

Defines interface for accessing filesystems. This class accesses the local filesystem through a proxy executable which allows switching user id in multithreaded systems without introducing conflict with other threads. Its methods are mostly replicas of corresponding POSIX functions with some convenience tweaking.

10.135.2 Member Function Documentation

10.135.2.1 `bool Arc::FileAccess::fa_copy (const std::string & oldpath, const std::string & newpath, mode_t mode)`

Copy file to new location. If new file is created it is assigned specified mode.

10.135.2.2 `bool Arc::FileAccess::fa_mkdirp (const std::string & path, mode_t mode)`

Make a directory and assign it specified mode. If missing all intermediate directories are created too.

10.135.2.3 `bool Arc::FileAccess::fa_mkstemp (std::string & path, mode_t mode)`

Open new temporary file for writing. On input path contains template of file name ending with XXXXXX. On output path is path to created file.

10.135.2.4 `bool Arc::FileAccess::fa_setuid (int uid, int gid)`

Modify user uid and gid. If any is set to 0 then executable is switched to original uid/gid.

The documentation for this class was generated from the following file:

- `FileAccess.h`

10.136 Arc::FileAccessContainer Class Reference

Container for shared [FileAccess](#) objects.

```
#include <arc/FileAccess.h>
```

Public Member Functions

- [FileAccessContainer](#) (unsigned int minval, unsigned int maxval)
- [FileAccessContainer](#) (void)
- [~FileAccessContainer](#) (void)
- [FileAccess](#) * [Acquire](#) (void)
- void [Release](#) ([FileAccess](#) *fa)
- void [SetMin](#) (unsigned int val)
- void [SetMax](#) (unsigned int val)

10.136.1 Detailed Description

Container for shared [FileAccess](#) objects. [FileAccessContainer](#) maintains a pool of executables and can be used to reduce the overhead in creating and destroying executables when using [FileAccess](#).

10.136.2 Member Function Documentation

10.136.2.1 [FileAccess](#)* Arc::FileAccessContainer::Acquire (void)

Get object from container. Object either is taken from stored ones or new one created. Acquired object loses its connection to container and can be safely destroyed or returned into other container.

10.136.2.2 void Arc::FileAccessContainer::Release ([FileAccess](#) *fa)

Returns object into container. It can be any object - taken from another container or created using new.

The documentation for this class was generated from the following file:

- [FileAccess.h](#)

10.137 Arc::FileLock Class Reference

A general file locking class.

```
#include <arc/FileLock.h>
```

Public Member Functions

- [FileLock](#) (const std::string &filename, unsigned int timeout=DEFAULT_LOCK_TIMEOUT, bool use_pid=true)
- bool [acquire](#) (bool &lock_removed)
- bool [acquire](#) ()
- bool [release](#) (bool force=false)
- int [check](#) (bool log_error=true)

Static Public Member Functions

- static std::string [getLockSuffix](#) ()

Static Public Attributes

- static const int [DEFAULT_LOCK_TIMEOUT](#)
- static const std::string [LOCK_SUFFIX](#)

10.137.1 Detailed Description

A general file locking class. This class can be used when protected access is required to files which are used by multiple processes or threads. Call [acquire\(\)](#) to obtain a lock and [release\(\)](#) to release it when finished. [check\(\)](#) can be used to verify if a lock is valid for the current process. Locks are independent of [FileLock](#) objects - locks are only created and destroyed through [acquire\(\)](#) and [release\(\)](#), not on creation or destruction of [FileLock](#) objects.

Unless use_pid is set false, the process ID and hostname of the calling process are stored in a file filename.lock in the form pid@hostname. This information is used to determine whether a lock is still valid. It is also possible to specify a timeout on the lock.

To ensure an atomic locking operation, [acquire\(\)](#) first creates a temporary lock file filename.lock.XXXXXXX, then attempts to rename this file to filename.lock. After a successful rename the lock file is checked to make sure the correct process ID and hostname are inside. This eliminates race conditions where multiple processes compete to obtain the lock.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 Arc::FileLock::FileLock (const std::string & filename, unsigned int timeout = DEFAULT_LOCK_TIMEOUT, bool use_pid = true)

Create a new [FileLock](#) object.

Parameters:

filename The name of the file to be locked

timeout The timeout of the lock

use_pid If true, use process id in the lock and to determine lock validity

10.137.3 Member Function Documentation

10.137.3.1 bool Arc::FileLock::acquire ()

Acquire the lock. Callers can use this version of [acquire\(\)](#) if they do not care whether an invalid lock was removed in the process of obtaining the lock.

Returns:

True if lock is successfully acquired

10.137.3.2 bool Arc::FileLock::acquire (bool & *lock_removed*)

Acquire the lock. Returns true if the lock was acquired successfully. Locks are acquired if no lock file currently exists, or if the current lock file is invalid. A lock is invalid if the process ID inside the lock no longer exists on the host inside the lock, or the age of the lock file is greater than the lock timeout.

Parameters:

lock_removed Set to true if an existing lock was removed due to being invalid. In this case the caller may decide to check or delete the file as it is potentially corrupted.

Returns:

True if lock is successfully acquired

10.137.3.3 int Arc::FileLock::check (bool *log_error* = `true`)

Check the lock is valid.

Parameters:

log_error may be set to false to log error messages at INFO level, in cases where the lock not existing or being owned by another host are not errors.

Returns:

0 if the lock is valid for the current process, the pid inside the lock if the lock is owned by another process on the same host, or -1 if the lock is owned by another host or any other error occurred.

10.137.3.4 bool Arc::FileLock::release (bool *force* = `false`)

Release the lock.

Parameters:

force Remove the lock without checking ownership or timeout

Returns:

True if lock is successfully released

The documentation for this class was generated from the following file:

- FileLock.h

10.138 Arc::FinderLoader Class Reference

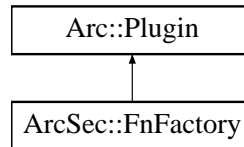
The documentation for this class was generated from the following file:

- FinderLoader.h

10.139 ArcSec::FnFactory Class Reference

Interface for function factory class.

#include <FnFactory.h>Inheritance diagram for ArcSec::FnFactory::



Public Member Functions

- virtual [Function](#) * [createFn](#) (const std::string &type)=0

10.139.1 Detailed Description

Interface for function factory class. [FnFactory](#) is in charge of creating [Function](#) object according to the algorithm type given as argument of method [createFn](#). This class can be inherited for implementing a factory class which can create some specific [Function](#) objects.

10.139.2 Member Function Documentation

10.139.2.1 virtual [Function](#)* ArcSec::FnFactory::createFn (const std::string & type) [pure virtual]

creat algorithm object based on the type algorithm type

Parameters:

type The type of [Function](#)

Returns:

The object of [Function](#)

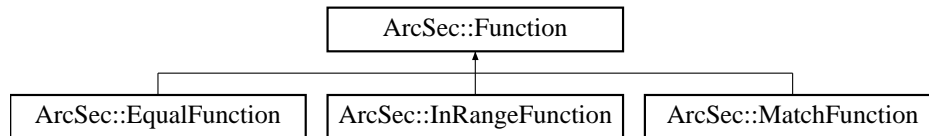
The documentation for this class was generated from the following file:

- FnFactory.h

10.140 ArcSec::Function Class Reference

Interface for function, which is in charge of evaluating two [AttributeValue](#).

#include <Function.h> Inheritance diagram for ArcSec::Function::



Public Member Functions

- virtual [AttributeValue](#) * **evaluate** ([AttributeValue](#) *arg0, [AttributeValue](#) *arg1, bool check_id=true)=0
- virtual std::list< [AttributeValue](#) * > **evaluate** (std::list< [AttributeValue](#) * > args, bool check_id=true)=0

10.140.1 Detailed Description

Interface for function, which is in charge of evaluating two [AttributeValue](#).

10.140.2 Member Function Documentation

10.140.2.1 virtual std::list< [AttributeValue](#) * > ArcSec::Function::evaluate (std::list< [AttributeValue](#) * > args, bool check_id = true) [pure virtual]

Evaluate a list of [AttributeValue](#) objects, and return a list of Attribute objects

Implemented in [ArcSec::EqualFunction](#), [ArcSec::InRangeFunction](#), and [ArcSec::MatchFunction](#).

10.140.2.2 virtual [AttributeValue](#)* ArcSec::Function::evaluate ([AttributeValue](#) * arg0, [AttributeValue](#) * arg1, bool check_id = true) [pure virtual]

Evaluate two [AttributeValue](#) objects, and return one [AttributeValue](#) object

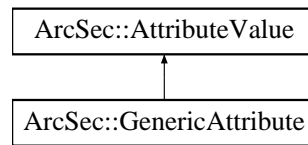
Implemented in [ArcSec::EqualFunction](#), [ArcSec::InRangeFunction](#), and [ArcSec::MatchFunction](#).

The documentation for this class was generated from the following file:

- Function.h

10.141 ArcSec::GenericAttribute Class Reference

Inheritance diagram for ArcSec::GenericAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.141.1 Member Function Documentation

10.141.1.1 virtual std::string ArcSec::GenericAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.141.1.2 virtual std::string ArcSec::GenericAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.141.1.3 virtual std::string ArcSec::GenericAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- GenericAttribute.h

10.142 Arc::GlobusResult Class Reference

The documentation for this class was generated from the following file:

- GlobusErrorUtils.h

10.143 Arc::GLUE2 Class Reference

GLUE2 parser.

```
#include <GLUE2.h>
```

Static Public Member Functions

- static void [ParseExecutionTargets](#) (XMLNode glue2tree, std::list< [ComputingServiceType](#) > &targets)

10.143.1 Detailed Description

GLUE2 parser. This class parses GLUE2 information rendered in XML and transfers information into various classes representing different types of objects which GLUE2 information model can describe. This parser uses GLUE Specification v. 2.0 (GFD-R-P.147).

10.143.2 Member Function Documentation

10.143.2.1 static void Arc::GLUE2::ParseExecutionTargets (XMLNode *glue2tree*, std::list< [ComputingServiceType](#) > & *targets*) [static]

Parses ComputingService elements of GLUE2 into [ComputingServiceType](#) objects. The glue2tree is either XML tree representing ComputingService object directly or ComputingService objects are immediate children of it. On exit targets contains [ComputingServiceType](#) objects found inside glue2tree. If targets contained any objects on entry those are not destroyed.

Parameters:

glue2tree

targets

The documentation for this class was generated from the following file:

- GLUE2.h

10.144 Arc::GLUE2Entity< T > Class Template Reference

```
template<typename T> class Arc::GLUE2Entity< T >
```

The documentation for this class was generated from the following file:

- [GLUE2Entity.h](#)

10.145 Arc::GSSCredential Class Reference

```
#include <GSSCredential.h>
```

10.145.1 Detailed Description

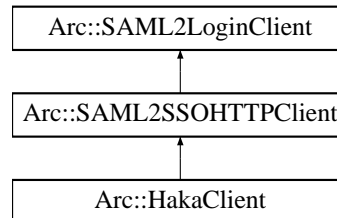
Class for converting credentials stored in file in PEM format into Globus structire. It works only for full credentials containing private key. This limitation is due to limited API of Globus.

The documentation for this class was generated from the following file:

- GSSCredential.h

10.146 Arc::HakaClient Class Reference

Inheritance diagram for Arc::HakaClient::



Protected Member Functions

- [MCC_Status processIdPLogin](#) (const std::string username, const std::string password)
- [MCC_Status processConsent](#) ()
- [MCC_Status processIdP2Confusa](#) ()

10.146.1 Member Function Documentation

10.146.1.1 MCC_Status Arc::HakaClient::processConsent () [protected, virtual]

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implements [Arc::SAML2SSOHTTPClient](#).

10.146.1.2 MCC_Status Arc::HakaClient::processIdP2Confusa () [protected, virtual]

Redirects the user back from identity provider to the Confusa SP

Implements [Arc::SAML2SSOHTTPClient](#).

10.146.1.3 MCC_Status Arc::HakaClient::processIdPLogin (const std::string *username*, const std::string *password*) [protected, virtual]

Actual identity provider parsers for next three methods implemented in subdirectory idp/

Parse identity provider login page and submit username and password in the previsioned way

Implements [Arc::SAML2SSOHTTPClient](#).

The documentation for this class was generated from the following file:

- HakaClient.h

10.147 Arc::FileAccess::header_t Struct Reference

Internal struct used for communication between processes.

```
#include <FileAccess.h>
```

10.147.1 Detailed Description

Internal struct used for communication between processes.

The documentation for this struct was generated from the following file:

- FileAccess.h

10.148 Arc::HTTPClientInfo Struct Reference

Data Fields

- std::string [reason](#)
- uint64_t [size](#)
- [Time](#) lastModified
- std::string [type](#)
- std::list< std::string > [cookies](#)
- std::string [location](#)

The documentation for this struct was generated from the following file:

- ClientInterface.h

10.149 Arc::InfoCache Class Reference

Stores XML document in filesystem split into parts.

```
#include <InfoCache.h>
```

Public Member Functions

- [InfoCache](#) (const [Config](#) &cfg, const std::string &service_id)

10.149.1 Detailed Description

Stores XML document in filesystem split into parts.

10.149.2 Constructor & Destructor Documentation

10.149.2.1 Arc::InfoCache::InfoCache (const Config & cfg, const std::string & service_id)

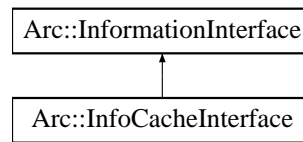
Creates object according to configuration (see InfoCacheConfig.xsd). XML configuration is passed in cfg. Argument service_id is used to distinguish between various documents stored under same path - corresponding files will be stored in subdirectory with service_id name.

The documentation for this class was generated from the following file:

- InfoCache.h

10.150 Arc::InfoCacheInterface Class Reference

Inheritance diagram for Arc::InfoCacheInterface::



Protected Member Functions

- virtual void [Get](#) (const std::list< std::string > &path, [XMLNodeContainer](#) &result)

10.150.1 Member Function Documentation

10.150.1.1 virtual void Arc::InfoCacheInterface::Get (const std::list< std::string > & path, XMLNodeContainer & result) [protected, virtual]

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented from [Arc::InformationInterface](#).

The documentation for this class was generated from the following file:

- InfoCache.h

10.151 Arc::InfoFilter Class Reference

Filters information document according to identity of requestor.

```
#include <InfoFilter.h>
```

Public Member Functions

- [InfoFilter](#) ([MessageAuth](#) &id)
- [Filter](#) ([XMLNode](#) doc) const
- [Filter](#) ([XMLNode](#) doc, const [InfoFilterPolicies](#) &policies, const [NS](#) &ns) const

10.151.1 Detailed Description

Filters information document according to identity of requestor. Identity is compared to policies stored inside information document and external ones. Parts of document which do not pass policy evaluation are removed.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 Arc::InfoFilter::InfoFilter (MessageAuth & id)

Creates object and associates identity. Associated identity is not copied, hence passed argument must not be destroyed while this method is used.

10.151.3 Member Function Documentation

10.151.3.1 bool Arc::InfoFilter::Filter (XMLNode doc, const InfoFilterPolicies & policies, const NS & ns) const

Filter information document according to internal and external policies. In provided document all policies and nodes which have their policies evaluated to negative result are removed. External policies are provided in policies argument. First element of every pair is XPath defining to which XML node policy must be applied. Second element is policy itself. Argument ns defines XML namespaces for XPath evaluation.

10.151.3.2 bool Arc::InfoFilter::Filter (XMLNode doc) const

Filter information document according to internal policies. In provided document all policies and nodes which have their policies evaluated to negative result are removed.

The documentation for this class was generated from the following file:

- InfoFilter.h

10.152 Arc::InfoRegister Class Reference

Registration to Information Indexing [Service](#).

```
#include <InfoRegister.h>
```

10.152.1 Detailed Description

Registration to Information Indexing [Service](#). This class represents service registering to Information Indexing [Service](#). It does not perform registration itself. It only collects configuration information. Configuration is as described in InfoRegisterConfig.xsd for element InfoRegistration.

The documentation for this class was generated from the following file:

- InfoRegister.h

10.153 Arc::InfoRegisterContainer Class Reference

```
#include <InfoRegister.h>
```

Public Member Functions

- [InfoRegistrar](#) * [addRegistrar](#) ([XMLNode](#) doc)
- void [addService](#) ([InfoRegister](#) *reg, const std::list< std::string > &ids, [XMLNode](#) cfg=[XMLNode\(\)](#))
- void [removeService](#) ([InfoRegister](#) *reg)

10.153.1 Detailed Description

Singleton class for scanning configuration and storing references to registration elements.

10.153.2 Member Function Documentation

10.153.2.1 [InfoRegistrar](#)* [Arc::InfoRegisterContainer::addRegistrar](#) ([XMLNode](#) doc)

Adds ISISes to list of handled services. Supplied configuration document is scanned for [InfoRegistrar](#) elements and those are turned into [InfoRegistrar](#) classes for handling connection to ISIS service each.

10.153.2.2 void [Arc::InfoRegisterContainer::addService](#) ([InfoRegister](#) * *reg*, const std::list< std::string > & *ids*, [XMLNode](#) *cfg* = [XMLNode\(\)](#))

Adds service to list of handled. This method must be called first time after last [addRegistrar](#) was called - services will be only associated with ISISes which are already added. Argument *ids* contains list of ISIS identifiers to which service is associated. If *ids* is empty then service is associated to all ISISes currently added. If argument *cfg* is available and no ISISes are configured then [addRegistrars](#) is called with *cfg* used as configuration document.

10.153.2.3 void [Arc::InfoRegisterContainer::removeService](#) ([InfoRegister](#) * *reg*)

This method must be called if service being destroyed.

The documentation for this class was generated from the following file:

- [InfoRegister.h](#)

10.154 Arc::InfoRegisters Class Reference

Handling registrations to multiple Information Indexing Services.

```
#include <InfoRegister.h>
```

Public Member Functions

- [InfoRegisters](#) ([XMLNode](#) cfg, [Service](#) *service)
- bool [addRegister](#) ([XMLNode](#) cfg, [Service](#) *service)

10.154.1 Detailed Description

Handling registrations to multiple Information Indexing Services.

10.154.2 Constructor & Destructor Documentation

10.154.2.1 Arc::InfoRegisters::InfoRegisters ([XMLNode](#) *cfg*, [Service](#) * *service*)

Constructor creates [InfoRegister](#) objects according to configuration. Inside *cfg* elements [InfoRegister](#) elements are found and for each corresponding [InfoRegister](#) object is created. Those objects are destroyed in destructor of this class.

The documentation for this class was generated from the following file:

- [InfoRegister.h](#)

10.155 Arc::InfoRegistrar Class Reference

Registration process associated with particular ISIS.

```
#include <InfoRegister.h>
```

Public Member Functions

- void [registration](#) (void)
- bool [addService](#) ([InfoRegister *](#), [XMLNode](#))
- bool [removeService](#) ([InfoRegister *](#))

10.155.1 Detailed Description

Registration process associated with particular ISIS. Instance of this class starts thread which takes care passing information about associated services to ISIS service defined in configuration. Configuration is as described in InfoRegister.xsd for element [InfoRegistrar](#).

10.155.2 Member Function Documentation

10.155.2.1 bool Arc::InfoRegistrar::addService (InfoRegister *, XMLNode)

Adds new service to list of handled services. [Service](#) is described by it's [InfoRegister](#) object which must be valid as long as this object is functional.

10.155.2.2 void Arc::InfoRegistrar::registration (void)

Performs registartion in a loop. Never exits unless there is a critical error or requested by destructor. Must be called only once.

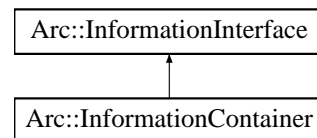
The documentation for this class was generated from the following file:

- InfoRegister.h

10.156 Arc::InformationContainer Class Reference

Information System document container and processor.

#include <InformationInterface.h> Inheritance diagram for Arc::InformationContainer::



Public Member Functions

- [InformationContainer](#) ([XMLNode](#) doc, bool copy=false)
- [XMLNode Acquire](#) (void)
- void [Assign](#) ([XMLNode](#) doc, bool copy=false)

Protected Member Functions

- virtual void [Get](#) (const std::list< std::string > &path, [XMLNodeContainer](#) &result)

Protected Attributes

- [XMLNode doc_](#)

10.156.1 Detailed Description

Information System document container and processor. This class inherits from [InformationInterface](#) and offers container for storing informational XML document.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 Arc::InformationContainer::InformationContainer ([XMLNode](#) doc, bool copy = false)

Creates an instance with XML document . If is true this method makes a copy of for internal use.

10.156.3 Member Function Documentation

10.156.3.1 XMLNode Arc::InformationContainer::Acquire (void)

Get a lock on contained XML document. To be used in multi-threaded environment. Do not forget to release it with Release()

10.156.3.2 void Arc::InformationContainer::Assign ([XMLNode](#) doc, bool copy = false)

Replaces internal XML document with . If is true this method makes a copy of for internal use.

10.156.3.3 `virtual void Arc::InformationContainer::Get (const std::list< std::string > & path, XMLNodeContainer & result)` `[protected, virtual]`

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented from [Arc::InformationInterface](#).

10.156.4 Field Documentation**10.156.4.1** `XMLNode Arc::InformationContainer::doc_` `[protected]`

Either link or container of XML document

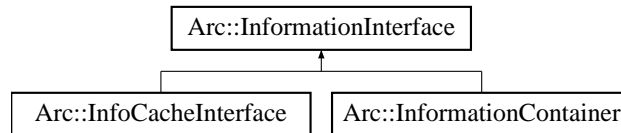
The documentation for this class was generated from the following file:

- InformationInterface.h

10.157 Arc::InformationInterface Class Reference

Information System message processor.

#include <InformationInterface.h> Inheritance diagram for Arc::InformationInterface::



Public Member Functions

- [InformationInterface](#) (bool safe=true)

Protected Member Functions

- virtual void [Get](#) (const std::list< std::string > &path, [XMLNodeContainer](#) &result)

Protected Attributes

- Glib::Mutex [lock_](#)

10.157.1 Detailed Description

Information System message processor. This class provides callback for 2 operations of WS-ResourceProperties and convenient parsing/generation of corresponding SOAP messages. In a future it may extend range of supported specifications.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 Arc::InformationInterface::InformationInterface (bool *safe* = true)

Constructor. If 'safe' is true all calls to Get will be locked.

10.157.3 Member Function Documentation

10.157.3.1 virtual void Arc::InformationInterface::Get (const std::list< std::string > & *path*, [XMLNodeContainer](#) & *result*) [protected, virtual]

This method is called by this object's Process method. Real implementation of this class should return (sub)tree of XML document. This method may be called multiple times per single Process call. Here is a set on XML element names specifying how to reach requested node(s).

Reimplemented in [Arc::InfoCacheInterface](#), and [Arc::InformationContainer](#).

10.157.4 Field Documentation

10.157.4.1 Glib::Mutex Arc::InformationInterface::lock_ [protected]

Mutex used to protect access to Get methods in multi-threaded env.

The documentation for this class was generated from the following file:

- InformationInterface.h

10.158 Arc::InformationRequest Class Reference

Request for information in InfoSystem.

```
#include <InformationInterface.h>
```

Public Member Functions

- [InformationRequest](#) (void)
- [InformationRequest](#) (const std::list< std::string > &path)
- [InformationRequest](#) (const std::list< std::list< std::string > > &paths)
- [InformationRequest](#) (XMLNode query)
- SOAPEnvelope * [SOAP](#) (void)

10.158.1 Detailed Description

Request for information in InfoSystem. This is a convenience wrapper creating proper WS-ResourceProperties request targeted InfoSystem interface of service.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 Arc::InformationRequest::InformationRequest (void)

Dummy constructor

10.158.2.2 Arc::InformationRequest::InformationRequest (const std::list< std::string > &path)

Request for attribute specified by elements of path. Currently only first element is used.

10.158.2.3 Arc::InformationRequest::InformationRequest (const std::list< std::list< std::string > > &paths)

Request for attribute specified by elements of paths. Currently only first element of every path is used.

10.158.2.4 Arc::InformationRequest::InformationRequest (XMLNode query)

Request for attributes specified by XPath query.

10.158.3 Member Function Documentation

10.158.3.1 SOAPEnvelope* Arc::InformationRequest::SOAP (void)

Returns generated SOAP message

The documentation for this class was generated from the following file:

- InformationInterface.h

10.159 Arc::InformationResponse Class Reference

Informational response from InfoSystem.

```
#include <InformationInterface.h>
```

Public Member Functions

- [InformationResponse](#) (SOAPEnvelope &soap)
- std::list< [XMLNode](#) > [Result](#) (void)

10.159.1 Detailed Description

Informational response from InfoSystem. This is a convenience wrapper analyzing WS-ResourceProperties response from InfoSystem interface of service.

10.159.2 Constructor & Destructor Documentation

10.159.2.1 Arc::InformationResponse::InformationResponse (SOAPEnvelope & soap)

Constructor parses WS-ResourceProperties response. Provided SOAPEnvelope object must be valid as long as this object is in use.

10.159.3 Member Function Documentation

10.159.3.1 std::list<XMLNode> Arc::InformationResponse::Result (void)

Returns set of attributes which were in SOAP message passed to constructor.

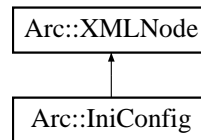
The documentation for this class was generated from the following file:

- InformationInterface.h

10.160 Arc::IniConfig Class Reference

Class representing "ini-style" configuration.

`#include <arc/IniConfig.h>`Inheritance diagram for Arc::IniConfig::



Public Member Functions

- [IniConfig](#) ()
- [IniConfig](#) (const std::string &filename)
- bool [Evaluate](#) ([Config](#) &cfg)

10.160.1 Detailed Description

Class representing "ini-style" configuration. It provides a way to convert configuration to XML for use with HED internally.

See also:

[Profile](#)

The documentation for this class was generated from the following file:

- IniConfig.h

10.161 Arc::initializeCredentialsType Class Reference

Defines how user credentials are looked for.

```
#include <arc/UserConfig.h>
```

Public Types

- enum [initializeType](#) {
 [SkipCredentials](#), [NotTryCredentials](#), [TryCredentials](#), [RequireCredentials](#),
 [SkipCANotTryCredentials](#), [SkipCATryCredentials](#), [SkipCAResquireCredentials](#) }

Public Member Functions

- [initializeCredentialsType](#) (void)
- [initializeCredentialsType](#) ([initializeType](#) v)
- bool [operator==](#) ([initializeType](#) v)
- bool [operator!=](#) ([initializeType](#) v)
- [operator initializeType](#) (void)

10.161.1 Detailed Description

Defines how user credentials are looked for. For complete information see description of [UserConfig::InitializeCredentials\(initializeCredentialsType\)](#) method.

10.161.2 Member Enumeration Documentation

10.161.2.1 enum Arc::initializeCredentialsType::initializeType

[initializeType](#) determines how [UserConfig](#) deals with credentials.

Enumerator:

SkipCredentials Don't look for credentials.

NotTryCredentials Look for credentials but don't evaluate them.

TryCredentials Look for credentials and test if they are valid.

RequireCredentials Look for credentials, test if they are valid and report errors if not valid.

SkipCANotTryCredentials Same as NotTryCredentials but skip checking CA certificates.

SkipCATryCredentials Same as TryCredentials but skip checking CA certificates.

SkipCAResquireCredentials Same as RequireCredentials but skip checking CA certificates.

The documentation for this class was generated from the following file:

- UserConfig.h

10.162 Arc::InputFileType Class Reference

Data Fields

- std::string [Checksum](#)

10.162.1 Field Documentation

10.162.1.1 std::string Arc::InputFileType::Checksum

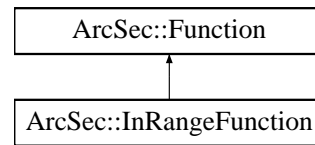
CRC32 checksum of file. The Checksum attribute specifies the textual representation of CRC32 checksum of file in base 10.

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.163 ArcSec::InRangeFunction Class Reference

Inheritance diagram for ArcSec::InRangeFunction::



Public Member Functions

- virtual [AttributeValue](#) * [evaluate](#) ([AttributeValue](#) *arg0, [AttributeValue](#) *arg1, bool check_id=true)
- virtual std::list< [AttributeValue](#) * > [evaluate](#) (std::list< [AttributeValue](#) * > args, bool check_id=true)

10.163.1 Member Function Documentation

10.163.1.1 virtual std::list< [AttributeValue](#)* > ArcSec::InRangeFunction::evaluate (std::list< [AttributeValue](#) * > args, bool check_id = true) [virtual]

Evaluate a list of [AttributeValue](#) objects, and return a list of Attribute objects

Implements [ArcSec::Function](#).

10.163.1.2 virtual [AttributeValue](#)* ArcSec::InRangeFunction::evaluate ([AttributeValue](#) * arg0, [AttributeValue](#) * arg1, bool check_id = true) [virtual]

Evaluate two [AttributeValue](#) objects, and return one [AttributeValue](#) object

Implements [ArcSec::Function](#).

The documentation for this class was generated from the following file:

- InRangeFunction.h

10.164 Arc::InterruptGuard Class Reference

Marks off a section of code which should not be interrupted by signals.

```
#include <arc/Utils.h>
```

10.164.1 Detailed Description

Marks off a section of code which should not be interrupted by signals.

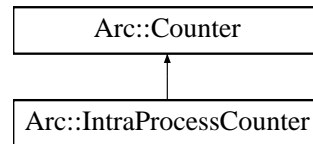
The documentation for this class was generated from the following file:

- Utils.h

10.165 Arc::IntraProcessCounter Class Reference

A class for counters used by threads within a single process.

#include <IntraProcessCounter.h> Inheritance diagram for Arc::IntraProcessCounter::



Public Member Functions

- [IntraProcessCounter](#) (int limit, int excess)
- virtual [~IntraProcessCounter](#) ()
- virtual int [getLimit](#) ()
- virtual int [setLimit](#) (int newLimit)
- virtual int [changeLimit](#) (int amount)
- virtual int [getExcess](#) ()
- virtual int [setExcess](#) (int newExcess)
- virtual int [changeExcess](#) (int amount)
- virtual int [getValue](#) ()
- virtual [CounterTicket reserve](#) (int amount=1, Glib::TimeVal duration=[ETERNAL](#), bool prioritized=false, Glib::TimeVal timeOut=[ETERNAL](#))

Protected Member Functions

- virtual void [cancel](#) (IDType reservationID)
- virtual void [extend](#) (IDType &reservationID, Glib::TimeVal &expiryTime, Glib::TimeVal duration=[ETERNAL](#))

10.165.1 Detailed Description

A class for counters used by threads within a single process. This is a class for shared among different threads within a single process. See the [Counter](#) class for further information about counters and examples of usage.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 Arc::IntraProcessCounter::IntraProcessCounter (int *limit*, int *excess*)

Creates an [IntraProcessCounter](#) with specified limit and excess. This constructor creates a counter with the specified limit (amount of resources available for reservation) and excess limit (an extra amount of resources that may be used for prioritized reservations).

Parameters:

- limit* The limit of the counter.
- excess* The excess limit of the counter.

10.165.2.2 virtual Arc::IntraProcessCounter::~~IntraProcessCounter () [virtual]

Destructor. This is the destructor of the [IntraProcessCounter](#) class. Does not need to do anything.

10.165.3 Member Function Documentation**10.165.3.1 virtual void Arc::IntraProcessCounter::cancel (IDType *reservationID*) [protected, virtual]**

Cancellation of a reservation. This method cancels a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters:

reservationID The identity number (key) of the reservation to cancel.

Implements [Arc::Counter](#).

10.165.3.2 virtual int Arc::IntraProcessCounter::changeExcess (int *amount*) [virtual]

Changes the excess limit of the counter. Changes the excess limit of the counter by adding a certain amount to the current excess limit.

Parameters:

amount The amount by which to change the excess limit.

Returns:

The new excess limit.

Implements [Arc::Counter](#).

10.165.3.3 virtual int Arc::IntraProcessCounter::changeLimit (int *amount*) [virtual]

Changes the limit of the counter. Changes the limit of the counter by adding a certain amount to the current limit.

Parameters:

amount The amount by which to change the limit.

Returns:

The new limit.

Implements [Arc::Counter](#).

10.165.3.4 virtual void Arc::IntraProcessCounter::extend (IDType & *reservationID*, Glib::TimeVal & *expiryTime*, Glib::TimeVal *duration* = ETERNAL) [protected, virtual]

Extension of a reservation. This method extends a reservation. It is called by the [CounterTicket](#) that corresponds to the reservation.

Parameters:

reservationID Used for input as well as output. Contains the identification number of the original reservation on entry and the new identification number of the extended reservation on exit.

expiryTime Used for input as well as output. Contains the expiry time of the original reservation on entry and the new expiry time of the extended reservation on exit.

duration The time by which to extend the reservation. The new expiration time is computed based on the current time, NOT the previous expiration time.

Implements [Arc::Counter](#).

10.165.3.5 virtual int Arc::IntraProcessCounter::getExcess () [virtual]

Returns the excess limit of the counter. Returns the excess limit of the counter, i.e. by how much the usual limit may be exceeded by prioritized reservations.

Returns:

The excess limit.

Implements [Arc::Counter](#).

10.165.3.6 virtual int Arc::IntraProcessCounter::getLimit () [virtual]

Returns the current limit of the counter. This method returns the current limit of the counter, i.e. how many units can be reserved simultaneously by different threads without claiming high priority.

Returns:

The current limit of the counter.

Implements [Arc::Counter](#).

10.165.3.7 virtual int Arc::IntraProcessCounter::getValue () [virtual]

Returns the current value of the counter. Returns the current value of the counter, i.e. the number of unreserved units. Initially, the value is equal to the limit of the counter. When a reservation is made, the the value is decreased. Normally, the value should never be negative, but this may happen if there are prioritized reservations. It can also happen if the limit is decreased after some reservations have been made, since reservations are never revoked.

Returns:

The current value of the counter.

Implements [Arc::Counter](#).

10.165.3.8 virtual CounterTicket Arc::IntraProcessCounter::reserve (int amount = 1, Glib::TimeVal duration = ETERNAL, bool prioritized = false, Glib::TimeVal timeOut = ETERNAL) [virtual]

Makes a reservation from the counter. This method makes a reservation from the counter. If the current value of the counter is too low to allow for the reservation, the method blocks until the reservation is possible or times out.

Parameters:

- amount* The amount to reserve, default value is 1.
- duration* The duration of a self expiring reservation, default is that it lasts forever.
- prioritized* Whether this reservation is prioritized and thus allowed to use the excess limit.
- timeOut* The maximum time to block if the value of the counter is too low, default is to allow "eternal" blocking.

Returns:

A [CounterTicket](#) that can be queried about the status of the reservation as well as for cancellations and extensions.

Implements [Arc::Counter](#).

10.165.3.9 virtual int Arc::IntraProcessCounter::setExcess (int *newExcess*) [virtual]

Sets the excess limit of the counter. This method sets a new excess limit for the counter.

Parameters:

newExcess The new excess limit, an absolute number.

Returns:

The new excess limit.

Implements [Arc::Counter](#).

10.165.3.10 virtual int Arc::IntraProcessCounter::setLimit (int *newLimit*) [virtual]

Sets the limit of the counter. This method sets a new limit for the counter.

Parameters:

newLimit The new limit, an absolute number.

Returns:

The new limit.

Implements [Arc::Counter](#).

The documentation for this class was generated from the following file:

- IntraProcessCounter.h

10.166 Arc::ISIS_description Struct Reference

The documentation for this struct was generated from the following file:

- InfoRegister.h

10.167 Arc::IString Class Reference

Class used for localised output of log messages.

```
#include <arc/IString.h>
```

10.167.1 Detailed Description

Class used for localised output of log messages. [IString](#) should only be used directly in rare cases. [Logger](#) should be used instead in most cases.

The documentation for this class was generated from the following file:

- IString.h

10.168 **Arc::JobDescriptionParserPluginLoader::iterator** **Class** **Reference**

The documentation for this class was generated from the following file:

- [JobDescriptionParserPlugin.h](#)

10.169 Arc::Job Class Reference

[Job](#).

```
#include <arc/compute/Job.h>
```

Public Member Functions

- [Job](#) ()
- void [SaveToStream](#) (std::ostream &out, bool longlist) const
- [Job](#) & [operator=](#) ([XMLNode](#) job)
- void [SetFromXML](#) ([XMLNode](#) job)
- void [ToXML](#) ([XMLNode](#) job) const

Static Public Member Functions

- static bool [ReadJobIDsFromFile](#) (const std::string &filename, std::list< std::string > &jobids, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobIDToFile](#) (const std::string &jobid, const std::string &filename, unsigned nTries=10, unsigned tryInterval=500000)
- static bool [WriteJobIDsToFile](#) (const std::list< std::string > &jobids, const std::string &filename, unsigned nTries=10, unsigned tryInterval=500000)

Data Fields

- std::string [JobDescription](#)
- std::string [JobDescriptionDocument](#)

10.169.1 Detailed Description

[Job](#). This class describe a Grid job. The class contains public accessible member attributes and methods for dealing with a Grid job. Most of the member attributes contained in this class are directly linked to the ComputingActivity defined in the GLUE Specification v. 2.0 (GFD-R-P.147).

10.169.2 Constructor & Destructor Documentation

10.169.2.1 Arc::Job::Job ()

Create a [Job](#) object. Default constructor. Takes no arguments.

10.169.3 Member Function Documentation

10.169.3.1 Job& Arc::Job::operator= ([XMLNode](#) *job*)

Set [Job](#) attributes from a [XMLNode](#). The attributes of the [Job](#) object is set to the values specified in the [XMLNode](#). The [XMLNode](#) should be a ComputingActivity type using the [GLUE2](#) XML hierarchical rendering, see <http://forge.gridforum.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/GLUE2XMLSchema> for more information. Note that associations are not parsed.

Parameters:

job is a [XMLNode](#) of [GLUE2](#) ComputingActivity type.

See also:

[ToXML](#)

10.169.3.2 `static bool Arc::Job::ReadJobIDsFromFile (const std::string & filename, std::list< std::string > & jobids, unsigned nTries = 10, unsigned tryInterval = 500000) [static]`

Read a list of [Job](#) IDs from a file, and append them to a list. This static method will read job IDs from the given file, and append the strings to the string list given as parameter. File locking will be done as described for the `ReadAllJobsFromFile` method. It returns false if the file was not readable, true otherwise, even if there were no IDs in the file. The lines of the file will be trimmed, and lines starting with # will be ignored.

Parameters:

filename is the filename of the jobidfile

jobids is a list of strings, to which the IDs read from the file will be appended

nTries specifies the maximal number of times the method will try to acquire a lock on file to read.

tryInterval specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns:

true in case of success, otherwise false.

10.169.3.3 `void Arc::Job::SaveToStream (std::ostream & out, bool longlist) const`

Write job information to a `std::ostream` object. This method will write job information to the passed `std::ostream` object. The `longlist` boolean specifies whether more (true) or less (false) information should be printed.

Parameters:

out is the `std::ostream` object to print the attributes to.

longlist is a boolean for switching on long listing (more details).

10.169.3.4 `void Arc::Job::SetFromXML (XMLNode job)`

Set [Job](#) attributes from a [XMLNode](#) representing [GLUE2](#) ComputingActivity. Because job XML representation follows [GLUE2](#) model this method is similar to `operator=(XMLNode)`. But it only covers job attributes which are part of [GLUE2](#) computing activity. Also it treats [Job](#) object as being iextended with information provided by [XMLNode](#). Contrary `operator=(XMLNode)` fully reinitializes [Job](#), hence removing any associations to other objects.

10.169.3.5 void Arc::Job::ToXML (XMLNode *job*) const

Add job information to a [XMLNode](#). Child nodes of GLUE ComputingActivity type containing job information of this object will be added to the passed [XMLNode](#).

Parameters:

job is the [XMLNode](#) to add job information to in form of [GLUE2](#) ComputingActivity type child nodes.

See also:

operator=

10.169.3.6 static bool Arc::Job::WriteJobIDsToFile (const std::list< std::string > &*jobids*, const std::string &*filename*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000) [static]

Append list of URLs to a file. This static method will put the ID given as a string, and append it to the given file. File locking will be done as described for the ReadAllJobsFromFile method. It returns false if the file was not writable, true otherwise.

Parameters:

jobid is a list of [URL](#) objects to be written to file

filename is the filename of file, where the [URL](#) objects will be appended to.

nTries specifies the maximal number of times the method will try to acquire a lock on file to read.

tryInterval specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns:

true in case of success, otherwise false.

10.169.3.7 static bool Arc::Job::WriteJobIDToFile (const std::string &*jobid*, const std::string &*filename*, unsigned *nTries* = 10, unsigned *tryInterval* = 500000) [static]

Append a jobID to a file. This static method will put the ID represented by a [URL](#) object, and append it to the given file. File locking will be done as described for the ReadAllJobsFromFile method. It returns false if the file is not writable, true otherwise.

Parameters:

jobid is a jobID as a [URL](#) object

filename is the filename of the jobidfile, where the jobID will be appended

nTries specifies the maximal number of times the method will try to acquire a lock on file to read.

tryInterval specifies the interval (in micro seconds) between each attempt to acquire a lock.

Returns:

true in case of success, otherwise false.

10.169.4 Field Documentation

10.169.4.1 `std::string Arc::Job::JobDescription`

Language of job description describing job. Equivalent to the [GLUE2](#) ComputingActivity entity [JobDescription](#) (open enumeration), which here is represented by a string.

10.169.4.2 `std::string Arc::Job::JobDescriptionDocument`

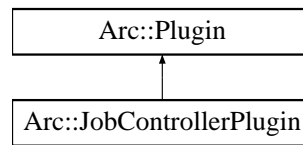
[Job](#) description document describing job. No [GLUE2](#) entity equivalent. Should hold the job description document which was submitted to the computing service for this job.

The documentation for this class was generated from the following file:

- Job.h

10.170 Arc::JobControllerPlugin Class Reference

Inheritance diagram for Arc::JobControllerPlugin::

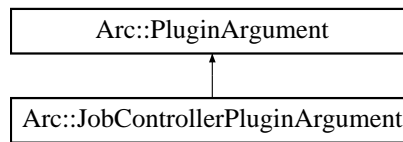


The documentation for this class was generated from the following file:

- [JobControllerPlugin.h](#)

10.171 Arc::JobControllerPluginArgument Class Reference

Inheritance diagram for Arc::JobControllerPluginArgument::

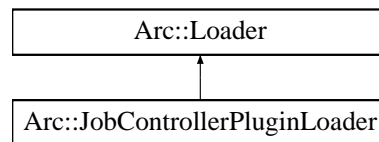


The documentation for this class was generated from the following file:

- [JobControllerPlugin.h](#)

10.172 Arc::JobControllerPluginLoader Class Reference

#include <arc/compute/JobControllerPlugin.h> Inheritance diagram for Arc::JobControllerPluginLoader::



Public Member Functions

- [JobControllerPluginLoader\(\)](#)
- [~JobControllerPluginLoader\(\)](#)
- [JobControllerPlugin * load](#) (const std::string &name, const [UserConfig](#) &uc)

10.172.1 Detailed Description

Class responsible for loading [JobControllerPlugin](#) plugins The [JobControllerPlugin](#) objects returned by a [JobControllerPluginLoader](#) must not be used after the [JobControllerPluginLoader](#) goes out of scope.

10.172.2 Constructor & Destructor Documentation

10.172.2.1 Arc::JobControllerPluginLoader::JobControllerPluginLoader ()

Constructor Creates a new [JobControllerPluginLoader](#).

10.172.2.2 Arc::JobControllerPluginLoader::~~JobControllerPluginLoader ()

Destructor Calling the destructor destroys all [JobControllerPlugins](#) loaded by the [JobControllerPluginLoader](#) instance.

10.172.3 Member Function Documentation

10.172.3.1 JobControllerPlugin* Arc::JobControllerPluginLoader::load (const std::string &name, const UserConfig &uc)

Load a new [JobControllerPlugin](#)

Parameters:

name The name of the [JobControllerPlugin](#) to load.

usercfg The [UserConfig](#) object for the new [JobControllerPlugin](#).

Returns:

A pointer to the new [JobControllerPlugin](#) (NULL on error).

The documentation for this class was generated from the following file:

- [JobControllerPlugin.h](#)

10.173 Arc::JobControllerPluginTestACCControl Class Reference

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.174 Arc::JobDescription Class Reference

```
#include <arc/compute/JobDescription.h>
```

Public Member Functions

- [JobDescriptionResult UnParse](#) (std::string &product, std::string language, const std::string &dialect="") const
- const std::string & [GetSourceLanguage](#) () const
- [JobDescriptionResult SaveToStream](#) (std::ostream &out, const std::string &format) const
- bool [Prepare](#) (const [ExecutionTarget](#) &et)

Static Public Member Functions

- static [JobDescriptionResult Parse](#) (const std::string &source, std::list< [JobDescription](#) > &jobdescs, const std::string &language="", const std::string &dialect="")

Data Fields

- std::map< std::string, std::string > [OtherAttributes](#)

10.174.1 Detailed Description

The [JobDescription](#) class is the internal representation of a job description in the ARC-lib. It is structured into a number of other classes/objects which should strictly follow the description given in the job description document <http://svn.nordugrid.org/trac/nordugrid/browser/arcl/trunk/doc/tech_doc/client/job_description.odt>.

The class consist of a parsing method [JobDescription::Parse](#) which tries to parse the passed source using a number of different parsers. The parser method is complemented by the [JobDescription::UnParse](#) method, a method to generate a job description document in one of the supported formats. Additionally the internal representation is contained in public members which makes it directly accessible and modifiable from outside the scope of the class.

10.174.2 Member Function Documentation

10.174.2.1 const std::string& Arc::JobDescription::GetSourceLanguage () const [inline]

Get input source language. If this object was created by a [JobDescriptionParserPlugin](#), then this method returns a string which indicates the job description language of the parsed source. If not created by a [JobDescriptionParser](#) the string returned is empty.

Returns:

const std::string& source language of parsed input source.

10.174.2.2 `static JobDescriptionResult Arc::JobDescription::Parse (const std::string & source, std::list< JobDescription > & jobdescs, const std::string & language = "", const std::string & dialect = "") [static]`

Parse string into [JobDescription](#) objects. The passed string will be tried parsed into the list of [JobDescription](#) objects. The available specialized [JobDescriptionParser](#) classes will be tried one by one, parsing the string, and if one succeeds the list of [JobDescription](#) objects is filled with the parsed contents and true is returned, otherwise false is returned. If no language specified, each [JobDescriptionParserPlugin](#) will try all its supported languages. On the other hand if a language is specified, only the [JobDescriptionParserPlugin](#) supporting that language will be tried. A dialect can also be specified, which only has an effect on the parsing if the [JobDescriptionParserPlugin](#) supports that dialect.

Parameters:

source
jobdescs
language
dialect

Returns:

true if the passed string can be parsed successfully by any of the available parsers.

10.174.2.3 `bool Arc::JobDescription::Prepare (const ExecutionTarget & et) [inline]`

Prepare for submission to target. The Prepare method, is used to check and adapt the [JobDescription](#) object to the passed [ExecutionTarget](#) object before submitting the job description to the target. This method is normally called by [SubmitterPlugin](#) plugin classes, before submitting the job description. First the method checks the [DataStaging.InputFiles](#) list, for identical file names, and non-existent local input files. If any of such files are found, the method returns false. Then if the [Application.Executable](#) and [Application.Input](#) objects are specified as local input files, and they are not among the files in the [DataStaging.InputFiles](#) list a existence check will be done and if not found, false will be returned, otherwise they will be added to the list. Likewise if the [Application.Output](#), [Application.Error](#) and [Application.LogDir](#) attributes have been specified, and is not among the files in the [DataStaging.OutputFiles](#) list, they will be added to this list. After the file check, the [Resources.RunTimeEnvironment](#), [Resources.CETType](#) and [Resources.OperatingSystem](#) [SoftwareRequirement](#) objects are respectively resolved against the [ExecutionTarget::ApplicationEnvironments](#), [ExecutionTarget::Implementation](#) and [ExecutionTarget::OperatingSystem](#) [Software](#) objects using the [SoftwareRequirement::selectSoftware](#) method. If that method returns false i.e. unable to resolve the requirements false will be returned. After resolving software requirements, the value of the [Resources.QueueName](#) attribute will be set to that of the [ExecutionTarget::ComputingShareName](#) attribute, and then true is returned.

Parameters:

et [ExecutionTarget](#) object which to resolve software requirements against, and to pick up queue name from.

Returns:

false is returned is file checks fails, or if unable to resolve software requirements.

References [Prepare\(\)](#).

Referenced by [Prepare\(\)](#).

10.174.2.4 JobDescriptionResult Arc::JobDescription::SaveToStream (std::ostream & *out*, const std::string & *format*) const

Print job description to a std::ostream object. The job description will be written to the passed std::ostream object out in the format indicated by the format parameter. The format parameter should specify the format of one of the job description languages supported by the library. Or by specifying the special "user" or "userlong" format the job description will be written as a attribute/value pair list with respectively less or more attributes.

The mote

Returns:

true if writing the job description to the out object succeeds, otherwise false.

Parameters:

out a std::ostream reference specifying the ostream to write the job description to.

format specifies the format the job description should written in.

10.174.2.5 JobDescriptionResult Arc::JobDescription::UnParse (std::string & *product*, std::string *language*, const std::string & *dialect* = "") const

Output contents in the specified language.

Parameters:

product

language

dialect

Returns:

10.174.3 Field Documentation

10.174.3.1 std::map<std::string, std::string> Arc::JobDescription::OtherAttributes

Holds attributes not fitting into this class. This member is used by [JobDescriptionParserPlugin](#) classes to store attribute/value pairs not fitting into attributes stored in this class. The form of the attribute (the key in the map) should be as follows: <language>;<attribute-name> E.g.: "nordugrid:xrsl;hostname".

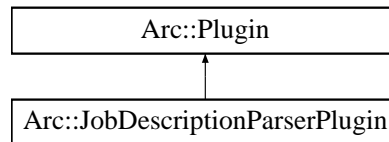
The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.175 Arc::JobDescriptionParserPlugin Class Reference

Abstract class for the different parsers.

`#include <arc/compute/JobDescriptionParserPlugin.h>`
Inheritance diagram for Arc::JobDescriptionParserPlugin::



10.175.1 Detailed Description

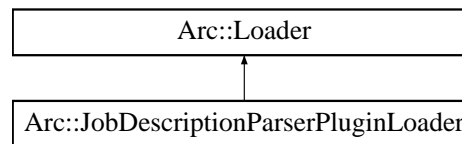
Abstract class for the different parsers. The [JobDescriptionParserPlugin](#) class is abstract which provide a interface for job description parsers. A job description parser should inherit this class and overwrite the `JobDescriptionParserPlugin::Parse` and `JobDescriptionParserPlugin::UnParse` methods.

The documentation for this class was generated from the following file:

- [JobDescriptionParserPlugin.h](#)

10.176 Arc::JobDescriptionParserPluginLoader Class Reference

`#include <arc/compute/JobDescriptionParserPlugin.h>` Inheritance diagram for Arc::JobDescriptionParserPluginLoader:



Data Structures

- class [iterator](#)

Public Member Functions

- [JobDescriptionParserPluginLoader](#) ()
- [~JobDescriptionParserPluginLoader](#) ()
- [JobDescriptionParserPlugin * load](#) (const std::string &name)
- const std::list< [JobDescriptionParserPlugin *](#) > & [GetJobDescriptionParserPlugins](#) () const

10.176.1 Detailed Description

Class responsible for loading [JobDescriptionParserPlugin](#) plugins The [JobDescriptionParserPlugin](#) objects returned by a [JobDescriptionParserPluginLoader](#) must not be used after the [JobDescriptionParserPluginLoader](#) goes out of scope.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 Arc::JobDescriptionParserPluginLoader::JobDescriptionParserPluginLoader ()

Constructor Creates a new [JobDescriptionParserPluginLoader](#).

10.176.2.2 Arc::JobDescriptionParserPluginLoader::~~JobDescriptionParserPluginLoader ()

Destructor Calling the destructor destroys all [JobDescriptionParserPlugin](#) object loaded by the [JobDescriptionParserPluginLoader](#) instance.

10.176.3 Member Function Documentation

10.176.3.1 const std::list<JobDescriptionParserPlugin*>& Arc::JobDescriptionParserPluginLoader::GetJobDescriptionParserPlugins () const [inline]

Retrieve the list of loaded [JobDescriptionParserPlugin](#) objects.

Returns:

A reference to the list of [JobDescriptionParserPlugin](#) objects.

10.176.3.2 JobDescriptionParserPlugin* Arc::JobDescriptionParserPluginLoader::load (const std::string & *name*)

Load a new [JobDescriptionParserPlugin](#)

Parameters:

name The name of the [JobDescriptionParserPlugin](#) to load.

Returns:

A pointer to the new [JobDescriptionParserPlugin](#) (NULL on error).

The documentation for this class was generated from the following file:

- [JobDescriptionParserPlugin.h](#)

10.177 Arc::JobDescriptionParserPluginResult Class Reference

The documentation for this class was generated from the following file:

- [JobDescriptionParserPlugin.h](#)

10.178 Arc::JobDescriptionParserPluginTestACCControl Class Reference

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.179 Arc::JobDescriptionResult Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.180 Arc::JobIdentificationType Class Reference

[Job](#) identification.

```
#include <arc/compute/JobDescription.h>
```

Data Fields

- std::string [JobName](#)
- std::string [Description](#)
- std::string [Type](#)
- std::list< std::string > [Annotation](#)
- std::list< std::string > [ActivityOldID](#)

10.180.1 Detailed Description

[Job](#) identification. This class serves to provide human readable information about a job description. Some of this information might also be passed to the execution service for providing information about the job created from this job description. An object of this class is part of the [JobDescription](#) class as the Identification public member.

10.180.2 Field Documentation

10.180.2.1 std::list<std::string> Arc::JobIdentificationType::ActivityOldID

ID of old activity. The ActivityOldID object is used to store a list of IDs corresponding to activities which were performed from this job description. This information is not intended to be used by the execution service, but rather used for keeping track of activities, e.g. when doing a job resubmission the old activity ID is appended to this list.

10.180.2.2 std::list<std::string> Arc::JobIdentificationType::Annotation

Annotation. The Annotation list is used for human readable comments, tags for free grouping or identifying different activities.

10.180.2.3 std::string Arc::JobIdentificationType::Description

Human readable description. The Description string can be used to provide a human readable description of e.g. the task which should be performed when processing the job description.

10.180.2.4 std::string Arc::JobIdentificationType::JobName

Name of job. The JobName string is used to specify a name of the job description, and it will most likely also be the name given to the job when created at the execution service.

10.180.2.5 `std::string Arc::JobIdentificationType::Type`

[Job](#) type. The Type string specifies a classification of the activity in compliance with [GLUE2](#). The possible values should follow those defined in the `ComputingActivityType_t` enumeration of [GLUE2](#).

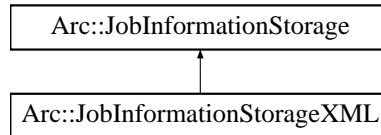
The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.181 Arc::JobInformationStorage Class Reference

Abstract class for storing job information.

#include <arc/compute/Job.h> Inheritance diagram for Arc::JobInformationStorage::



Public Member Functions

- [JobInformationStorage](#) (const std::string &name, unsigned nTries=10, unsigned tryInterval=500000)
- virtual bool [ReadAll](#) (std::list< [Job](#) > &jobs, const std::list< std::string > &rejectEndpoints=std::list< std::string >())=0
- virtual bool [Read](#) (std::list< [Job](#) > &jobs, std::list< std::string > &jobIdentifiers, const std::list< std::string > &endpoints=std::list< std::string >(), const std::list< std::string > &rejectEndpoints=std::list< std::string >())=0
- virtual bool [Write](#) (const std::list< [Job](#) > &jobs)
- virtual bool [Write](#) (const std::list< [Job](#) > &jobs, const std::set< std::string > &prunedServices, std::list< const [Job](#) * > &newJobs)=0
- virtual bool [Clean](#) ()=0
- virtual bool [Remove](#) (const std::list< std::string > &jobids)=0
- const std::string & [GetName](#) () const

10.181.1 Detailed Description

Abstract class for storing job information. This abstract class provides an interface which can be used to store job information, which can then later be used to initialise [Job](#) objects from the stored information.

Note:

This class is abstract. All functionality is provided by specialised child classes.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 Arc::JobInformationStorage::JobInformationStorage (const std::string & name, unsigned nTries = 10, unsigned tryInterval = 500000) [inline]

Constructor. Construct a [JobInformationStorage](#) object with name *name*. The name could be a file name or maybe a database, that is implementation specific. The *nTries* argument specifies the number times a lock on the storage should be tried obtained for each method invocation. The constructor it self should not acquire a lock through-out the object lifetime. *tryInterval* is the waiting period in micro seconds between each locking attempt.

Parameters:

name name of the storage.

nTries specifies the maximal number of times try to acquire a lock on file to read.

tryInterval specifies the interval (in micro seconds) between each attempt to acquire a lock.

10.181.3 Member Function Documentation

10.181.3.1 `virtual bool Arc::JobInformationStorage::Clean () [pure virtual]`

Clean storage. Invoking this method causes the storage to be cleaned of any jobs it holds.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Returns:

`true` is returned if the storage was successfully cleaned, otherwise `false` is returned.

Implemented in [Arc::JobInformationStorageXML](#).

10.181.3.2 `const std::string& Arc::JobInformationStorage::GetName () const [inline]`

Get name.

Returns:

Returns the name of the storage.

10.181.3.3 `virtual bool Arc::JobInformationStorage::Read (std::list< Job > & jobs, std::list< std::string > & jobIdentifiers, const std::list< std::string > & endpoints = std::list< std::string > (), const std::list< std::string > & rejectEndpoints = std::list< std::string > ()) [pure virtual]`

Read specified jobs. Read jobs specified by job identifiers and/or endpoints from storage. Only jobs which has a JobID or a Name attribute matching any of the items in the `identifiers` list parameter, and also jobs for which the `JobManagementURL` attribute matches any of those endpoints specified in the `endpoints` list parameter, will be added to the list of [Job](#) objects reference to by the `jobs` parameter, except those jobs for which the `JobManagementURL` attribute matches any of those endpoints specified in the `rejectEndpoints` list parameter. Identifiers specified in the `jobIdentifiers` list parameter which matches a job in the storage will be removed from the referenced list. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs reference to list of [Job](#) objects which will be filled with matching jobs.

jobIdentifiers specifies the job IDs and names of jobs to be added to the job list. Entries in this list is removed if they match a job from the storage.

endpoints is a list of strings specifying endpoints for which [Job](#) objects with the `JobManagementURL` attribute matching any of those endpoints will added to the job list. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

rejectEndpoints is a list of strings specifying endpoints for which [Job](#) objects with the `JobManagementURL` attribute matching any of those endpoints will not be part of the retrieved jobs. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Returns:

`false` is returned in case a job failed to be read from storage, otherwise `true` is returned. This method will also return in case an identifier does not match any jobs in the storage.

Implemented in [Arc::JobInformationStorageXML](#).

10.181.3.4 `virtual bool Arc::JobInformationStorage::ReadAll (std::list< Job > &jobs, const std::list< std::string > &rejectEndpoints = std::list< std::string >()) [pure virtual]`

Read all jobs from storage. Read all jobs contained in storage, except those managed by a service at an endpoint which matches any of those in the `rejectEndpoints` list parameter. The read jobs are added to the list of [Job](#) objects referenced by the `jobs` parameter. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs is a reference to a list of [Job](#) objects, which will be filled with the jobs read from file (cleared before use).

rejectEndpoints is a list of strings specifying endpoints for which [Job](#) objects with `JobManagementURL` matching any of those endpoints will not be part of the retrieved jobs. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Returns:

`true` is returned if all jobs contained in the storage was retrieved (except those rejected, if any), otherwise `false`.

Implemented in [Arc::JobInformationStorageXML](#).

10.181.3.5 `virtual bool Arc::JobInformationStorage::Remove (const std::list< std::string > &jobids) [pure virtual]`

Remove jobs. The jobs with matching job IDs (`Job::JobID` attribute) as specified with the list of job IDs (`jobids` parameter) will be remove from the storage.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobids list job IDs for which matching jobs should be remove from storage.

Returns:

`is` returned if any of the matching jobs failed to be removed from the storage, otherwise `true` is returned.

Implemented in [Arc::JobInformationStorageXML](#).

10.181.3.6 `virtual bool Arc::JobInformationStorage::Write (const std::list< Job > & jobs, const std::set< std::string > & prunedServices, std::list< const Job * > & newJobs) [pure virtual]`

Write jobs. Add jobs to storage. If there already exist a job with a specific job ID in the storage, and a job with the same job ID is tried added to the storage then the existing job will be overwritten. For jobs in the storage with a ServiceEndpointURL attribute where the host name is equal to any of the entries in the set referenced by the `prunedServices` parameter, is removed from the storage, if they are not among the list of jobs referenced by the `jobs` parameter. A pointer to jobs in the job list (`jobs`) which does not already exist in the storage will be added to the list of `Job` object pointers referenced by the `newJobs` parameter.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs is the list of `Job` objects which should be added to the storage.

prunedServices is a set of host names of services whose jobs should be removed if not replaced. This is typically the list of host names for which at least one endpoint was successfully queried. By passing an empty set, all existing jobs are kept, even if jobs are outdated.

newJobs is a reference to a list of pointers to `Job` objects which are not duplicates.

Returns:

`true` is returned if all jobs in the `jobs` list are written to to storage, otherwise `false` is returned.

Implemented in [Arc::JobInformationStorageXML](#).

10.181.3.7 `virtual bool Arc::JobInformationStorage::Write (const std::list< Job > & jobs) [inline, virtual]`

Write jobs. Add jobs to storage. If there already exist a job with a specific job ID in the storage, and a job with the same job ID is tried added to the storage then the existing job will be overwritten.

A specialised implementaion does not necessarily need to be provided. If not provided `Write(const std::list<Job>&, std::set<std::string>&, std::list<const Job*>&)` will be used.

Parameters:

jobs is the list of `Job` objects which should be added to the storage.

Returns:

`true` is returned if all jobs in the `jobs` list are written to to storage, otherwise `false` is returned.

See also:

`Write(const std::list<Job>&, std::set<std::string>&, std::list<const Job*>&)`

References `Write()`.

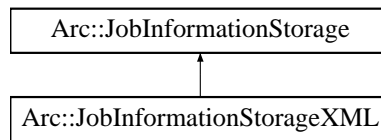
Referenced by `Write()`.

The documentation for this class was generated from the following file:

- `Job.h`

10.182 Arc::JobInformationStorageXML Class Reference

Inheritance diagram for Arc::JobInformationStorageXML::



Public Member Functions

- bool [ReadAll](#) (std::list< [Job](#) > &jobs, const std::list< std::string > &rejectEndpoints=std::list< std::string >())
- bool [Read](#) (std::list< [Job](#) > &jobs, std::list< std::string > &jobIdentifiers, const std::list< std::string > &endpoints=std::list< std::string >(), const std::list< std::string > &rejectEndpoints=std::list< std::string >())
- bool [Write](#) (const std::list< [Job](#) > &jobs, const std::set< std::string > &prunedServices, std::list< const [Job](#) * > &newJobs)
- bool [Clean](#) ()
- bool [Remove](#) (const std::list< std::string > &jobids)

10.182.1 Member Function Documentation

10.182.1.1 bool Arc::JobInformationStorageXML::Clean () [virtual]

Clean storage. Invoking this method causes the storage to be cleaned of any jobs it holds.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Returns:

`true` is returned if the storage was successfully cleaned, otherwise `false` is returned.

Implements [Arc::JobInformationStorage](#).

10.182.1.2 bool Arc::JobInformationStorageXML::Read (std::list< [Job](#) > &jobs, std::list< std::string > &jobIdentifiers, const std::list< std::string > &endpoints = std::list< std::string >(), const std::list< std::string > &rejectEndpoints = std::list< std::string >()) [virtual]

Read specified jobs. Read jobs specified by job identifiers and/or endpoints from storage. Only jobs which has a JobID or a Name attribute matching any of the items in the `identifiers` list parameter, and also jobs for which the `JobManagementURL` attribute matches any of those endpoints specified in the `endpoints` list parameter, will be added to the list of [Job](#) objects reference to by the `jobs` parameter, except those jobs for which the `JobManagementURL` attribute matches any of those endpoints specified in the `rejectEndpoints` list parameter. Identifiers specified in the `jobIdentifiers` list parameter which matches a job in the storage will be removed from the referenced list. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs reference to list of [Job](#) objects which will be filled with matching jobs.

jobIdentifiers specifies the job IDs and names of jobs to be added to the job list. Entries in this list is removed if they match a job from the storage.

endpoints is a list of strings specifying endpoints for which [Job](#) objects with the `JobManagementURL` attribute matching any of those endpoints will added to the job list. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

rejectEndpoints is a list of strings specifying endpoints for which [Job](#) objects with the `JobManagementURL` attribute matching any of those endpoints will not be part of the retrieved jobs. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Returns:

`false` is returned in case a job failed to be read from storage, otherwise `true` is returned. This method will also return in case an identifier does not match any jobs in the storage.

Implements [Arc::JobInformationStorage](#).

10.182.1.3 `bool Arc::JobInformationStorageXML::ReadAll (std::list< Job > & jobs, const std::list< std::string > & rejectEndpoints = std::list< std::string > ()) [virtual]`

Read all jobs from storage. Read all jobs contained in storage, except those managed by a service at an endpoint which matches any of those in the `rejectEndpoints` list parameter. The read jobs are added to the list of [Job](#) objects referenced by the `jobs` parameter. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs is a reference to a list of [Job](#) objects, which will be filled with the jobs read from file (cleared before use).

rejectEndpoints is a list of strings specifying endpoints for which [Job](#) objects with `JobManagementURL` matching any of those endpoints will not be part of the retrieved jobs. The algorithm used for matching should be equivalent to that used in the [URL::StringMatches](#) method.

Returns:

`true` is returned if all jobs contained in the storage was retrieved (except those rejected, if any), otherwise `false`.

Implements [Arc::JobInformationStorage](#).

10.182.1.4 **bool Arc::JobInformationStorageXML::Remove (const std::list< std::string > & *jobids*) [virtual]**

Remove jobs. The jobs with matching job IDs (Job::JobID attribute) as specified with the list of job IDs (*jobids* parameter) will be remove from the storage.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobids list job IDs for which matching jobs should be remove from storage.

Returns:

is returned if any of the matching jobs failed to be removed from the storage, otherwise *true* is returned.

Implements [Arc::JobInformationStorage](#).

10.182.1.5 **bool Arc::JobInformationStorageXML::Write (const std::list< Job > & *jobs*, const std::set< std::string > & *prunedServices*, std::list< const Job * > & *newJobs*) [virtual]**

Write jobs. Add jobs to storage. If there already exist a job with a specific job ID in the storage, and a job with the same job ID is tried added to the storage then the existing job will be overwritten. For jobs in the storage with a ServiceEndpointURL attribute where the host name is equal to any of the entries in the set referenced by the *prunedServices* parameter, is removed from the storage, if they are not among the list of jobs referenced by the *jobs* parameter. A pointer to jobs in the job list (*jobs*) which does not already exist in the storage will be added to the list of [Job](#) object pointers referenced by the *newJobs* parameter.

Note:

This method is abstract and an implementation must be provided by specialised classes.

Parameters:

jobs is the list of [Job](#) objects which should be added to the storage.

prunedServices is a set of host names of services whose jobs should be removed if not replaced. This is typically the list of host names for which at least one endpoint was successfully queried. By passing an empty set, all existing jobs are kept, even if jobs are outdated.

newJobs is a reference to a list of pointers to [Job](#) objects which are not duplicates.

Returns:

true is returned if all jobs in the *jobs* list are written to to storage, otherwise *false* is returned.

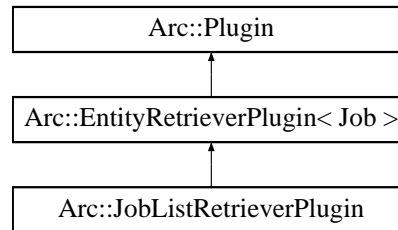
Implements [Arc::JobInformationStorage](#).

The documentation for this class was generated from the following file:

- Job.h

10.183 Arc::JobListRetrieverPlugin Class Reference

Inheritance diagram for Arc::JobListRetrieverPlugin::



The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

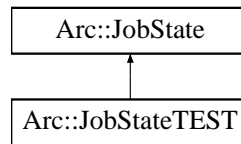
10.184 Arc::JobListRetrieverPluginTESTControl Class Reference

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.185 Arc::JobState Class Reference

`#include <arc/compute/JobState.h>` Inheritance diagram for Arc::JobState::



Public Member Functions

- bool [IsFinished](#) () const
- const std::string & [operator](#)() () const
- const std::string & [GetGeneralState](#) () const
- std::string [GetSpecificState](#) () const

10.185.1 Detailed Description

ARC general state model. The class comprise the general state model of the ARC-lib, and are herein used to compare job states from the different middlewares supported by the plugin structure of the ARC-lib. Which is why every ACC plugin should contain a class derived from this class. The derived class should consist of a constructor and a mapping function (a JobStateMap) which maps a std::string to a [JobState](#):StateType. An example of a constructor in a plugin could be: `JobStatePlugin::JobStatePlugging(const std::string& state) : JobState(state, &pluginStateMap) {}` where `&pluginStateMap` is a reference to the JobStateMap defined by the derived class.

10.185.2 Member Function Documentation

10.185.2.1 const std::string& Arc::JobState::GetGeneralState () const `[inline]`

General string representation of job state. Get the string representation of the job state as mapped to the libarccompute job state model.

Returns:

string representing general job state

See also:

enum StateType
[GetSpecificState](#)

10.185.2.2 std::string Arc::JobState::GetSpecificState () const `[inline]`

Specific string representation of job state. Get the string representation of the job state as returned by the CE service possibly formatted to a human readable string.

Returns:

string representing specific, possibly formatted, job state

See also:

[GetGeneralState](#)
[operator\(\)](#)

10.185.2.3 bool Arc::JobState::IsFinished () const [inline]

Check if state is finished.

Returns:

true is returned if the StateType is equal to FINISHED, KILLED, FAILED or DELETED, otherwise false is returned.

10.185.2.4 const std::string& Arc::JobState::operator() () const [inline]

Unformatted specific job state. Get the unformatted specific job state as returned by the CE.

Returns:

job state as returned by CE

See also:

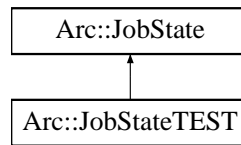
[GetSpecificState](#)
[GetGeneralState](#)

The documentation for this class was generated from the following file:

- JobState.h

10.186 Arc::JobStateTEST Class Reference

Inheritance diagram for Arc::JobStateTEST::



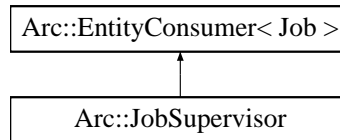
The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.187 Arc::JobSupervisor Class Reference

[JobSupervisor](#) class.

`#include <arc/compute/JobSupervisor.h>` Inheritance diagram for Arc::JobSupervisor::



Public Member Functions

- [JobSupervisor](#) (const [UserConfig](#) &usercfg, const std::list< [Job](#) > &jobs=std::list< [Job](#) >())
- bool [AddJob](#) (const [Job](#) &job)
- void [addEntity](#) (const [Job](#) &job)
- void [Update](#) ()
- bool [Retrieve](#) (const std::string &downloadirprefix, bool usejobname, bool force, std::list< std::string > &downloaddirectories)
- bool [Renew](#) ()
- bool [Resume](#) ()
- bool [Resubmit](#) (int destination, const std::list< [Endpoint](#) > &, std::list< [Job](#) > &resubmittedJobs, const std::list< std::string > &=std::list< std::string >())
- bool [Migrate](#) (bool forcemigration, const std::list< [Endpoint](#) > &, std::list< [Job](#) > &migratedJobs, const std::list< std::string > &=std::list< std::string >())
- bool [Cancel](#) ()
- bool [Clean](#) ()

10.187.1 Detailed Description

[JobSupervisor](#) class. The [JobSupervisor](#) class is tool for loading [JobControllerPlugin](#) plugins for managing Grid jobs.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 Arc::JobSupervisor::JobSupervisor (const UserConfig & usercfg, const std::list< Job > &jobs = std::list< Job >())

Create a [JobSupervisor](#). The list of [Job](#) objects passed to the constructor will be managed by this [JobSupervisor](#), through the [JobControllerPlugin](#) class. It is important that the InterfaceName member of each [Job](#) object is set and names a interface supported by one of the available [JobControllerPlugin](#) plugins. The [JobControllerPlugin](#) plugin will be loaded using the [JobControllerPluginLoader](#) class, loading a plugin of type "HED:JobControllerPlugin" which supports the particular interface, and the a reference to the [UserConfig](#) object usercfg will be passed to the plugin. Additionally a reference to the [UserConfig](#) object usercfg will be stored, thus usercfg must exist throughout the scope of the created object. If the InterfaceName member of a [Job](#) object is unset, a VERBOSE log message will be reported and that [Job](#) object will be ignored. If the [JobControllerPlugin](#) plugin for a given interface cannot be loaded, a WARNING log message will be reported and any [Job](#) object requesting that interface will be ignored. If loading of a specific plugin failed,

that plugin will not be tried loaded for subsequent [Job](#) objects requiring that plugin. [Job](#) objects will be added to the corresponding [JobControllerPlugin](#) plugin, if loaded successfully.

Parameters:

- usercfg* [UserConfig](#) object to pass to [JobControllerPlugin](#) plugins and to use in member methods.
- jobs* List of [Job](#) objects which will be managed by the created object.

10.187.3 Member Function Documentation

10.187.3.1 void Arc::JobSupervisor::addEntity (const Job &) [inline, virtual]

Send an entity to this consumer. This is the method which will be called by the retrievers when a new result is available.

Implements [Arc::EntityConsumer< Job >](#).

References [AddJob\(\)](#).

10.187.3.2 bool Arc::JobSupervisor::AddJob (const Job & job)

Add job. Add [Job](#) object to this [JobSupervisor](#) for job management. The [Job](#) object will be passed to the corresponding specialized [JobControllerPlugin](#).

Parameters:

- job* [Job](#) object to add for job management

Returns:

- true is returned if the passed [Job](#) object was added to the underlying [JobControllerPlugin](#), otherwise false is returned and a log message emitted with the reason.

Referenced by [addEntity\(\)](#).

10.187.3.3 bool Arc::JobSupervisor::Cancel ()

Cancel jobs. This method cancels jobs managed by this [JobSupervisor](#).

Before identifying jobs to cancel, the [JobControllerPlugin::UpdateJobs](#) method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information.

Since jobs in the [JobState::DELETED](#), [JobState::FINISHED](#), [JobState::KILLED](#) or [JobState::FAILED](#) states is already in a terminal state, a cancel request will not be send for those. Also no request will be send for jobs in the [JobState::UNDEFINED](#) state, since job information is not available. If the status-filter is non-empty, a cancel request will only be send to jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter, excluding the states mentioned above.

For each job to be cancelled, the specialized [JobControllerPlugin::CancelJob](#) method is called and is responsible for cancelling the given job. If the method fails to cancel a job, this method will return false (otherwise true), and the job ID ([IDFromEndpoint](#)) of such jobs is appended to the notcancelled list. The job ID of successfully cancelled jobs will be appended to the passed cancelled list.

Returns:

- false if any call to [JobControllerPlugin::CancelJob](#) failed, true otherwise.

See also:

JobControllerPlugin::CancelJob.

10.187.3.4 bool Arc::JobSupervisor::Clean ()

Clean jobs. This method removes from services jobs managed by this [JobSupervisor](#). Before cleaning jobs, the JobController::GetInformation method is called in order to update job information, and that jobs are selected by job status instead of by job IDs. The status list argument should contain states for which cleaning of job in any of those states should be carried out. The states are compared using both the JobState::operator() and [JobState::GetGeneralState\(\)](#) methods. If the status list is empty, all jobs will be selected for cleaning.

Returns:

false if calls to JobControllerPlugin::CleanJob fails, true otherwise.

10.187.3.5 bool Arc::JobSupervisor::Migrate (bool *forcemigration*, const std::list< Endpoint > &, std::list< Job > & *migratedJobs*, const std::list< std::string > & = std::list< std::string >())

Migrate jobs. Jobs managed by this [JobSupervisor](#) will be migrated when invoking this method, that is the job description of a job will be tried obtained, and if successful a job migration request will be sent, based on that job description.

Before identifying jobs to be migrated, the JobControllerPlugin::UpdateJobs method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information. Only jobs for which the State member of the [Job](#) object has the value JobState::QUEUEING, will be considered for migration. Furthermore the job description must be obtained (either locally or remote) and successfully parsed in order for a job to be migrated. If the job description cannot be obtained or parsed an ERROR log message is reported, and the IDFromEndpoint [URL](#) of the [Job](#) object is appended to the notmigrated list. If no jobs have been identified for migration, false will be returned in case ERRORS were reported, otherwise true is returned.

The execution services which can be targeted for migration are those specified in the [UserConfig](#) object of this class, as selected services. Before initiating any job migration request, resource discovery and broker* loading is carried out using the TargetGenerator and [Broker](#) classes, initialised by the [UserConfig](#) object of this class. If [Broker](#) loading fails, or no ExecutionTargets are found, an ERROR log message is reported and all IDFromEndpoint URLs for job considered for migration will be appended to the notmigrated list and then false will be returned.

When the above checks have been carried out successfully, the following is done for each job considered for migration. The ActivityOldID member of the Identification member in the job description will be set to that of the [Job](#) object, and the IDFromEndpoint [URL](#) will be appended to ActivityOldID member of the job description. After that the [Broker](#) object will be used to find a suitable [ExecutionTarget](#) object, and if found a migrate request will tried sent using the ExecutionTarget::Migrate method, passing the [UserConfig](#) object of this class. The passed forcemigration boolean indicates whether the migration request at the service side should ignore failures in cancelling the existing queuing job. If the request succeeds, the corresponding new [Job](#) object is appended to the migratedJobs list. If no suitable [ExecutionTarget](#) objects are found an ERROR log message is reported and the IDFromEndpoint [URL](#) of the [Job](#) object is appended to the notmigrated list. When all jobs have been processed, false is returned if any ERRORS were reported, otherwise true.

Parameters:

forcemigration indicates whether migration should succeed if service fails to cancel the existing queuing job.

migratedJobs list of [Job](#) objects which migrated jobs will be appended to.

TODO

Returns:

false if any error is encountered, otherwise true.

10.187.3.6 bool Arc::JobSupervisor::Renew ()

Renew job credentials. This method will renew credentials of jobs managed by this [JobSupervisor](#).

Before identifying jobs for which to renew credentials, the `JobControllerPlugin::UpdateJobs` method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information.

Since jobs in the `JobState::DELETED`, `JobState::FINISHED` or `JobState::KILLED` states is in a terminal state credentials for those jobs will not be renewed. Also jobs in the `JobState::UNDEFINED` state will not get their credentials renewed, since job information is not available. The `JobState::FAILED` state is also a terminal state, but since jobs in this state can be restarted, credentials for such jobs can be renewed. If the status-filter is non-empty, a renewal of credentials will be done for jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter, excluding the already filtered states as mentioned above.

For each job for which to renew credentials, the specialized `JobControllerPlugin::RenewJob` method is called and is responsible for renewing the credentials for the given job. If the method fails to renew any job credentials, this method will return false (otherwise true), and the job ID (`IDFromEndpoint`) of such jobs is appended to the notrenewed list. The job ID of successfully renewed jobs will be appended to the passed renewed list.

Returns:

false if any call to `JobControllerPlugin::RenewJob` fails, true otherwise.

See also:

`JobControllerPlugin::RenewJob`.

10.187.3.7 bool Arc::JobSupervisor::Resubmit (int destination, const std::list< Endpoint > &, std::list< Job > & resubmittedJobs, const std::list< std::string > & = std::list< std::string > ())

Resubmit jobs. Jobs managed by this [JobSupervisor](#) will be resubmitted when invoking this method, that is the job description of a job will be tried obtained, and if successful a new job will be submitted.

Before identifying jobs to be resubmitted, the `JobControllerPlugin::UpdateJobs` method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information. If an empty status-filter is specified, all jobs managed by this [JobSupervisor](#) will be considered for resubmission, except jobs in the undefined state (see [JobState](#)). If the status-filter is not empty, then only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be considered, except jobs in the undefined state. Jobs for which a job description cannot be obtained and successfully parsed will not be considered and an ERROR log message is reported, and the `IDFromEndpoint URL` is appended to the notresubmitted list. [Job](#) descriptions will be tried obtained either from [Job](#) object itself, or fetching them

remotely. Furthermore if a [Job](#) object has the LocalInputFiles object set, then the checksum of each of the local input files specified in that object (key) will be calculated and verified to match the checksum LocalInputFiles object (value). If checksums are not matching the job will be filtered, and an ERROR log message is reported and the IDFromEndpoint [URL](#) is appended to the notresubmitted list. If no job have been identified for resubmission, false will be returned if ERRORS were reported, otherwise true is returned.

The destination for jobs is partly determined by the destination parameter. If a value of 1 is specified a job will only be targeted to the execution service (ES) on which it reside. A value of 2 indicates that a job should not be targeted to the ES it currently reside. Specifying any other value will target any ES. The ESs which can be targeted are those specified in the [UserConfig](#) object of this class, as selected services. Before initiating any job submission, resource discovery and broker loading is carried out using the TargetGenerator and [Broker](#) classes, initialised by the [UserConfig](#) object of this class. If [Broker](#) loading fails, or no ExecutionTargets are found, an ERROR log message is reported and all IDFromEndpoint URLs for job considered for resubmission will be appended to the notresubmitted list and then false will be returned.

When the above checks have been carried out successfully, then the Broker::Submit method will be invoked for each considered for resubmission. If it fails the IDFromEndpoint [URL](#) for the job is appended to the notresubmitted list, and an ERROR is reported. If submission succeeds the new job represented by a [Job](#) object will be appended to the resubmittedJobs list - it will not be added to this [JobSupervisor](#). The method returns false if ERRORS were reported otherwise true is returned.

Parameters:

destination specifies how target destination should be determined (1 = same target, 2 = not same, any other = any target).

resubmittedJobs list of [Job](#) objects which resubmitted jobs will be appended to.

TODO

Returns:

false if any error is encountered, otherwise true.

10.187.3.8 bool Arc::JobSupervisor::Resume ()

Resume jobs by status. This method resumes jobs managed by this [JobSupervisor](#).

Before identifying jobs to resume, the JobControllerPlugin::UpdateJobs method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information.

Since jobs in the JobState::DELETED, JobState::FINISHED or JobState::KILLED states is in a terminal state credentials for those jobs will not be renewed. Also jobs in the JobState::UNDEFINED state will not be resumed, since job information is not available. The JobState::FAILED state is also a terminal state, but jobs in this state are allowed to be restarted. If the status-filter is non-empty, only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be resumed, excluding the already filtered states as mentioned above.

For each job to resume, the specialized JobControllerPlugin::ResumeJob method is called and is responsible for resuming the particular job. If the method fails to resume a job, this method will return false, otherwise true is returned. The job ID of successfully resumed jobs will be appended to the passed resumedJobs list.

Returns:

false if any call to JobControllerPlugin::ResumeJob fails, true otherwise.

See also:

`JobControllerPlugin::ResumeJob`.

10.187.3.9 `bool Arc::JobSupervisor::Retrieve (const std::string & downloaddirprefix, bool usejobname, bool force, std::list< std::string > & downloaddirdirectories)`

Retrieve job output files. This method retrieves output files of jobs managed by this [JobSupervisor](#).

Before identifying jobs for which to retrieve output files, the `JobControllerPlugin::UpdateJobs` method is called for each loaded [JobControllerPlugin](#) in order to retrieve the most up to date job information. If an empty status-filter is specified, all jobs managed by this [JobSupervisor](#) will be considered for retrieval, except jobs in the undefined state (see [JobState](#)). If the status-filter is not empty, then only jobs with a general or specific state (see [JobState](#)) identical to any of the entries in the status-filter will be considered, except jobs in the undefined state. Jobs in the state `JobState::DELETED` and unfinished jobs (see [JobState::IsFinished](#)) will also not be considered.

For each of the jobs considered for retrieval, the files will be downloaded to a directory named either as the last part of the job ID or the job name, which is determined by the 'usejobname' argument. The download directories will be located in the directory specified by the 'downloaddir' argument, as either a relative or absolute path. If the 'force' argument is set to 'true', and a download directory for a given job already exist it will be overwritten, otherwise files for that job will not be downloaded. This method calls the `JobControllerPlugin::GetJob` method in order to download jobs, and if a job is successfully retrieved the job ID will be appended to the 'retrievedJobs' list. If all jobs are successfully retrieved this method returns true, otherwise false.

Parameters:

downloaddir specifies the path to in which job download directories will be located.

usejobname specifies whether to use the job name or job ID as directory name to store job output files in.

force indicates whether existing job directories should be overwritten or not.

See also:

`JobControllerPlugin::RetrieveJob`.

Returns:

true if all jobs are successfully retrieved, otherwise false.

10.187.3.10 `void Arc::JobSupervisor::Update ()`

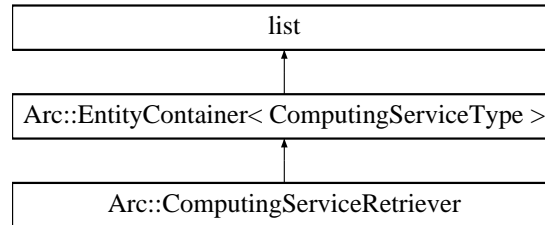
Update job information. When invoking this method the job information for the jobs managed by this [JobSupervisor](#) will be updated. Internally, for each loaded [JobControllerPlugin](#) the `JobControllerPlugin::UpdateJobs` method will be called, which will be responsible for updating job information.

The documentation for this class was generated from the following file:

- `JobSupervisor.h`

10.188 list Class Reference

Inheritance diagram for list::



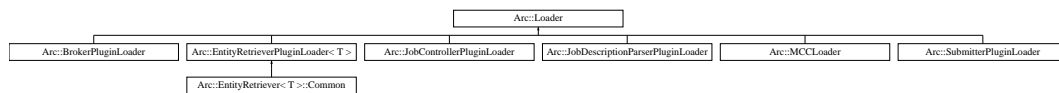
The documentation for this class was generated from the following file:

- EntityRetriever.h

10.189 Arc::Loader Class Reference

Plugins loader.

#include <Loader.h>Inheritance diagram for Arc::Loader::



Public Member Functions

- [Loader](#) ([XMLNode](#) cfg)
- [~Loader](#) ()

Protected Attributes

- [PluginsFactory](#) * [factory_](#)

10.189.1 Detailed Description

Plugins loader. This class processes XML configuration and loads specified plugins. Accepted configuration is defined by XML schema mcc.xsd. "Plugins" elements are parsed by this class and corresponding libraries are loaded. Main functionality is provided by class [PluginsFactory](#).

10.189.2 Constructor & Destructor Documentation

10.189.2.1 Arc::Loader::Loader (XMLNode cfg)

Constructor that takes whole XML configuration and performs common configuration part

10.189.2.2 Arc::Loader::~~Loader ()

Destructor destroys all components created by constructor

10.189.3 Field Documentation

10.189.3.1 PluginsFactory* Arc::Loader::factory_ [protected]

Link to Factory responsible for loading and creation of [Plugin](#) and derived objects

Referenced by Arc::ChainContext::operator PluginsFactory *().

The documentation for this class was generated from the following file:

- Loader.h

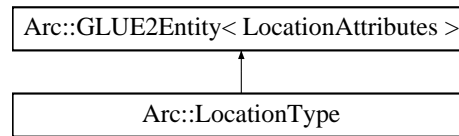
10.190 Arc::LocationAttributes Class Reference

The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.191 Arc::LocationType Class Reference

Inheritance diagram for Arc::LocationType::



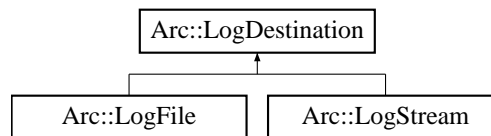
The documentation for this class was generated from the following file:

- [ExecutionTarget.h](#)

10.192 Arc::LogDestination Class Reference

A base class for log destinations.

`#include <arc/Logger.h>`Inheritance diagram for Arc::LogDestination::



Public Member Functions

- virtual void [log](#) (const [LogMessage](#) &message)=0
- void [setFormat](#) (const [LogFormat](#) &newformat)

Protected Member Functions

- [LogDestination](#) ()

Protected Attributes

- [LogFormat](#) format

10.192.1 Detailed Description

A base class for log destinations. This class defines an interface for LogDestinations. [LogDestination](#) objects will typically contain synchronization mechanisms and should therefore never be copied. If `setlocale()` has been called with a supported locale, log messages will be logged in that locale.

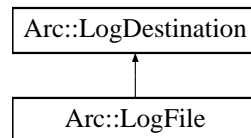
The documentation for this class was generated from the following file:

- `Logger.h`

10.193 Arc::LogFile Class Reference

A class for logging to files.

`#include <arc/Logger.h>`Inheritance diagram for Arc::LogFile::



Public Member Functions

- [LogFile](#) (const std::string &path)
- void [setMaxSize](#) (int newsize)
- void [setBackups](#) (int newbackup)
- void [setReopen](#) (bool newreopen)
- [operator bool](#) (void)
- bool [operator!](#) (void)
- virtual void [log](#) (const [LogMessage](#) &message)

10.193.1 Detailed Description

A class for logging to files. This class is used for logging to files. It provides synchronization in order to prevent different LogMessages to appear mixed with each other in the stream. It is possible to limit size of created file. Whenever specified size is exceeded file is deleted and new one is created. Old files may be moved into backup files instead of being deleted. Those files have names same as initial file with additional number suffix - similar to those found in /var/log of many Unix-like systems.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 Arc::LogFile::LogFile (const std::string & path)

Creates a [LogFile](#) connected to a file. Creates a [LogFile](#) connected to the file located at specified path. In order not to break synchronization, it is important not to connect more than one [LogFile](#) object to a certain file. If file does not exist it will be created.

Parameters:

path The path to file to which to write LogMessages.

10.193.3 Member Function Documentation

10.193.3.1 virtual void Arc::LogFile::log (const LogMessage & message) [virtual]

Writes a [LogMessage](#) to the file. This method writes a [LogMessage](#) to the file that is connected to this [LogFile](#) object. If after writing size of file exceeds one set by [setMaxSize\(\)](#) file is moved to backup and new one is created.

Parameters:

message The [LogMessage](#) to write.

Implements [Arc::LogDestination](#).

10.193.3.2 void Arc::LogFile::setBackups (int *newbackup*)

Set number of backups to store. Set number of backups to store. When file size exceeds one specified with [setMaxSize\(\)](#) file is closed and moved to one named path.1. If path.1 exists it is moved to path.2 and so on. Number of path.# files is one set in newbackup.

Parameters:

newbackup Number of backup files.

10.193.3.3 void Arc::LogFile::setMaxSize (int *newsiz*)

Set maximal allowed size of file. Set maximal allowed size of file. This value is not obeyed exactly. Specified size may be exceeded by amount of one [LogMessage](#). To disable limit specify -1.

Parameters:

newsiz Max size of log file.

10.193.3.4 void Arc::LogFile::setReopen (bool *newreopen*)

Set file reopen on every write. Set file reopen on every write. If set to true file is opened before writing every log record and closed afterward.

Parameters:

newreopen If file to be reopened for every log record.

The documentation for this class was generated from the following file:

- [Logger.h](#)

10.194 Arc::Logger Class Reference

A logger class.

```
#include <Logger.h>
```

Public Member Functions

- [Logger](#) ([Logger](#) &parent, const std::string &subdomain)
- [Logger](#) ([Logger](#) &parent, const std::string &subdomain, [LogLevel](#) threshold)
- [~Logger](#) ()
- void [addDestination](#) ([LogDestination](#) &destination)
- void [addDestinations](#) (const std::list< [LogDestination](#) * > &destinations)
- void [setDestinations](#) (const std::list< [LogDestination](#) * > &destinations)
- const std::list< [LogDestination](#) * > & [getDestinations](#) (void) const
- void [removeDestinations](#) (void)
- void [deleteDestinations](#) (void)
- void [setThreshold](#) ([LogLevel](#) threshold)
- [LogLevel](#) [getThreshold](#) () const
- void [setThreadContext](#) (void)
- void [msg](#) ([LogMessage](#) message)
- void [msg](#) ([LogLevel](#) level, const std::string &str)

Static Public Member Functions

- static [Logger](#) & [getRootLogger](#) ()
- static void [setThresholdForDomain](#) ([LogLevel](#) threshold, const std::list< std::string > &subdomains)
- static void [setThresholdForDomain](#) ([LogLevel](#) threshold, const std::string &domain)

10.194.1 Detailed Description

A logger class. This class defines a [Logger](#) to which LogMessages can be sent.

Every [Logger](#) (except for the rootLogger) has a parent [Logger](#). The domain of a [Logger](#) (a string that indicates the origin of LogMessages) is composed by adding a subdomain to the domain of its parent [Logger](#).

A [Logger](#) also has a threshold. Every [LogMessage](#) that have a level that is greater than or equal to the threshold is forwarded to any [LogDestination](#) connected to this [Logger](#) as well as to the parent [Logger](#).

Typical usage of the [Logger](#) class is to declare a global [Logger](#) object for each library/module/component to be used by all classes and methods there.

[Logger](#) messages may be localised according to the current locale. Some locales are better supported than others.

Example code for setting up logger in main():

```
// Set up stderr as a log stream
Arc::LogStream logcerr(std::cerr);
// Log message is prefixed by level only
logcerr.setFormat(Arc::ShortFormat);
// Add the stderr destination to the root logger
```

```
Arc::Logger::getRootLogger().addDestination(logcerr);
// Set the logging threshold to WARNING
Arc::Logger::getRootLogger().setThreshold(Arc::WARNING);

// Logger to use in main() - it inherits all properties from the root Logger
Arc::Logger logger(Arc::Logger::getRootLogger(), "main");
// this message will not be logged since it is below the threshold
logger.msg(Arc::INFO, "main started");
int i = 5;
// This message will be logged
logger.msg(Arc::ERROR, "Oops, an error occurred when i was %i", i);
```

10.194.2 Constructor & Destructor Documentation

10.194.2.1 Arc::Logger::Logger (Logger & *parent*, const std::string & *subdomain*)

Creates a logger. The threshold is inherited from its parent [Logger](#).

Parameters:

parent The parent [Logger](#) of the new [Logger](#).
subdomain The subdomain of the new logger.

10.194.2.2 Arc::Logger::Logger (Logger & *parent*, const std::string & *subdomain*, LogLevel *threshold*)

Creates a logger.

Parameters:

parent The parent [Logger](#) of the new [Logger](#).
subdomain The subdomain of the new logger.
threshold The threshold of the new logger.

10.194.3 Member Function Documentation

10.194.3.1 void Arc::Logger::addDestination (LogDestination & *destination*)

Adds a [LogDestination](#). Adds a [LogDestination](#) to which to forward LogMessages sent to this logger (if they pass the threshold). Since LogDestinations should not be copied, the new [LogDestination](#) is passed by reference and a pointer to it is kept for later use. It is therefore important that the [LogDestination](#) passed to this [Logger](#) exists at least as long as the [Logger](#) itself.

10.194.3.2 void Arc::Logger::addDestinations (const std::list< LogDestination * > & *destinations*)

Adds LogDestinations. See [addDestination\(LogDestination& destination\)](#).

10.194.3.3 const std::list<LogDestination*>& Arc::Logger::getDestinations (void) const

Obtains current LogDestinations. Returns list of pointers to [LogDestination](#) objects. Returned result refers directly to internal member of [Logger](#) instance. Hence it should not be used after this [Logger](#) is destroyed.

10.194.3.4 static Logger& Arc::Logger::getRootLogger () [static]

The root [Logger](#). This is the root [Logger](#). It is an ancestor of any other [Logger](#) and always exists.

10.194.3.5 void Arc::Logger::msg (LogLevel *level*, const std::string & *str*) [inline]

Logs a message text. Logs a message text string at the specified LogLevel. This is a convenience method to save some typing. It simply creates a [LogMessage](#) and sends it to the other [msg\(\)](#) methods. It is also possible to use [msg\(\)](#) with multiple arguments and printf-style string formatting, for example

```
logger.msg(INFO, "Operation no %i failed: %s", number, reason);
```

Parameters:

level The level of the message.

str The message text.

References [msg\(\)](#).

10.194.3.6 void Arc::Logger::msg (LogMessage *message*)

Sends a [LogMessage](#).

Parameters:

message The [LogMessage](#) to send.

Referenced by [msg\(\)](#), and [Arc::stringto\(\)](#).

10.194.3.7 void Arc::Logger::setDestinations (const std::list< LogDestination * > & *destinations*)

Set LogDestinations. A safe atomic way to remove and add LogDestinations.

10.194.3.8 void Arc::Logger::setThreadContext (void)

Creates per-thread context. Creates new context for this logger which becomes effective for operations initiated by this thread. All new threads started by this one will inherit new context. Context stores current threshold and pointers to destinations. Hence new context is identical to current one. One can modify new context using [setThreshold\(\)](#), [removeDestinations\(\)](#) and [addDestination\(\)](#). All such operations will not affect old context.

10.194.3.9 void Arc::Logger::setThreshold (LogLevel *threshold*)

Sets the logging threshold. This method sets the threshold of the [Logger](#). Any message sent to this [Logger](#) that has a level below this threshold will be discarded.

Parameters:

threshold The threshold

10.194.3.10 static void Arc::Logger::setThresholdForDomain (LogLevel *threshold*, const std::string & *domain*) [static]

Sets the threshold for domain. This method sets the default threshold of the domain. All new loggers created with specified domain will have specified threshold set by default. The domain is composed of all subdomains of all loggers in chain by merging them with '.' as separator.

Parameters:

threshold The threshold

domain The domain of logger

10.194.3.11 static void Arc::Logger::setThresholdForDomain (LogLevel *threshold*, const std::list< std::string > & *subdomains*) [static]

Sets the threshold for domain. This method sets the default threshold of the domain. All new loggers created with specified domain will have specified threshold set by default. The subdomains of all loggers in chain are matched against list of provided subdomains.

Parameters:

threshold The threshold

subdomains The subdomains of all loggers in chain

The documentation for this class was generated from the following file:

- Logger.h

10.195 Arc::LoggerFormat Struct Reference

Struct to contain LogFormat, to use with [operator<<\(std::ostream&, const LoggerFormat&\)](#).

```
#include <Logger.h>
```

Public Member Functions

- [LoggerFormat](#) ([LogFormat](#) format)

10.195.1 Detailed Description

Struct to contain LogFormat, to use with [operator<<\(std::ostream&, const LoggerFormat&\)](#).

The documentation for this struct was generated from the following file:

- [Logger.h](#)

10.196 Arc::LogMessage Class Reference

A class for log messages.

```
#include <arc/Logger.h>
```

Public Member Functions

- [LogMessage](#) ([LogLevel](#) level, const [IString](#) &message)
- [LogMessage](#) ([LogLevel](#) level, const [IString](#) &message, const std::string &identifier)
- [LogLevel](#) [getLevel](#) () const

Protected Member Functions

- void [setIdentifier](#) (std::string identifier)

Friends

- class [Logger](#)
- std::ostream & [operator<<](#) (std::ostream &os, const [LogMessage](#) &message)

10.196.1 Detailed Description

A class for log messages. This class is used to represent log messages internally. It contains the time the message was created, its level, from which domain it was sent, an identifier and the message text itself.

10.196.2 Constructor & Destructor Documentation

10.196.2.1 Arc::LogMessage::LogMessage (LogLevel level, const IString & message)

Creates a [LogMessage](#) with the specified level and message text. This constructor creates a [LogMessage](#) with the specified level and message text. The time is set automatically, the domain is set by the [Logger](#) to which the [LogMessage](#) is sent and the identifier is composed from the process ID and the address of the Thread object corresponding to the calling thread.

Parameters:

level The level of the [LogMessage](#).

message The message text.

10.196.2.2 Arc::LogMessage::LogMessage (LogLevel level, const IString & message, const std::string & identifier)

Creates a [LogMessage](#) with the specified attributes. This constructor creates a [LogMessage](#) with the specified level, message text and identifier. The time is set automatically and the domain is set by the [Logger](#) to which the [LogMessage](#) is sent.

Parameters:

level The level of the [LogMessage](#).

message The message text.

identifier The identifier of the [LogMessage](#).

10.196.3 Member Function Documentation

10.196.3.1 LogLevel Arc::LogMessage::getLevel () const

Returns the level of the [LogMessage](#). Returns the level of the [LogMessage](#).

Returns:

The level of the [LogMessage](#).

10.196.3.2 void Arc::LogMessage::setIdentifier (std::string *identifier*) [protected]

Sets the identifier of the [LogMessage](#). The purpose of this method is to allow subclasses (in case there are any) to set the identifier of a [LogMessage](#).

Parameters:

identifier The identifier.

10.196.4 Friends And Related Function Documentation

10.196.4.1 friend class Logger [friend]

The [Logger](#) class is a friend. The [Logger](#) class must have some privileges (e.g. ability to call the setDomain() method), therefore it is a friend.

10.196.4.2 std::ostream& operator<< (std::ostream & *os*, const LogMessage & *message*) [friend]

Printing of LogMessages to ostreams. Output operator so that LogMessages can be printed conveniently by LogDestinations.

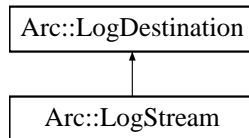
The documentation for this class was generated from the following file:

- [Logger.h](#)

10.197 Arc::LogStream Class Reference

A class for logging to ostreams.

#include <arc/Logger.h> Inheritance diagram for Arc::LogStream::



Public Member Functions

- [LogStream](#) (std::ostream &destination)
- virtual void [log](#) (const [LogMessage](#) &message)

10.197.1 Detailed Description

A class for logging to ostreams. This class is used for logging to ostreams (cout, cerr, files). It provides synchronization in order to prevent different LogMessages to appear mixed with each other in the stream. In order not to break the synchronization, LogStreams should never be copied. Therefore the copy constructor and assignment operator are private. Furthermore, it is important to keep a [LogStream](#) object as long as the [Logger](#) to which it has been registered.

10.197.2 Constructor & Destructor Documentation

10.197.2.1 Arc::LogStream::LogStream (std::ostream & destination)

Creates a [LogStream](#) connected to an ostream. Creates a [LogStream](#) connected to the specified ostream. In order not to break synchronization, it is important not to connect more than one [LogStream](#) object to a certain stream.

Parameters:

destination The ostream to which to write LogMessages.

10.197.3 Member Function Documentation

10.197.3.1 virtual void Arc::LogStream::log (const LogMessage & message) [virtual]

Writes a [LogMessage](#) to the stream. This method writes a [LogMessage](#) to the ostream that is connected to this [LogStream](#) object. It is synchronized so that not more than one [LogMessage](#) can be written at a time.

Parameters:

message The [LogMessage](#) to write.

Implements [Arc::LogDestination](#).

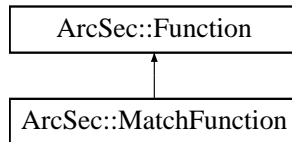
The documentation for this class was generated from the following file:

- `Logger.h`

10.198 ArcSec::MatchFunction Class Reference

Evaluate whether `arg1` (value in regular expression) matched `arg0` (lable in regular expression).

`#include <MatchFunction.h>` Inheritance diagram for `ArcSec::MatchFunction`:



Public Member Functions

- virtual `AttributeValue * evaluate (AttributeValue *arg0, AttribueValue *arg1, bool check_id=true)`
- virtual `std::list< AttribueValue * > evaluate (std::list< AttribueValue * > args, bool check_id=true)`

Static Public Member Functions

- static `std::string getFunctionName (std::string datatype)`

10.198.1 Detailed Description

Evaluate whether `arg1` (value in regular expression) matched `arg0` (lable in regular expression).

10.198.2 Member Function Documentation

10.198.2.1 `virtual std::list<AttributeValue*> ArcSec::MatchFunction::evaluate (std::list< AttribueValue * > args, bool check_id = true) [virtual]`

Evaluate a list of `AttributeValue` objects, and return a list of Attribute objects

Implements `ArcSec::Function`.

10.198.2.2 `virtual AttribueValue* ArcSec::MatchFunction::evaluate (AttribueValue * arg0, AttribueValue * arg1, bool check_id = true) [virtual]`

Evaluate two `AttributeValue` objects, and return one `AttributeValue` object

Implements `ArcSec::Function`.

10.198.2.3 `static std::string ArcSec::MatchFunction::getFunctionName (std::string datatype) [static]`

help function to get the FunctionName

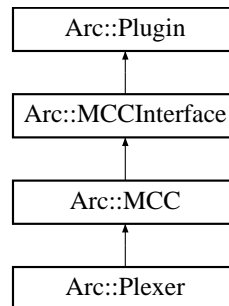
The documentation for this class was generated from the following file:

- `MatchFunction.h`

10.199 Arc::MCC Class Reference

[Message](#) Chain Component - base class for every [MCC](#) plugin.

`#include <MCC.h>`Inheritance diagram for Arc::MCC::



Public Member Functions

- [MCC](#) ([Config](#) *, [PluginArgument](#) *arg)
- virtual void [Next](#) ([MCCInterface](#) *next, const std::string &label="")
- virtual void [AddSecHandler](#) ([Config](#) *cfg, [ArcSec::SecHandler](#) *sechandler, const std::string &label="")
- virtual void [Unlink](#) ()
- virtual [MCC_Status](#) process ([Message](#) &, [Message](#) &)

Protected Member Functions

- [MCCInterface](#) * [Next](#) (const std::string &label="")
- bool [ProcessSecHandlers](#) ([Message](#) &message, const std::string &label="") const

Protected Attributes

- std::map< std::string, [MCCInterface](#) * > [next_](#)
- Glib::Mutex [next_lock_](#)
- std::map< std::string, std::list< [ArcSec::SecHandler](#) * > > [sechandlers_](#)

Static Protected Attributes

- static [Logger](#) [logger](#)

10.199.1 Detailed Description

[Message](#) Chain Component - base class for every [MCC](#) plugin. This is partialy virtual class which defines interface and common functionality for every [MCC](#) plugin needed for managing of component in a chain.

10.199.2 Constructor & Destructor Documentation

10.199.2.1 Arc::MCC::MCC (Config *, PluginArgument * *arg*)

Example constructor - [MCC](#) takes at least it's configuration subtree

10.199.3 Member Function Documentation

10.199.3.1 virtual void Arc::MCC::AddSecHandler (Config * *cfg*, ArcSec::SecHandler * *sechandler*, const std::string & *label* = "") [virtual]

Add security components/handlers to this [MCC](#). Security handlers are stacked into a few queues with each queue identified by its label. The queue labelled 'incoming' is executed for every 'request' message after the message is processed by the [MCC](#) on the service side and before processing on the client side. The queue labelled 'outgoing' is run for response message before it is processed by [MCC](#) algorithms on the service side and after processing on the client side. Those labels are just a matter of agreement and some MCCs may implement different queues executed at various message processing steps.

10.199.3.2 virtual void Arc::MCC::Next (MCCInterface * *next*, const std::string & *label* = "") [virtual]

Add reference to next [MCC](#) in chain. This method is called by [Loader](#) for every potentially labeled link to next component which implements [MCCInterface](#). If next is NULL corresponding link is removed.

Reimplemented in [Arc::Plexer](#).

10.199.3.3 MCCInterface* Arc::MCC::Next (const std::string & *label* = "") [protected]

Returns "next" component associated with provided label.

10.199.3.4 virtual MCC_Status Arc::MCC::process (Message &, Message &) [inline, virtual]

Dummy [Message](#) processing method. Just a placeholder.

Implements [Arc::MCCInterface](#).

Reimplemented in [Arc::Plexer](#).

10.199.3.5 bool Arc::MCC::ProcessSecHandlers (Message & *message*, const std::string & *label* = "") const [protected]

Executes security handlers of specified queue. Returns true if the message is authorized for further processing or if there are no security handlers which implement authorization functionality. This is a convenience method and has to be called by the implementation of the [MCC](#).

10.199.3.6 virtual void Arc::MCC::Unlink () [virtual]

Removing all links. Useful for destroying chains.

10.199.4 Field Documentation

10.199.4.1 Logger `Arc::MCC::logger` `[static, protected]`

A logger for MCCs. A logger intended to be the parent of loggers in the different MCCs.

Reimplemented in [Arc::Plexer](#).

10.199.4.2 `std::map<std::string, MCCInterface *> Arc::MCC::next_` `[protected]`

Set of labeled "next" components. Each implemented [MCC](#) must call `process()` method of corresponding [MCCInterface](#) from this set in own `process()` method.

10.199.4.3 `Glib::Mutex Arc::MCC::next_lock_` `[protected]`

Mutex to protect access to `next_`.

10.199.4.4 `std::map<std::string, std::list<ArcSec::SecHandler *> > Arc::MCC::sechandlers_` `[protected]`

Set of labeled authentication and authorization handlers. [MCC](#) calls sequence of handlers at specific point depending on associated identifier. In most cases those are "in" and "out" for incoming and outgoing messages correspondingly.

The documentation for this class was generated from the following file:

- `MCC.h`

10.200 Arc::MCC_Status Class Reference

A class for communication of [MCC](#) processing results.

```
#include <MCC_Status.h>
```

Public Member Functions

- [MCC_Status](#) ([StatusKind](#) kind=STATUS_UNDEFINED, const std::string &origin="???", const std::string &explanation="No explanation.")
- bool [isOk](#) () const
- [StatusKind](#) [getKind](#) () const
- const std::string & [getOrigin](#) () const
- const std::string & [getExplanation](#) () const
- [operator std::string](#) () const
- [operator bool](#) (void) const
- bool [operator!](#) (void) const

10.200.1 Detailed Description

A class for communication of [MCC](#) processing results. This class is used to communicate result status between MCCs. It contains a status kind, a string specifying the origin ([MCC](#)) of the status object and an explanation.

10.200.2 Constructor & Destructor Documentation

10.200.2.1 Arc::MCC_Status::MCC_Status ([StatusKind](#) *kind* = STATUS_UNDEFINED, const std::string & *origin* = "???", const std::string & *explanation* = "No explanation.")

The constructor. Creates a [MCC_Status](#) object.

Parameters:

- kind* The StatusKind (default: STATUS_UNDEFINED)
origin The origin [MCC](#) (default: "??")
explanation An explanation (default: "No explanation.")

10.200.3 Member Function Documentation

10.200.3.1 const std::string& Arc::MCC_Status::getExplanation () const

Returns an explanation. This method returns an explanation of this object.

Returns:

An explanation of this object.

10.200.3.2 StatusKind Arc::MCC_Status::getKind () const

Returns the status kind. Returns the status kind of this object.

Returns:

The status kind of this object.

10.200.3.3 const std::string& Arc::MCC_Status::getOrigin () const

Returns the origin. This method returns a string specifying the origin [MCC](#) of this object.

Returns:

A string specifying the origin [MCC](#) of this object.

10.200.3.4 bool Arc::MCC_Status::isOk () const

Is the status kind ok? This method returns true if the status kind of this object is STATUS_OK

Returns:

true if kind==STATUS_OK

Referenced by operator bool(), and operator!().

10.200.3.5 Arc::MCC_Status::operator bool (void) const [inline]

Is the status kind ok? This method returns true if the status kind of this object is STATUS_OK

Returns:

true if kind==STATUS_OK

References isOk().

10.200.3.6 Arc::MCC_Status::operator std::string () const

Conversion to string. This operator converts a [MCC_Status](#) object to a string.

10.200.3.7 bool Arc::MCC_Status::operator! (void) const [inline]

not operator Returns true if the status kind is not OK

Returns:

true if kind!=STATUS_OK

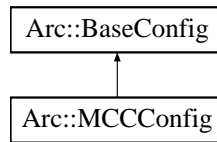
References isOk().

The documentation for this class was generated from the following file:

- [MCC_Status.h](#)

10.201 Arc::MCCConfig Class Reference

Inheritance diagram for Arc::MCCConfig::



Public Member Functions

- virtual [XMLNode MakeConfig](#) ([XMLNode](#) *cfg*) const

10.201.1 Member Function Documentation

10.201.1.1 virtual XMLNode Arc::MCCConfig::MakeConfig (XMLNode *cfg*) const [virtual]

Adds plugin configuration into common configuration tree supplied in 'cfg' argument.

Returns:

reference to XML node representing configuration of [ModuleManager](#)

Reimplemented from [Arc::BaseConfig](#).

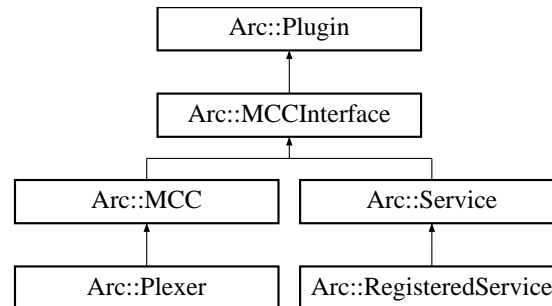
The documentation for this class was generated from the following file:

- MCC.h

10.202 Arc::MCCInterface Class Reference

Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects.

#include <MCC.h> Inheritance diagram for Arc::MCCInterface::



Public Member Functions

- virtual [MCC_Status](#) [process](#) ([Message](#) &request, [Message](#) &response)=0

10.202.1 Detailed Description

Interface for communication between [MCC](#), [Service](#) and [Plexer](#) objects. The Interface consists of the method [process\(\)](#) which is called by the previous [MCC](#) in the chain. For memory management policies please read the description of the [Message](#) class.

10.202.2 Member Function Documentation

10.202.2.1 virtual MCC_Status Arc::MCCInterface::process (Message & request, Message & response) [pure virtual]

Method for processing of requests and responses. This method is called by preceeding [MCC](#) in chain when a request needs to be processed. This method must call similar method of next [MCC](#) in chain unless any failure happens. Result returned by call to next [MCC](#) should be processed and passed back to previous [MCC](#). In case of failure this method is expected to generate valid error response and return it back to previous [MCC](#) without calling the next one.

Parameters:

- request* The request that needs to be processed.
- response* A [Message](#) object that will contain the response of the request when the method returns.

Returns:

- An object representing the status of the call.

Implemented in [Arc::MCC](#), and [Arc::Plexer](#).

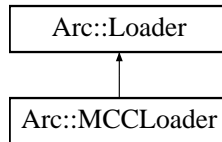
The documentation for this class was generated from the following file:

- MCC.h

10.203 Arc::MCCLoader Class Reference

Creator of [Message](#) Component Chains ([MCC](#)).

#include <MCCLoader.h> Inheritance diagram for Arc::MCCLoader::



Public Member Functions

- [MCCLoader](#) ([Config](#) &cfg)
- [~MCCLoader](#) ()
- [MCC * operator\[\]](#) (const std::string &id)

10.203.1 Detailed Description

Creator of [Message](#) Component Chains ([MCC](#)). This class processes XML configuration and creates message chains. Accepted configuration is defined by XML schema mcc.xsd. Supported components are of types [MCC](#), [Service](#) and [Plexer](#). [MCC](#) and [Service](#) are loaded from dynamic libraries. For [Plexer](#) only internal implementation is supported. This object is also a container for loaded componets. All components and chains are destroyed if this object is destroyed. Chains are created in 2 steps. First all components are loaded and corresponding objects are created. Constructors are supplied with corresponding configuration subtrees. During next step components are linked together by calling their Next() methods. Each call creates labeled link to next component in a chain. 2 step method has an advantage over single step because it allows loops in chains and makes loading procedure more simple. But that also means during short period of time components are only partly configured. Components in such state must produce proper error response if [Message](#) arrives. Note: Current implementation requires all components and links to be labeled. All labels must be unique. Future implementation will be able to assign labels automatically.

10.203.2 Constructor & Destructor Documentation

10.203.2.1 Arc::MCCLoader::MCCLoader (Config & cfg)

Constructor that takes whole XML configuration and creates component chains

10.203.2.2 Arc::MCCLoader::~~MCCLoader ()

Destructor destroys all components created by constructor

10.203.3 Member Function Documentation

10.203.3.1 MCC* Arc::MCCLoader::operator[] (const std::string & id)

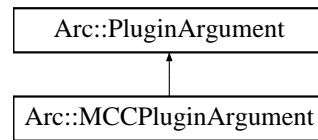
Access entry MCCs in chains. Those are components exposed for external access using 'entry' attribute

The documentation for this class was generated from the following file:

- MCCLoader.h

10.204 Arc::MCCPluginArgument Class Reference

Inheritance diagram for Arc::MCCPluginArgument::



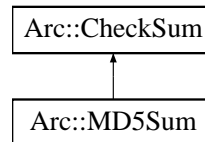
The documentation for this class was generated from the following file:

- MCC.h

10.205 Arc::MD5Sum Class Reference

Implementation of MD5 checksum.

`#include <arc/Checksum.h>`Inheritance diagram for Arc::MD5Sum::



Public Member Functions

- virtual void [start](#) (void)
- virtual void [add](#) (void *buf, unsigned long long int len)
- virtual void [end](#) (void)
- virtual void [result](#) (unsigned char *&res, unsigned int &len) const
- virtual int [print](#) (char *buf, int len) const
- virtual void [scan](#) (const char *buf)
- virtual [operator bool](#) (void) const
- virtual bool [operator!](#) (void) const

10.205.1 Detailed Description

Implementation of MD5 checksum. This class is a specialized class of the [Checksum](#) class. It provides an implementation of the MD5 message-digest algorithm specified in RFC 1321.

10.205.2 Member Function Documentation

10.205.2.1 virtual void Arc::MD5Sum::add (void * *buf*, unsigned long long int *len*) [**virtual**]

Add data to be checksummed. This method calculates the checksum of the passed data chunk, taking into account the previous state of this object.

Parameters:

- buf* pointer to data chunk to be checksummed.
- len* size of the data chunk.

Implements [Arc::Checksum](#).

10.205.2.2 virtual void Arc::MD5Sum::end (void) [**virtual**]

Finalize the checksumming. This method finalizes the checksum algorithm, that is calculating the final checksum result.

Implements [Arc::Checksum](#).

10.205.2.3 virtual int Arc::MD5Sum::print (char * *buf*, int *len*) const [virtual]

Retrieve result of checksum into a string. The passed string *buf* is filled with result of checksum algorithm in base 16. At most *len* characters are filled into buffer *buf*. The hexadecimal value is prepended with "algorithm:", where algorithm is one of "cksum", "md5" or "adler32" respectively corresponding to the result from the [CRC32Sum](#), [MD5Sum](#) and Adler32 classes.

Parameters:

buf pointer to buffer which should be filled with checksum result.
len max number of character filled into buffer.

Returns:

0 on success

Reimplemented from [Arc::Checksum](#).

10.205.2.4 virtual void Arc::MD5Sum::scan (const char * *buf*) [virtual]

Set internal checksum state. This method sets the internal state to that of the passed textual representation. The format passed to this method must be the same as retrieved from the [Checksum::print](#) method.

Parameters:

buf string containing textual representation of checksum

See also:

[Checksum::print](#)

Implements [Arc::Checksum](#).

10.205.2.5 virtual void Arc::MD5Sum::start (void) [virtual]

Initiate the checksum algorithm. This method must be called before starting a new checksum calculation.

Implements [Arc::Checksum](#).

The documentation for this class was generated from the following file:

- CheckSum.h

10.206 Arc::Message Class Reference

Object being passed through chain of MCCs.

```
#include <Message.h>
```

Public Member Functions

- [Message](#) (void)
- [Message](#) ([Message](#) &msg)
- [Message](#) (long msg_ptr_addr)
- [~Message](#) (void)
- [Message](#) & operator= ([Message](#) &msg)
- [MessagePayload](#) * [Payload](#) (void)
- [MessagePayload](#) * [Payload](#) ([MessagePayload](#) *payload)
- [MessageAttributes](#) * [Attributes](#) (void)
- [MessageAuth](#) * [Auth](#) (void)
- [MessageContext](#) * [Context](#) (void)
- [MessageAuthContext](#) * [AuthContext](#) (void)
- void [Context](#) ([MessageContext](#) *ctx)
- void [AuthContext](#) ([MessageAuthContext](#) *auth_ctx)

10.206.1 Detailed Description

Object being passed through chain of MCCs. An instance of this class refers to objects with main content ([MessagePayload](#)), authentication/authorization information ([MessageAuth](#)) and common purpose attributes ([MessageAttributes](#)). [Message](#) class does not manage pointers to objects and their content. It only serves for grouping those objects. [Message](#) objects are supposed to be processed by MCCs and Services implementing [MCCInterface](#) method process(). All objects constituting content of [Message](#) object are subject to following policies:

1. All objects created inside call to process() method using new command must be explicitly destroyed within same call using delete command with following exceptions. a) Objects which are assigned to 'response' [Message](#). b) Objects whose management is completely acquired by objects assigned to 'response' [Message](#).
2. All objects not created inside call to process() method are not explicitly destroyed within that call with following exception. a) Objects which are part of 'response' Method returned from call to next's process() method. Unless those objects are passed further to calling process(), of course.
3. It is not allowed to make 'response' point to same objects as 'request' does on entry to process() method. That is needed to avoid double destruction of same object. (Note: if in a future such need arises it may be solved by storing additional flags in [Message](#) object).
4. It is allowed to change content of pointers of 'request' [Message](#). Calling process() method must not rely on that object to stay intact.
5. Called process() method should either fill 'response' [Message](#) with pointers to valid objects or to keep them intact. This makes it possible for calling process() to preload 'response' with valid error message.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 Arc::Message::Message (void) [inline]

true if `auth_ctx_` was created internally Dummy constructor

10.206.2.2 Arc::Message::Message (Message & *msg*) [inline]

Copy constructor. Ensures shallow copy.

10.206.2.3 Arc::Message::Message (long *msg_ptr_addr*)

Copy constructor. Used by language bindings

10.206.2.4 Arc::Message::~~Message (void) [inline]

Destructor does not affect referred objects except those created internally

10.206.3 Member Function Documentation

10.206.3.1 MessageAttributes* Arc::Message::Attributes (void) [inline]

Returns a pointer to the current attributes object or creates it if no attributes object has been assigned.

Referenced by `operator=()`.

10.206.3.2 MessageAuth* Arc::Message::Auth (void) [inline]

Returns a pointer to the current authentication/authorization object or creates it if no object has been assigned.

Referenced by `operator=()`.

10.206.3.3 void Arc::Message::AuthContext (MessageAuthContext * *auth_ctx*) [inline]

Assigns `auth*` context object

10.206.3.4 MessageAuthContext* Arc::Message::AuthContext (void) [inline]

Returns a pointer to the current `auth*` context object or creates it if no object has been assigned.

Referenced by `operator=()`.

10.206.3.5 void Arc::Message::Context (MessageContext * *ctx*) [inline]

Assigns message context object

10.206.3.6 MessageContext* Arc::Message::Context (void) [inline]

Returns a pointer to the current context object or creates it if no object has been assigned. Last case should happen only if first [MCC](#) in a chain is connectionless like one implementing UDP protocol.

Referenced by operator=().

10.206.3.7 Message& Arc::Message::operator= (Message & msg) [inline]

Assignment. Ensures shallow copy.

References Attributes(), Auth(), AuthContext(), and Context().

10.206.3.8 MessagePayload* Arc::Message::Payload (MessagePayload * payload) [inline]

Replaces payload with new one. Returns the old one.

10.206.3.9 MessagePayload* Arc::Message::Payload (void) [inline]

Returns pointer to current payload or NULL if no payload assigned.

The documentation for this class was generated from the following file:

- Message.h

10.207 Arc::MessageAttributes Class Reference

A class for storage of attribute values.

```
#include <MessageAttributes.h>
```

Public Member Functions

- [MessageAttributes](#) ()
- void [set](#) (const std::string &key, const std::string &value)
- void [add](#) (const std::string &key, const std::string &value)
- void [removeAll](#) (const std::string &key)
- void [remove](#) (const std::string &key, const std::string &value)
- int [count](#) (const std::string &key) const
- const std::string & [get](#) (const std::string &key) const
- [AttributeIterator](#) [getAll](#) (const std::string &key) const
- [AttributeIterator](#) [getAll](#) (void) const

Protected Attributes

- [AttrMap](#) [attributes_](#)

10.207.1 Detailed Description

A class for storage of attribute values. This class is used to store attributes of messages. All attribute keys and their corresponding values are stored as strings. Any key or value that is not a string must thus be represented as a string during storage. Furthermore, an attribute is usually a key-value pair with a unique key, but there may also be multiple such pairs with equal keys.

The key of an attribute is composed by the name of the [Message](#) Chain Component ([MCC](#)) which produce it and the name of the attribute itself with a colon (:) in between, i.e. MCC_Name:Attribute_Name. For example, the key of the "Content-Length" attribute of the HTTP [MCC](#) is thus "HTTP:Content-Length".

There are also "global attributes", which may be produced by different MCCs depending on the configuration. The keys of such attributes are NOT prefixed by the name of the producing [MCC](#). Before any new global attribute is introduced, it must be agreed upon by the core development team and added below. The global attributes decided so far are:

- Request-URI Identifies the service to which the message shall be sent. This attribute is produced by e.g. the HTTP [MCC](#) and used by the plexer for routing the message to the appropriate service.

10.207.2 Constructor & Destructor Documentation

10.207.2.1 Arc::MessageAttributes::MessageAttributes ()

The default constructor. This is the default constructor of the [MessageAttributes](#) class. It constructs an empty object that initially contains no attributes.

10.207.3 Member Function Documentation

10.207.3.1 `void Arc::MessageAttributes::add (const std::string & key, const std::string & value)`

Adds a value to an attribute. This method adds a new value to an attribute. Any previous value will be preserved, i.e. the attribute may become multiple valued.

Parameters:

key The key of the attribute.

value The (new) value of the attribute.

10.207.3.2 `int Arc::MessageAttributes::count (const std::string & key) const`

Returns the number of values of an attribute. Returns the number of values of an attribute that matches a certain key.

Parameters:

key The key of the attribute for which to count values.

Returns:

The number of values that corresponds to the key.

10.207.3.3 `const std::string& Arc::MessageAttributes::get (const std::string & key) const`

Returns the value of a single-valued attribute. This method returns the value of a single-valued attribute. If the attribute is not single valued (i.e. there is no such attribute or it is a multiple-valued attribute) an empty string is returned.

Parameters:

key The key of the attribute for which to return the value.

Returns:

The value of the attribute.

10.207.3.4 `AttributeIterator Arc::MessageAttributes::getAll (const std::string & key) const`

Access the value(s) of an attribute. This method returns an [AttributeIterator](#) that can be used to access the values of an attribute.

Parameters:

key The key of the attribute for which to return the values.

Returns:

An [AttributeIterator](#) for access of the values of the attribute.

10.207.3.5 void Arc::MessageAttributes::remove (const std::string & *key*, const std::string & *value*)

Removes one value of an attribute. This method removes a certain value from the attribute that matches a certain key.

Parameters:

key The key of the attribute from which the value shall be removed.

value The value to remove.

10.207.3.6 void Arc::MessageAttributes::removeAll (const std::string & *key*)

Removes all attributes with a certain key. This method removes all attributes that match a certain key.

Parameters:

key The key of the attributes to remove.

10.207.3.7 void Arc::MessageAttributes::set (const std::string & *key*, const std::string & *value*)

Sets a unique value of an attribute. This method removes any previous value of an attribute and sets the new value as the only value.

Parameters:

key The key of the attribute.

value The (new) value of the attribute.

10.207.4 Field Documentation**10.207.4.1 AttrMap Arc::MessageAttributes::attributes_ [protected]**

Internal storage of attributes. An AttrMap (multimap) in which all attributes (key-value pairs) are stored.

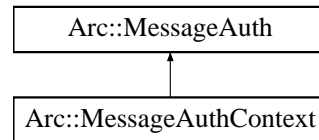
The documentation for this class was generated from the following file:

- MessageAttributes.h

10.208 Arc::MessageAuth Class Reference

Contains authenticity information, authorization tokens and decisions.

#include <MessageAuth.h> Inheritance diagram for Arc::MessageAuth::



Public Member Functions

- void [set](#) (const std::string &key, [SecAttr](#) *value)
- void [remove](#) (const std::string &key)
- [SecAttr](#) * [get](#) (const std::string &key)
- [SecAttr](#) * [operator\[\]](#) (const std::string &key)
- bool [Export](#) ([SecAttrFormat](#) format, [XMLNode](#) &val) const
- [MessageAuth](#) * [Filter](#) (const std::list< std::string > &selected_keys, const std::list< std::string > &rejected_keys)

10.208.1 Detailed Description

Contains authenticity information, authorization tokens and decisions. This class only supports string keys and [SecAttr](#) values.

10.208.2 Member Function Documentation

10.208.2.1 bool Arc::MessageAuth::Export (SecAttrFormat *format*, XMLNode & *val*) const

Returns properly catenated attributes in specified format. Content of XML node at is replaced with generated information if XML tree is empty. If tree at is not empty then [Export\(\)](#) tries to merge generated information to already existing like everything would be generated inside same [Export\(\)](#) method. If does not represent valid node then new XML tree is created.

10.208.2.2 MessageAuth* Arc::MessageAuth::Filter (const std::list< std::string > & *selected_keys*, const std::list< std::string > & *rejected_keys*)

Creates new instance of [MessageAuth](#) with attributes filtered. In new instance all attributes with keys listed in are removed. If is not empty only corresponding attributes are transferred to new instance. Created instance does not own referred attributes. Hence parent instance must not be deleted as long as this one is in use.

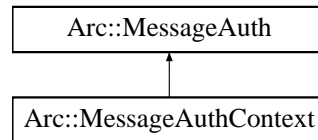
The documentation for this class was generated from the following file:

- MessageAuth.h

10.209 Arc::MessageAuthContext Class Reference

Handler for content of message auth* context.

#include <Message.h>Inheritance diagram for Arc::MessageAuthContext::



10.209.1 Detailed Description

Handler for content of message auth* context. This class is a container for authorization and authentication information. It gets associated with [Message](#) object usually by first [MCC](#) in a chain and is kept as long as connection persists.

The documentation for this class was generated from the following file:

- Message.h

10.210 Arc::MessageContext Class Reference

Handler for content of message context.

```
#include <Message.h>
```

Public Member Functions

- void [Add](#) (const std::string &name, [MessageContextElement](#) *element)

10.210.1 Detailed Description

Handler for content of message context. This class is a container for objects derived from [MessageContextElement](#). It gets associated with [Message](#) object usually by first [MCC](#) in a chain and is kept as long as connection persists.

10.210.2 Member Function Documentation

10.210.2.1 void Arc::MessageContext::Add (const std::string & *name*, [MessageContextElement](#) * *element*)

Provided element is taken over by this class. It is remembered by it and destroyed when this class is destroyed.

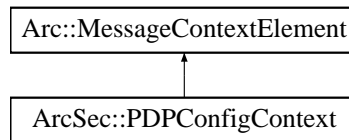
The documentation for this class was generated from the following file:

- Message.h

10.211 Arc::MessageContextElement Class Reference

Top class for elements contained in message context.

#include <Message.h>Inheritance diagram for Arc::MessageContextElement::



10.211.1 Detailed Description

Top class for elements contained in message context. Objects of classes inherited with this one may be stored in [MessageContext](#) container.

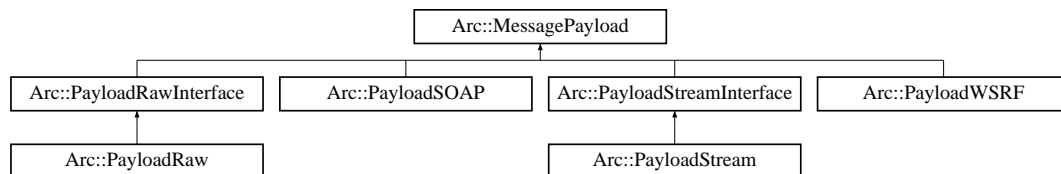
The documentation for this class was generated from the following file:

- Message.h

10.212 Arc::MessagePayload Class Reference

Base class for content of message passed through chain.

`#include <Message.h>`Inheritance diagram for Arc::MessagePayload::



Public Member Functions

- [MCC_Status Failure](#) (void)

10.212.1 Detailed Description

Base class for content of message passed through chain. It's not intended to be used directly. Instead functional classes must be derived from it.

The documentation for this class was generated from the following file:

- Message.h

10.213 Arc::PluginsFactory::modules_t::miterator Class Reference

The documentation for this class was generated from the following file:

- Plugin.h

10.214 Arc::ModuleDesc Class Reference

Description of loadable module.

```
#include <Plugin.h>
```

10.214.1 Detailed Description

Description of loadable module. This class is used for reports

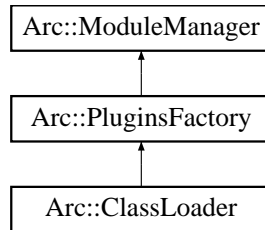
The documentation for this class was generated from the following file:

- Plugin.h

10.215 Arc::ModuleManager Class Reference

Manager of shared libraries.

#include <ModuleManager.h> Inheritance diagram for Arc::ModuleManager::



Data Structures

- class **LoadableModuleDescription**

Public Member Functions

- **ModuleManager** (**XMLNode** cfg)
- **Glib::Module *** **load** (const std::string &name, bool probe)
- std::string **find** (const std::string &name)
- **Glib::Module *** **reload** (**Glib::Module ***module)
- void **use** (**Glib::Module ***module)
- void **unuse** (**Glib::Module ***module)
- std::string **findLocation** (const std::string &name)
- bool **makePersistent** (**Glib::Module ***module)
- bool **makePersistent** (const std::string &name)
- void **setCfg** (**XMLNode** cfg)

Protected Member Functions

- void **unload** (**Glib::Module ***module)
- void **unload** (const std::string &name)

10.215.1 Detailed Description

Manager of shared libraries. This class loads shared libraries/modules. There supposed to be created one instance of it per executable. In such circumstances it would cache handles to loaded modules and not load them multiple times. But creating multiple instances is not prohibited. Instance of this class handles loading of shared libraries through call to **load()** method. All loaded libraries are remembered internally and by default are unloaded when instance of this class is destroyed. Sometimes it is not safe to unload library. In such case **makePersistent()** for this library must be called. Upon first **load()** of library **ModuleManager** looks for function called `__arc_module_constructor__` and calls it. This makes it possible for library to do some preparations. Currently it is used to make some libraries persistent in memory. Before unloading library from memory `__arc_module_destructor__` is called if present. Every loaded library has load counter associated. Each call to **load()** for specific library increases that counter and **unload()** decreases it. Library is unloaded when counter reaches zero. When instance of **ModuleManager** is destroyed all load counters

are reset to 0 and libraries are unloaded unless claimed to stay persistent in memory. Each library also has usage counter associated. Those counters are increased and decreased by `use()` and `unuse()` methods. This counter is used to claim usage of code provided by loaded library. It is automatically increased and decreased in constructor and destructor of `Plugin` class. Having non-zero usage counter prevents library from being unloaded. Please note that destructor of `ModuleManager` waits for all usage counters to reach zero. This is especially useful in multithreaded environments. To avoid dealocks make sure Plugins loaded by instance of `ModuleManager` are destroyed before destroying `ModuleManager` or in independent threads.

10.215.2 Constructor & Destructor Documentation

10.215.2.1 `Arc::ModuleManager::ModuleManager (XMLNode cfg)`

Constructor. It is supposed to process corresponding configuration subtree and tune module loading parameters accordingly.

10.215.3 Member Function Documentation

10.215.3.1 `std::string Arc::ModuleManager::find (const std::string & name)`

Finds loadable module by 'name' looking in same places as `load()` does, but does not load it.

10.215.3.2 `std::string Arc::ModuleManager::findLocation (const std::string & name)`

Finds shared library corresponding to module 'name' and returns path to it

10.215.3.3 `Glib::Module* Arc::ModuleManager::load (const std::string & name, bool probe)`

Finds module 'name' in cache or loads corresponding loadable module

10.215.3.4 `bool Arc::ModuleManager::makePersistent (const std::string & name)`

Make sure this module is never unloaded. Even if `unload()` is called.

10.215.3.5 `bool Arc::ModuleManager::makePersistent (Glib::Module * module)`

Make sure this module is never unloaded. Even if `unload()` is called. Call to this method does not affect how other methods are behaving. Just loaded module stays in memory after all unloading procedures.

10.215.3.6 `Glib::Module* Arc::ModuleManager::reload (Glib::Module * module)`

Reload module previously loaded in probe mode. New module is loaded with all symbols resolved and old module handler is unloaded. In case of error old module is not unloaded.

10.215.3.7 `void Arc::ModuleManager::setCfg (XMLNode cfg)`

Input the configuration subtree, and trigger the module loading (do almost the same as the Constructor). This method is designed for `ClassLoader` to adopt the singleton pattern.

10.215.3.8 void Arc::ModuleManager::unload (const std::string & *name*) [protected]

Unload module by its name

10.215.3.9 void Arc::ModuleManager::unload (Glib::Module * *module*) [protected]

Unload module by its identifier. Decreases load counter and unloads module when it reaches 0.

10.215.3.10 void Arc::ModuleManager::unuse (Glib::Module * *module*)

Decrease usage count till it reaches 0. This call does not unload module. Usage counter is only for preventing unexpected unload. Unloading is done by [unload\(\)](#) methods and by desctructor if usage counter is zero.

10.215.3.11 void Arc::ModuleManager::use (Glib::Module * *module*)

Increase usage count of loaded module. It is intended to be called by plugins or other code which needs prevent module to be unloaded while its code is running. Must be accompanied by unuse when module is not needed.

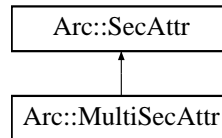
The documentation for this class was generated from the following file:

- ModuleManager.h

10.216 Arc::MultiSecAttr Class Reference

Container of multiple [SecAttr](#) attributes.

`#include <SecAttr.h>`Inheritance diagram for Arc::MultiSecAttr:



Public Member Functions

- virtual [operator bool](#) () const
- virtual bool [Export](#) ([SecAttrFormat](#) format, [XMLNode](#) &val) const

10.216.1 Detailed Description

Container of multiple [SecAttr](#) attributes. This class combines multiple attributes. It's export/import methods catenate results of underlying objects. Primary meaning of this class is to serve as base for classes implementing multi level hierarchical tree-like descriptions of user identity. It may also be used for collecting information of same source or kind. Like all information extracted from X509 certificate.

10.216.2 Member Function Documentation

10.216.2.1 virtual bool Arc::MultiSecAttr::Export (SecAttrFormat *format*, XMLNode & *val*) const [virtual]

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute. XML node referenced by is turned into top level element of specified format.

Reimplemented from [Arc::SecAttr](#).

10.216.2.2 virtual Arc::MultiSecAttr::operator bool () const [virtual]

This function should return false if the value is to be considered null, e.g. if it hasn't been set or initialized. In other cases it should return true.

Reimplemented from [Arc::SecAttr](#).

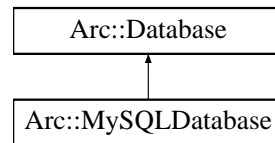
The documentation for this class was generated from the following file:

- SecAttr.h

10.217 Arc::MySQLDatabase Class Reference

Implements a MySQL version of the [Database](#) interface.

#include <arc/MySQLWrapper.h> Inheritance diagram for Arc::MySQLDatabase:



Public Member Functions

- virtual bool [connect](#) (std::string &dbname, std::string &user, std::string &password)
- virtual bool [isconnected](#) () const
- virtual void [close](#) ()
- virtual bool [enable_ssl](#) (const std::string &keyfile="", const std::string &certfile="", const std::string &cafile="", const std::string &capath="")
- virtual bool [shutdown](#) ()

10.217.1 Detailed Description

Implements a MySQL version of the [Database](#) interface.

10.217.2 Member Function Documentation

10.217.2.1 virtual bool Arc::MySQLDatabase::connect (std::string & *dbname*, std::string & *user*, std::string & *password*) [**virtual**]

Do connection with database server.

Parameters:

- dbname* The database name which will be used.
- user* The username which will be used to access database.
- password* The password which will be used to access database.

Implements [Arc::Database](#).

10.217.2.2 virtual bool Arc::MySQLDatabase::enable_ssl (const std::string & *keyfile* = "", const std::string & *certfile* = "", const std::string & *cafile* = "", const std::string & *capath* = "") [**virtual**]

Enable ssl communication for the connection.

Parameters:

- keyfile* The location of key file.
- certfile* The location of certificate file.

cafile The location of ca file.

capath The location of ca directory

Implements [Arc::Database](#).

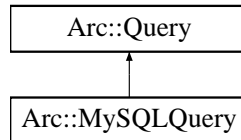
The documentation for this class was generated from the following file:

- MysqlWrapper.h

10.218 Arc::MySQLQuery Class Reference

Implements a MySQL version of the [Query](#) database query class.

#include <arc/MySQLWrapper.h> Inheritance diagram for Arc::MySQLQuery:



Public Member Functions

- virtual int [get_num_columns](#) ()
- virtual int [get_num_rows](#) ()
- virtual bool [execute](#) (const std::string &sqlstr)
- virtual QueryRowResult [get_row](#) (int row_number) const
- virtual QueryRowResult [get_row](#) () const
- virtual std::string [get_row_field](#) (int row_number, std::string &field_name)
- virtual bool [get_array](#) (std::string &sqlstr, QueryArrayResult &result, std::vector< std::string > &arguments)

10.218.1 Detailed Description

Implements a MySQL version of the [Query](#) database query class.

10.218.2 Member Function Documentation

10.218.2.1 virtual bool Arc::MySQLQuery::execute (const std::string &sqlstr) [virtual]

Execute the query.

Parameters:

sqlstr The sql sentence used to query

Implements [Arc::Query](#).

10.218.2.2 virtual bool Arc::MySQLQuery::get_array (std::string &sqlstr, QueryArrayResult &result, std::vector< std::string > &arguments) [virtual]

[Query](#) the database by using some parameters into sql sentence. An example sentence: "select table.value from table where table.name = ?"

Parameters:

sqlstr The sql sentence with some parameters marked with "?".

result The result in an array which includes all of the value in query result.

arguments The argument list which should exactly correspond with the parameters in the sql sentence.

Implements [Arc::Query](#).

10.218.2.3 virtual QueryRowResult Arc::MySQLQuery::get_row () const [virtual]

Get the value of one row in the query result. The row number will be automatically increased each time the method is called.

Implements [Arc::Query](#).

10.218.2.4 virtual QueryRowResult Arc::MySQLQuery::get_row (int *row_number*) const [virtual]

Get the value of one row in the query result.

Parameters:

row_number The number of the row

Returns:

A vector includes all the values in the row

Implements [Arc::Query](#).

10.218.2.5 virtual std::string Arc::MySQLQuery::get_row_field (int *row_number*, std::string & *field_name*) [virtual]

Get the value of one specific field in one specific row.

Parameters:

row_number The row number inside the query result

field_name The field name for the value which will be return

Returns:

The value of the specified filed in the specified row

Implements [Arc::Query](#).

The documentation for this class was generated from the following file:

- MysqlWrapper.h

10.219 Arc::NotificationType Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.220 Arc::NS Class Reference

Class to represent an XML namespace.

```
#include <arc/XMLNode.h>
```

Public Member Functions

- [NS](#) (void)
- [NS](#) (const char *prefix, const char *uri)
- [NS](#) (const char *nslist[][2])
- [NS](#) (const std::map< std::string, std::string > &nslist)

10.220.1 Detailed Description

Class to represent an XML namespace.

10.220.2 Constructor & Destructor Documentation

10.220.2.1 Arc::NS::NS (const char * nslist[][2]) [inline]

Constructor creates namespace with multiple entries.

Parameters:

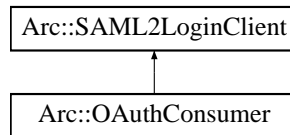
nslist Array made of prefix and URI pairs and must be NULL terminated

The documentation for this class was generated from the following file:

- XMLNode.h

10.221 Arc::OAuthConsumer Class Reference

#include <OAuthConsumer.h> Inheritance diagram for Arc::OAuthConsumer::



Public Member Functions

- [OAuthConsumer](#) (const [MCCConfig](#) cfg, const [URL](#) url, std::list< std::string > idp_stack)
- [MCC_Status parseDN](#) (std::string *dn)
- [MCC_Status approveCSR](#) (const std::string approve_page)
- [MCC_Status pushCSR](#) (const std::string b64_pub_key, const std::string pub_key_hash, std::string *approve_page)
- [MCC_Status storeCert](#) (const std::string cert_path, const std::string auth_token, const std::string b64_dn)

Protected Member Functions

- [MCC_Status processLogin](#) (const std::string username="", const std::string password="")

10.221.1 Detailed Description

The OAuth functionality depends on the availability of the liboauth C-bindings library

10.221.2 Constructor & Destructor Documentation

10.221.2.1 Arc::OAuthConsumer::OAuthConsumer (const [MCCConfig](#) *cfg*, const [URL](#) *url*, std::list< std::string > *idp_stack*)

Construct an OAuth consumer with url as service provider. idp_name is currently ignored, since the idp to which the SAML2 redirect will take place is presently a hardcoded value on the SAML2 SP side. This is expected to change in the future.

10.221.3 Member Function Documentation

10.221.3.1 [MCC_Status](#) Arc::OAuthConsumer::approveCSR (const std::string *approve_page*) [virtual]

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

10.221.3.2 `MCC_Status Arc::OAuthConsumer::parseDN (std::string * dn) [virtual]`

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

10.221.3.3 `MCC_Status Arc::OAuthConsumer::processLogin (const std::string username = "", const std::string password = "") [protected, virtual]`

Main function performing all the OAuth login steps. Username and password will be ignored.

Implements [Arc::SAML2LoginClient](#).

10.221.3.4 `MCC_Status Arc::OAuthConsumer::pushCSR (const std::string b64_pub_key, const std::string pub_key_hash, std::string * approve_page) [virtual]`

Unsupported placeholder function until Confusa supports OAuth.

Implements [Arc::SAML2LoginClient](#).

10.221.3.5 `MCC_Status Arc::OAuthConsumer::storeCert (const std::string cert_path, const std::string auth_token, const std::string b64_dn) [virtual]`

Unsupported placeholder function until Confusa supports OAuth.

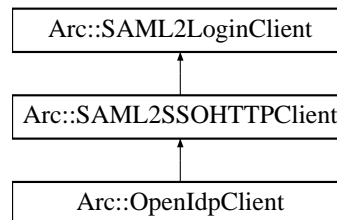
Implements [Arc::SAML2LoginClient](#).

The documentation for this class was generated from the following file:

- OAuthConsumer.h

10.222 Arc::OpenIdpClient Class Reference

Inheritance diagram for Arc::OpenIdpClient::



Protected Member Functions

- [MCC_Status processIdPLogin](#) (const std::string username, const std::string password)
- [MCC_Status processConsent](#) ()
- [MCC_Status processIdP2Confusa](#) ()

10.222.1 Member Function Documentation

10.222.1.1 MCC_Status Arc::OpenIdpClient::processConsent () [protected, virtual]

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implements [Arc::SAML2SSOHTTPClient](#).

10.222.1.2 MCC_Status Arc::OpenIdpClient::processIdP2Confusa () [protected, virtual]

Redirects the user back from identity provider to the Confusa SP

Implements [Arc::SAML2SSOHTTPClient](#).

10.222.1.3 MCC_Status Arc::OpenIdpClient::processIdPLogin (const std::string *username*, const std::string *password*) [protected, virtual]

Actual identity provider parsers for next three methods implemented in subdirectory idp/

Parse identity provider login page and submit username and password in the previsioned way

Implements [Arc::SAML2SSOHTTPClient](#).

The documentation for this class was generated from the following file:

- OpenIdpClient.h

10.223 Arc::OptIn< T > Class Template Reference

`template<class T> class Arc::OptIn< T >`

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.224 Arc::OptionParser Class Reference

Command line option parser used by ARC command line tools.

```
#include <arc/OptionParser.h>
```

Public Member Functions

- [OptionParser](#) (const std::string &arguments="", const std::string &summary="", const std::string &description="")
- void [AddOption](#) (const char shortOpt, const std::string &longOpt, const std::string &optDesc, bool &val)
- void [AddOption](#) (const char shortOpt, const std::string &longOpt, const std::string &optDesc, const std::string &argDesc, int &val)
- void [AddOption](#) (const char shortOpt, const std::string &longOpt, const std::string &optDesc, const std::string &argDesc, std::string &val)
- void [AddOption](#) (const char shortOpt, const std::string &longOpt, const std::string &optDesc, const std::string &argDesc, std::list< std::string > &val)
- std::list< std::string > [Parse](#) (int argc, char **argv)

10.224.1 Detailed Description

Command line option parser used by ARC command line tools. The command line arguments and a brief and detailed description can be set in the constructor. Each command line option should be added with an [AddOption\(\)](#) method, corresponding to the type of the option. [Parse\(\)](#) can then be called with the same arguments as `main()` takes. It returns a list of arguments and fills each "val" passed in [AddOption\(\)](#) if the corresponding option is specified on the command line.

A help text is automatically generated and displayed on stdout if a help option (-h or -?) is used on the command line. Note that [Parse\(\)](#) calls `exit(0)` after displaying the help text.

Both short and long format options are supported.

10.224.2 Constructor & Destructor Documentation

10.224.2.1 Arc::OptionParser::OptionParser (const std::string &arguments = "", const std::string &summary = "", const std::string &description = "")

Create a new [OptionParser](#).

Parameters:

arguments Command line arguments
summary Brief summary of command
description Detailed description of command

10.224.3 Member Function Documentation

10.224.3.1 void Arc::OptionParser::AddOption (const char shortOpt, const std::string &longOpt, const std::string &optDesc, const std::string &argDesc, std::list< std::string > &val)

Add an option which takes a string argument and can be specified multiple times.

Parameters:

shortOpt Short version of this option
longOpt Long version of this option
optDesc Description of option
argDesc Value of option argument
val Value filled during [Parse\(\)](#)

10.224.3.2 void Arc::OptionParser::AddOption (const char *shortOpt*, const std::string & *longOpt*, const std::string & *optDesc*, const std::string & *argDesc*, std::string & *val*)

Add an option which takes a string argument.

Parameters:

shortOpt Short version of this option
longOpt Long version of this option
optDesc Description of option
argDesc Value of option argument
val Value filled during [Parse\(\)](#)

10.224.3.3 void Arc::OptionParser::AddOption (const char *shortOpt*, const std::string & *longOpt*, const std::string & *optDesc*, const std::string & *argDesc*, int & *val*)

Add an option which takes an integer argument.

Parameters:

shortOpt Short version of this option
longOpt Long version of this option
optDesc Description of option
argDesc Value of option argument
val Value filled during [Parse\(\)](#)

10.224.3.4 void Arc::OptionParser::AddOption (const char *shortOpt*, const std::string & *longOpt*, const std::string & *optDesc*, bool & *val*)

Add an option which does not take any arguments.

Parameters:

shortOpt Short version of this option
longOpt Long version of this option
optDesc Description of option
val Value filled during [Parse\(\)](#)

10.224.3.5 `std::list<std::string> Arc::OptionParser::Parse (int argc, char ** argv)`

Parse the options and arguments. Should be called after all options have been added with [AddOption\(\)](#). The parameters can be the same as those taken by `main()`. Note that if a help option is given this method calls `exit(0)` after printing help text to `stdout`.

Returns:

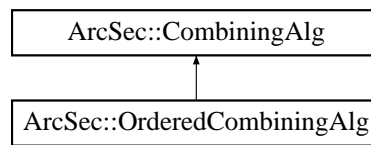
The list of command line arguments

The documentation for this class was generated from the following file:

- OptionParser.h

10.225 ArcSec::OrderedCombiningAlg Class Reference

Inheritance diagram for ArcSec::OrderedCombiningAlg::



The documentation for this class was generated from the following file:

- OrderedAlg.h

10.226 Arc::OutputFileType Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.227 **Arc::ParallelEnvironmentType Class Reference**

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.228 Arc::PathIterator Class Reference

Class to iterate through elements of a path.

```
#include <URL.h>
```

Public Member Functions

- [PathIterator](#) (const std::string &path, bool end=false)
- [PathIterator](#) & [operator++](#) ()
- [PathIterator](#) & [operator--](#) ()
- [operator bool](#) () const
- std::string [operator*](#) () const
- std::string [Rest](#) () const

10.228.1 Detailed Description

Class to iterate through elements of a path.

10.228.2 Constructor & Destructor Documentation

10.228.2.1 Arc::PathIterator::PathIterator (const std::string & *path*, bool *end* = *false*)

Constructor accepts path and stores it internally. If end is set to false iterator points at first element in path. Otherwise selected element is one before last.

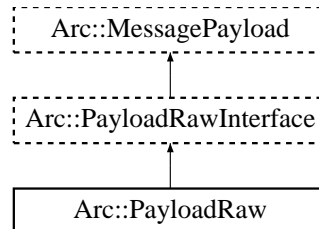
The documentation for this class was generated from the following file:

- URL.h

10.229 Arc::PayloadRaw Class Reference

Raw byte multi-buffer.

#include <PayloadRaw.h> Inheritance diagram for Arc::PayloadRaw::



Public Member Functions

- [PayloadRaw](#) (void)
- virtual [~PayloadRaw](#) (void)
- virtual char [operator\[\]](#) (Size_t pos) const
- virtual char * [Content](#) (Size_t pos=-1)
- virtual Size_t [Size](#) (void) const
- virtual char * [Insert](#) (Size_t pos=0, Size_t size=0)
- virtual char * [Insert](#) (const char *s, Size_t pos=0, Size_t size=-1)
- virtual char * [Buffer](#) (unsigned int num=0)
- virtual Size_t [BufferSize](#) (unsigned int num=0) const
- virtual Size_t [BufferPos](#) (unsigned int num=0) const
- virtual bool [Truncate](#) (Size_t size)

10.229.1 Detailed Description

Raw byte multi-buffer. This is implementation of [PayloadRawInterface](#). Buffers are memory blocks logically placed one after another.

10.229.2 Constructor & Destructor Documentation

10.229.2.1 Arc::PayloadRaw::PayloadRaw (void) [inline]

List of handled buffers. Constructor. Created object contains no buffers.

10.229.2.2 virtual Arc::PayloadRaw::~~PayloadRaw (void) [virtual]

Destructor. Frees allocated buffers.

10.229.3 Member Function Documentation

10.229.3.1 virtual char* Arc::PayloadRaw::Buffer (unsigned int num = 0) [virtual]

Returns pointer to num'th buffer

Implements [Arc::PayloadRawInterface](#).

10.229.3.2 virtual Size_t Arc::PayloadRaw::BufferPos (unsigned int *num* = 0) const [virtual]

Returns position of *num*'th buffer

Implements [Arc::PayloadRawInterface](#).

10.229.3.3 virtual Size_t Arc::PayloadRaw::BufferSize (unsigned int *num* = 0) const [virtual]

Returns length of *num*'th buffer

Implements [Arc::PayloadRawInterface](#).

10.229.3.4 virtual char* Arc::PayloadRaw::Content (Size_t *pos* = -1) [virtual]

Get pointer to buffer content at global position '*pos*'. By default to beginning of main buffer whatever that means.

Implements [Arc::PayloadRawInterface](#).

10.229.3.5 virtual char* Arc::PayloadRaw::Insert (const char * *s*, Size_t *pos* = 0, Size_t *size* = -1) [virtual]

Create new buffer at global position '*pos*' of size '*size*'. Created buffer is filled with content of memory at '*s*'. If '*size*' is negative content at '*s*' is expected to be null-terminated.

Implements [Arc::PayloadRawInterface](#).

10.229.3.6 virtual char* Arc::PayloadRaw::Insert (Size_t *pos* = 0, Size_t *size* = 0) [virtual]

Create new buffer at global position '*pos*' of size '*size*'.

Implements [Arc::PayloadRawInterface](#).

10.229.3.7 virtual char Arc::PayloadRaw::operator[] (Size_t *pos*) const [virtual]

Returns content of byte at specified position. Specified position '*pos*' is treated as global one and goes through all buffers placed one after another.

Implements [Arc::PayloadRawInterface](#).

10.229.3.8 virtual Size_t Arc::PayloadRaw::Size (void) const [virtual]

Returns logical size of whole structure.

Implements [Arc::PayloadRawInterface](#).

10.229.3.9 virtual bool Arc::PayloadRaw::Truncate (Size_t size) [virtual]

Change size of stored information. If size exceeds end of allocated buffer, buffers are not re-allocated, only logical size is extended. Buffers with location behind new size are deallocated.

Implements [Arc::PayloadRawInterface](#).

The documentation for this class was generated from the following file:

- PayloadRaw.h

10.230 Arc::PayloadRawBuf Struct Reference

Data Fields

- int [size](#)
- int [length](#)
- bool [allocated](#)

10.230.1 Field Documentation

10.230.1.1 bool Arc::PayloadRawBuf::allocated

size of used memory - size of buffer

10.230.1.2 int Arc::PayloadRawBuf::length

size of allocated memory

10.230.1.3 int Arc::PayloadRawBuf::size

pointer to buffer in memory

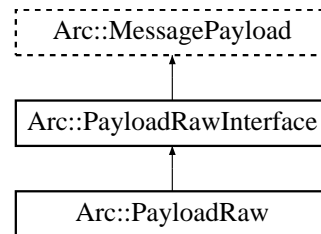
The documentation for this struct was generated from the following file:

- PayloadRaw.h

10.231 Arc::PayloadRawInterface Class Reference

Random Access Payload for [Message](#) objects.

`#include <PayloadRaw.h>`Inheritance diagram for `Arc::PayloadRawInterface`:



Public Member Functions

- virtual char [operator\[\]](#) (Size_t pos) const =0
- virtual char * [Content](#) (Size_t pos=-1)=0
- virtual Size_t [Size](#) (void) const =0
- virtual char * [Insert](#) (Size_t pos=0, Size_t size=0)=0
- virtual char * [Insert](#) (const char *s, Size_t pos=0, Size_t size=-1)=0
- virtual char * [Buffer](#) (unsigned int num)=0
- virtual Size_t [BufferSize](#) (unsigned int num) const =0
- virtual Size_t [BufferPos](#) (unsigned int num) const =0
- virtual bool [Truncate](#) (Size_t size)=0

10.231.1 Detailed Description

Random Access Payload for [Message](#) objects. This class is a virtual interface for managing [Message](#) payload with arbitrarily accessible content. Inheriting classes are supposed to implement memory-resident or memory-mapped content made of optionally multiple chunks/buffers. Every buffer has own size and offset. This class is purely virtual.

10.231.2 Member Function Documentation

10.231.2.1 virtual char* Arc::PayloadRawInterface::Buffer (unsigned int *num*) [pure virtual]

Returns pointer to num'th buffer

Implemented in [Arc::PayloadRaw](#).

10.231.2.2 virtual Size_t Arc::PayloadRawInterface::BufferPos (unsigned int *num*) const [pure virtual]

Returns position of num'th buffer

Implemented in [Arc::PayloadRaw](#).

10.231.2.3 virtual Size_t Arc::PayloadRawInterface::BufferSize (unsigned int *num*) const [pure virtual]

Returns length of *num*'th buffer

Implemented in [Arc::PayloadRaw](#).

10.231.2.4 virtual char* Arc::PayloadRawInterface::Content (Size_t *pos* = -1) [pure virtual]

Get pointer to buffer content at global position '*pos*'. By default to beginning of main buffer whatever that means.

Implemented in [Arc::PayloadRaw](#).

10.231.2.5 virtual char* Arc::PayloadRawInterface::Insert (const char * *s*, Size_t *pos* = 0, Size_t *size* = -1) [pure virtual]

Create new buffer at global position '*pos*' of size '*size*'. Created buffer is filled with content of memory at '*s*'. If '*size*' is negative content at '*s*' is expected to be null-terminated.

Implemented in [Arc::PayloadRaw](#).

10.231.2.6 virtual char* Arc::PayloadRawInterface::Insert (Size_t *pos* = 0, Size_t *size* = 0) [pure virtual]

Create new buffer at global position '*pos*' of size '*size*'.

Implemented in [Arc::PayloadRaw](#).

10.231.2.7 virtual char Arc::PayloadRawInterface::operator[] (Size_t *pos*) const [pure virtual]

Returns content of byte at specified position. Specified position '*pos*' is treated as global one and goes through all buffers placed one after another.

Implemented in [Arc::PayloadRaw](#).

10.231.2.8 virtual Size_t Arc::PayloadRawInterface::Size (void) const [pure virtual]

Returns logical size of whole structure.

Implemented in [Arc::PayloadRaw](#).

10.231.2.9 virtual bool Arc::PayloadRawInterface::Truncate (Size_t *size*) [pure virtual]

Change size of stored information. If size exceeds end of allocated buffer, buffers are not re-allocated, only logical size is extended. Buffers with location behind new size are deallocated.

Implemented in [Arc::PayloadRaw](#).

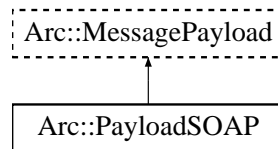
The documentation for this class was generated from the following file:

- PayloadRaw.h

10.232 Arc::PayloadSOAP Class Reference

Payload of [Message](#) with SOAP content.

#include <PayloadSOAP.h> Inheritance diagram for Arc::PayloadSOAP::



Public Member Functions

- [PayloadSOAP](#) (const [NS](#) &ns, bool fault=false)
- [PayloadSOAP](#) (const [SOAPEnvelope](#) &soap)
- [PayloadSOAP](#) (const [MessagePayload](#) &source)

10.232.1 Detailed Description

Payload of [Message](#) with SOAP content. This class combines [MessagePayload](#) with [SOAPEnvelope](#) to make it possible to pass SOAP messages through [MCC](#) chain.

10.232.2 Constructor & Destructor Documentation

10.232.2.1 Arc::PayloadSOAP::PayloadSOAP (const [NS](#) & ns, bool fault = false)

Constructor - creates new [Message](#) payload

10.232.2.2 Arc::PayloadSOAP::PayloadSOAP (const [SOAPEnvelope](#) & soap)

Constructor - creates [Message](#) payload from SOAP document. Provided SOAP document is copied to new object.

10.232.2.3 Arc::PayloadSOAP::PayloadSOAP (const [MessagePayload](#) & source)

Constructor - creates SOAP message from payload. [PayloadRawInterface](#) and derived classes are supported.

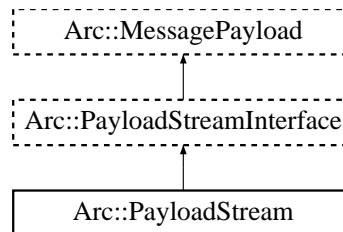
The documentation for this class was generated from the following file:

- [PayloadSOAP.h](#)

10.233 Arc::PayloadStream Class Reference

POSIX handle as Payload.

`#include <PayloadStream.h>`Inheritance diagram for Arc::PayloadStream::



Public Member Functions

- [PayloadStream](#) (int h=-1)
- virtual [~PayloadStream](#) (void)
- virtual bool [Get](#) (char *buf, int &size)
- virtual bool [Put](#) (const char *buf, Size_t size)
- virtual [operator bool](#) (void)
- virtual bool [operator!](#) (void)
- virtual int [Timeout](#) (void) const
- virtual void [Timeout](#) (int to)
- virtual Size_t [Pos](#) (void) const
- virtual Size_t [Size](#) (void) const
- virtual Size_t [Limit](#) (void) const

Protected Attributes

- int [handle_](#)
- bool [seekable_](#)

10.233.1 Detailed Description

POSIX handle as Payload. This is an implemetation of [PayloadStreamInterface](#) for generic POSIX handle.

10.233.2 Constructor & Destructor Documentation

10.233.2.1 Arc::PayloadStream::PayloadStream (int *h* = -1)

true if lseek operation is applicable to open handle Constructor. Attaches to already open handle. Handle is not managed by this class and must be closed by external code.

10.233.2.2 virtual Arc::PayloadStream::~~PayloadStream (void) [[inline](#), [virtual](#)]

Destructor.

10.233.3 Member Function Documentation

10.233.3.1 `virtual bool Arc::PayloadStream::Get (char * buf, int & size) [virtual]`

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.2 `virtual Size_t Arc::PayloadStream::Limit (void) const [inline, virtual]`

Returns position at which stream reading will stop if supported. That may be not same as [Size\(\)](#) if instance is meant to provide access to only part of underlying object.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.3 `virtual Arc::PayloadStream::operator bool (void) [inline, virtual]`

Returns true if stream is valid.

Implements [Arc::PayloadStreamInterface](#).

References `handle_`.

10.233.3.4 `virtual bool Arc::PayloadStream::operator! (void) [inline, virtual]`

Returns true if stream is invalid.

Implements [Arc::PayloadStreamInterface](#).

References `handle_`.

10.233.3.5 `virtual Size_t Arc::PayloadStream::Pos (void) const [inline, virtual]`

Returns current position in stream if supported.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.6 `virtual bool Arc::PayloadStream::Put (const char * buf, Size_t size) [virtual]`

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.7 `virtual Size_t Arc::PayloadStream::Size (void) const [inline, virtual]`

Returns size of underlying object if supported.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.8 `virtual void Arc::PayloadStream::Timeout (int to) [inline, virtual]`

Set current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implements [Arc::PayloadStreamInterface](#).

10.233.3.9 `virtual int Arc::PayloadStream::Timeout (void) const` `[inline, virtual]`

[Query](#) current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implements [Arc::PayloadStreamInterface](#).

10.233.4 Field Documentation

10.233.4.1 `int Arc::PayloadStream::handle_` `[protected]`

Timeout for read/write operations

Referenced by operator bool(), and operator!().

10.233.4.2 `bool Arc::PayloadStream::seekable_` `[protected]`

Handle for operations

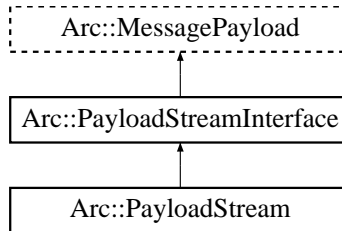
The documentation for this class was generated from the following file:

- PayloadStream.h

10.234 Arc::PayloadStreamInterface Class Reference

Stream-like Payload for [Message](#) object.

#include <PayloadStream.h> Inheritance diagram for Arc::PayloadStreamInterface::



Public Member Functions

- virtual bool [Get](#) (char *buf, int &size)=0
- virtual bool [Get](#) (std::string &buf)
- virtual std::string [Get](#) (void)
- virtual bool [Get](#) ([PayloadStreamInterface](#) &dest, int &size)
- virtual bool [Put](#) (const char *buf, [Size_t](#) size)=0
- virtual bool [Put](#) (const std::string &buf)
- virtual bool [Put](#) (const char *buf)
- virtual bool [Put](#) ([PayloadStreamInterface](#) &source, [Size_t](#) size)
- virtual [operator bool](#) (void)=0
- virtual bool [operator!](#) (void)=0
- virtual int [Timeout](#) (void) const =0
- virtual void [Timeout](#) (int to)=0
- virtual [Size_t](#) [Pos](#) (void) const =0
- virtual [Size_t](#) [Size](#) (void) const =0
- virtual [Size_t](#) [Limit](#) (void) const =0

10.234.1 Detailed Description

Stream-like Payload for [Message](#) object. This class is a virtual interface for managing stream-like source and destination. It's supposed to be passed through [MCC](#) chain as payload of [Message](#). It must be treated by MCCs and Services as dynamic payload.

10.234.2 Member Function Documentation

10.234.2.1 virtual bool Arc::PayloadStreamInterface::Get (PayloadStreamInterface & dest, int & size) [virtual]

Read up to 'size' bytes and pass them to 'dest'. 'size' contains number of read bytes on exit. If on input 'size' contains -1 then as much as possible is transfered. This method is both for convenience and for making it possible to have optimized implementations.

10.234.2.2 virtual std::string Arc::PayloadStreamInterface::Get (void) [virtual]

Read and return as many as possible (sane amount) of bytes. Implemented through call to [Get\(std::string&\)](#).

10.234.2.3 virtual bool Arc::PayloadStreamInterface::Get (std::string & buf) [virtual]

Read as many as possible (sane amount) of bytes into buf. Implemented through call to [Get\(char*,int\)](#).

10.234.2.4 virtual bool Arc::PayloadStreamInterface::Get (char * buf, int & size) [pure virtual]

Extracts information from stream up to 'size' bytes. 'size' contains number of read bytes on exit. Returns true in case of success.

Implemented in [Arc::PayloadStream](#).

10.234.2.5 virtual Size_t Arc::PayloadStreamInterface::Limit (void) const [pure virtual]

Returns position at which stream reading will stop if supported. That may be not same as [Size\(\)](#) if instance is meant to provide access to only part of underlying object.

Implemented in [Arc::PayloadStream](#).

10.234.2.6 virtual Arc::PayloadStreamInterface::operator bool (void) [pure virtual]

Returns true if stream is valid.

Implemented in [Arc::PayloadStream](#).

10.234.2.7 virtual bool Arc::PayloadStreamInterface::operator! (void) [pure virtual]

Returns true if stream is invalid.

Implemented in [Arc::PayloadStream](#).

10.234.2.8 virtual Size_t Arc::PayloadStreamInterface::Pos (void) const [pure virtual]

Returns current position in stream if supported.

Implemented in [Arc::PayloadStream](#).

10.234.2.9 virtual bool Arc::PayloadStreamInterface::Put (PayloadStreamInterface & source, Size_t size) [virtual]

Push 'size' bytes from 'source' into stream. If on 'size' contains -1 then as much as possible is transferred. This method is both for convenience and for making it possible to have optimized implementations.

10.234.2.10 virtual bool Arc::PayloadStreamInterface::Put (const char * *buf*) [virtual]

Push null terminated information from 'buf' into stream. Returns true on success. Implemented though call to [Put\(const char*,Size_t\)](#).

10.234.2.11 virtual bool Arc::PayloadStreamInterface::Put (const std::string & *buf*) [virtual]

Push information from 'buf' into stream. Returns true on success. Implemented though call to [Put\(const char*,Size_t\)](#).

10.234.2.12 virtual bool Arc::PayloadStreamInterface::Put (const char * *buf*, Size_t *size*) [pure virtual]

Push 'size' bytes from 'buf' into stream. Returns true on success.

Implemented in [Arc::PayloadStream](#).

10.234.2.13 virtual Size_t Arc::PayloadStreamInterface::Size (void) const [pure virtual]

Returns size of underlying object if supported.

Implemented in [Arc::PayloadStream](#).

10.234.2.14 virtual void Arc::PayloadStreamInterface::Timeout (int *to*) [pure virtual]

Set current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implemented in [Arc::PayloadStream](#).

10.234.2.15 virtual int Arc::PayloadStreamInterface::Timeout (void) const [pure virtual]

[Query](#) current timeout for [Get\(\)](#) and [Put\(\)](#) operations.

Implemented in [Arc::PayloadStream](#).

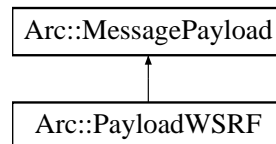
The documentation for this class was generated from the following file:

- [PayloadStream.h](#)

10.235 Arc::PayloadWSRF Class Reference

This class combines [MessagePayload](#) with [WSRF](#).

#include <PayloadWSRF.h> Inheritance diagram for Arc::PayloadWSRF::



Public Member Functions

- [PayloadWSRF](#) (const SOAPEnvelope &soap)
- [PayloadWSRF](#) ([WSRF](#) &wsrp)
- [PayloadWSRF](#) (const [MessagePayload](#) &source)

10.235.1 Detailed Description

This class combines [MessagePayload](#) with [WSRF](#). It's intention is to make it possible to pass [WSRF](#) messages through [MCC](#) chain as one more Payload type.

10.235.2 Constructor & Destructor Documentation

10.235.2.1 Arc::PayloadWSRF::PayloadWSRF (const SOAPEnvelope & soap)

Constructor - creates [Message](#) payload from SOAP message. Returns invalid [WSRF](#) if SOAP does not represent WS-ResourceProperties

10.235.2.2 Arc::PayloadWSRF::PayloadWSRF (WSRF & wsrp)

Constructor - creates [Message](#) payload with acquired [WSRF](#) message. [WSRF](#) message will be destroyed by destructor of this object.

10.235.2.3 Arc::PayloadWSRF::PayloadWSRF (const MessagePayload & source)

Constructor - creates [WSRF](#) message from payload. All classes derived from SOAPEnvelope are supported.

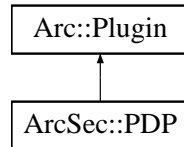
The documentation for this class was generated from the following file:

- PayloadWSRF.h

10.236 ArcSec::PDP Class Reference

Base class for [Policy](#) Decision Point plugins.

#include <PDP.h>Inheritance diagram for ArcSec::PDP::



10.236.1 Detailed Description

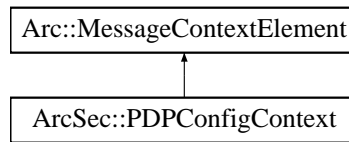
Base class for [Policy](#) Decision Point plugins. This virtual class defines method `isPermitted()` which processes security related information/attributes in `Message` and makes security decision - permit (true) or deny (false). Configuration of [PDP](#) is consumed during creation of instance through XML subtree fed to constructor.

The documentation for this class was generated from the following file:

- PDP.h

10.237 ArcSec::PDPCfgContext Class Reference

Inheritance diagram for ArcSec::PDPCfgContext::

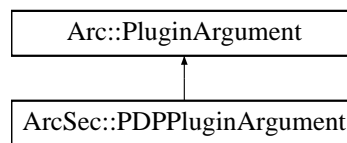


The documentation for this class was generated from the following file:

- PDP.h

10.238 ArcSec::PDPPluginArgument Class Reference

Inheritance diagram for ArcSec::PDPPluginArgument::



The documentation for this class was generated from the following file:

- PDP.h

10.239 Arc::Period Class Reference

A [Period](#) represents a length of time.

```
#include <arc/DateTime.h>
```

Public Member Functions

- [Period](#) ()
- [Period](#) (time_t)
- [Period](#) (time_t seconds, uint32_t nanoseconds)
- [Period](#) (const std::string &, [PeriodBase](#) base=PeriodSeconds)
- [Period](#) & [operator=](#) (time_t)
- [Period](#) & [operator=](#) (const [Period](#) &)
- void [SetPeriod](#) (time_t sec)
- void [SetPeriod](#) (time_t sec, uint32_t nanosec)
- time_t [GetPeriod](#) () const
- time_t [GetPeriodNanoseconds](#) () const
- const sigc::slot< const char * > * [istr](#) () const
- [operator std::string](#) () const
- bool [operator<](#) (const [Period](#) &) const
- bool [operator>](#) (const [Period](#) &) const
- bool [operator<=](#) (const [Period](#) &) const
- bool [operator>=](#) (const [Period](#) &) const
- bool [operator==](#) (const [Period](#) &) const
- bool [operator!=](#) (const [Period](#) &) const

10.239.1 Detailed Description

A [Period](#) represents a length of time. [Period](#) represents a length of time (eg 2 mins and 30.1 seconds), whereas [Time](#) represents a moment of time (eg midnight on 1st Jan 2000).

See also:

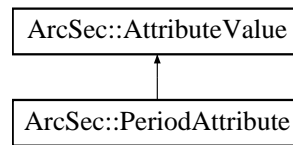
[Time](#)

The documentation for this class was generated from the following file:

- DateTime.h

10.240 ArcSec::PeriodAttribute Class Reference

#include <DateTimeAttribute.h> Inheritance diagram for ArcSec::PeriodAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.240.1 Detailed Description

Format: datetime"/"duration datetime"/"datetime duration"/"datetime

10.240.2 Member Function Documentation

10.240.2.1 virtual std::string ArcSec::PeriodAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.240.2.2 virtual std::string ArcSec::PeriodAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.240.2.3 virtual std::string ArcSec::PeriodAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

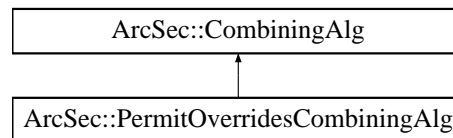
The documentation for this class was generated from the following file:

- DateTimeAttribute.h

10.241 ArcSec::PermitOverridesCombiningAlg Class Reference

Implement the "Permit-Overrides" algorithm.

#include <PermitOverridesAlg.h> Inheritance diagram for ArcSec::PermitOverridesCombiningAlg::



Public Member Functions

- virtual Result [combine](#) (EvaluationCtx *ctx, std::list< Policy * > policies)
- virtual const std::string & [getalgId](#) (void) const

10.241.1 Detailed Description

Implement the "Permit-Overrides" algorithm. Permit-Overrides, scans the policy set which is given as the parameters of "combine" method, if gets "permit" result from any policy, then stops scanning and gives "permit" as result, otherwise gives "deny".

10.241.2 Member Function Documentation

10.241.2.1 virtual Result ArcSec::PermitOverridesCombiningAlg::combine (EvaluationCtx * ctx, std::list< Policy * > policies) [virtual]

If there is one policy which return positive evaluation result, then omit the other policies and return DECISION_PERMIT

Parameters:

- ctx* This object contains request information which will be used to evaluated against policy.
policies This is a container which contains policy objects.

Returns:

The combined result according to the algorithm.

Implements [ArcSec::CombiningAlg](#).

10.241.2.2 virtual const std::string& ArcSec::PermitOverridesCombiningAlg::getalgId (void) const [inline, virtual]

Get the identifier

Implements [ArcSec::CombiningAlg](#).

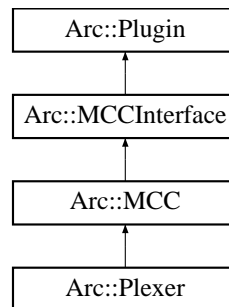
The documentation for this class was generated from the following file:

- `PermitOverridesAlg.h`

10.242 Arc::Plexer Class Reference

The [Plexer](#) class, used for routing messages to services.

`#include <Plexer.h>`Inheritance diagram for Arc::Plexer::



Public Member Functions

- [Plexer](#) ([Config](#) *cfg, [PluginArgument](#) *arg)
- virtual [~Plexer](#) ()
- virtual void [Next](#) ([MCCInterface](#) *next, const std::string &label)
- virtual [MCC_Status](#) process ([Message](#) &request, [Message](#) &response)

Static Public Attributes

- static [Logger](#) logger

10.242.1 Detailed Description

The [Plexer](#) class, used for routing messages to services. This is the [Plexer](#) class. Its purpose is to route incoming messages to appropriate Services and [MCC](#) chains.

10.242.2 Constructor & Destructor Documentation

10.242.2.1 Arc::Plexer::Plexer (Config * cfg, PluginArgument * arg)

The constructor. This is the constructor. Since all member variables are instances of "well-behaving" STL classes, nothing needs to be done.

10.242.2.2 virtual Arc::Plexer::~~Plexer () [virtual]

The destructor. This is the destructor. Since all member variables are instances of "well-behaving" STL classes, nothing needs to be done.

10.242.3 Member Function Documentation

10.242.3.1 `virtual void Arc::Plexer::Next (MCCInterface * next, const std::string & label)` [`virtual`]

Add reference to next [MCC](#) in chain. This method is called by [Loader](#) for every potentially labeled link to next component which implements [MCCInterface](#). If next is set NULL corresponding link is removed.

Reimplemented from [Arc::MCC](#).

10.242.3.2 `virtual MCC_Status Arc::Plexer::process (Message & request, Message & response)` [`virtual`]

Route request messages to appropriate services. Routes the request message to the appropriate service. Routing is based on the path part of value of the ENDPOINT attribute. Routed message is assigned following attributes: PLEXER:PATTERN - matched pattern, PLEXER:EXTENSION - last unmatched part of ENDPOINT path.

Reimplemented from [Arc::MCC](#).

10.242.4 Field Documentation

10.242.4.1 `Logger Arc::Plexer::logger` [`static`]

A logger for MCCs. A logger intended to be the parent of loggers in the different MCCs.

Reimplemented from [Arc::MCC](#).

The documentation for this class was generated from the following file:

- `Plexer.h`

10.243 Arc::PlexerEntry Class Reference

A pair of label (regex) and pointer to [MCC](#).

```
#include <Plexer.h>
```

10.243.1 Detailed Description

A pair of label (regex) and pointer to [MCC](#). A helper class that stores a label (regex) and a pointer to a service.

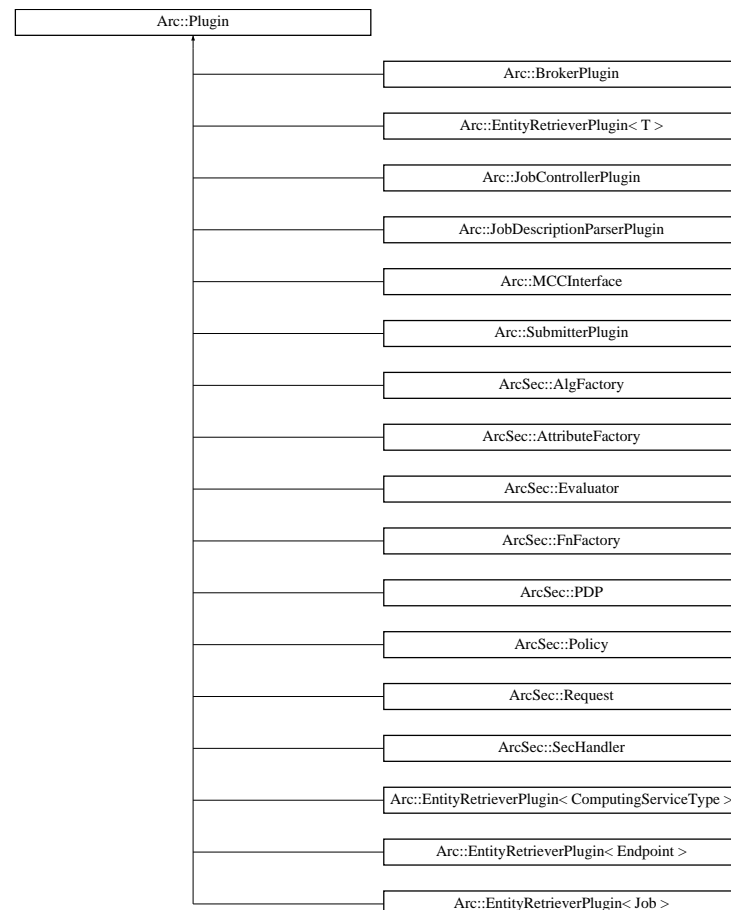
The documentation for this class was generated from the following file:

- Plexer.h

10.244 Arc::Plugin Class Reference

Base class for loadable ARC components.

#include <Plugin.h>Inheritance diagram for Arc::Plugin::



Protected Member Functions

- [Plugin](#) ([PluginArgument](#) *arg)
- [Plugin](#) (const [Plugin](#) &obj)

10.244.1 Detailed Description

Base class for loadable ARC components. All classes representing loadable ARC components must be either descendants of this class or be wrapped by its offspring.

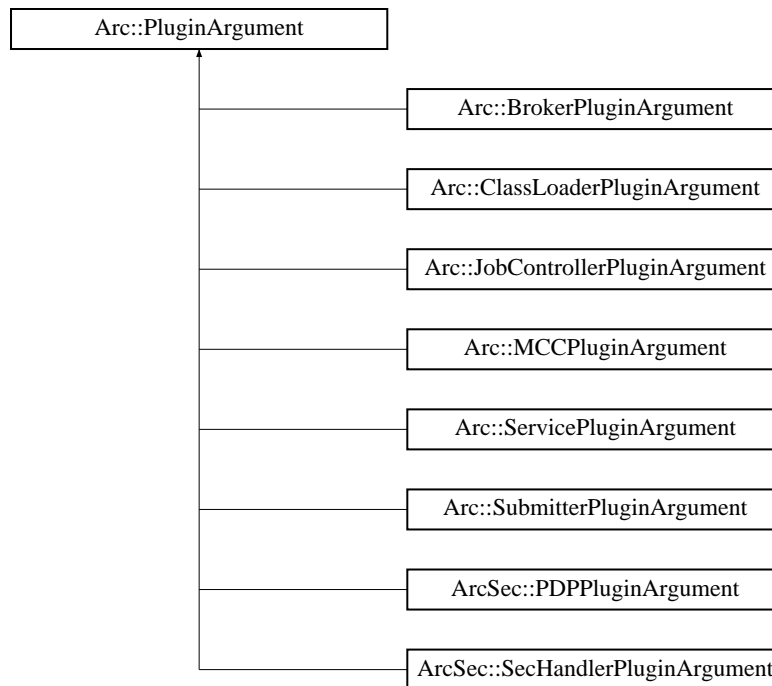
The documentation for this class was generated from the following file:

- [Plugin.h](#)

10.245 Arc::PluginArgument Class Reference

Base class for passing arguments to loadable ARC components.

#include <Plugin.h>Inheritance diagram for Arc::PluginArgument::



Public Member Functions

- `PluginsFactory * get_factory (void)`
- `Glib::Module * get_module (void)`

10.245.1 Detailed Description

Base class for passing arguments to loadable ARC components. During its creation constructor function of ARC loadable component expects instance of class inherited from this one or wrapped in it. Then dynamic type casting is used for obtaining class of expected kind.

10.245.2 Member Function Documentation

10.245.2.1 PluginsFactory* Arc::PluginArgument::get_factory (void)

Returns pointer to factory which instantiated plugin. Because factory usually destroys/unloads plugins in its destructor it should be safe to keep this pointer inside plugin for later use. But one must always check.

10.245.2.2 Glib::Module* Arc::PluginArgument::get_module (void)

Returns pointer to loadable module/library which contains plugin. Corresponding factory keeps list of modules till itself is destroyed. So it should be safe to keep that pointer. But care must be taken if module contains persistent plugins. Such modules stay in memory after factory is destroyed. So it is advisable to use obtained pointer only in constructor function of plugin.

The documentation for this class was generated from the following file:

- Plugin.h

10.246 Arc::PluginDesc Class Reference

Description of plugin.

```
#include <Plugin.h>
```

10.246.1 Detailed Description

Description of plugin. This class is used for reports

The documentation for this class was generated from the following file:

- Plugin.h

10.247 Arc::PluginDescriptor Struct Reference

Description of ARC lodable component.

```
#include <Plugin.h>
```

10.247.1 Detailed Description

Description of ARC lodable component.

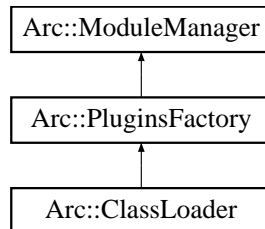
The documentation for this struct was generated from the following file:

- Plugin.h

10.248 Arc::PluginsFactory Class Reference

Generic ARC plugins loader.

`#include <Plugin.h>`Inheritance diagram for Arc::PluginsFactory::



Data Structures

- class **descriptor_t_**
- class **module_t_**
- class **modules_t_**

Public Member Functions

- [PluginsFactory](#) ([XMLNode](#) cfg)
- void [TryLoad](#) (bool v)
- [Plugin](#) * [get_instance](#) (const std::string &kind, [PluginArgument](#) *arg, bool search=true)
- bool [load](#) (const std::string &name)
- bool [scan](#) (const std::string &name, [ModuleDesc](#) &desc)
- void [report](#) (std::list< [ModuleDesc](#) > &descs)

Static Public Member Functions

- static void [FilterByKind](#) (const std::string &kind, std::list< [ModuleDesc](#) > &descs)

10.248.1 Detailed Description

Generic ARC plugins loader. The instance of this class provides functionality of loading pluggable ARC components stored in shared libraries. It also makes use of [Arc Plugin](#) Description (*.apd) files which contain textual plugin identifiers. [Arc Plugin](#) Description files contain attributes describing pluggable components stored in corresponding shared libraries. Using those files allows to save on actually loading and resolving libraries while looking for specific component.

Specifically this class uses 'priority' attribute to sort plugin description in internal lists. Please note that priority affects order in which plugins tried in [get_instance\(...\)](#) methods. But it only works for plugins which were already loaded by previous calls to [load\(...\)](#) and [get_instance\(...\)](#) methods. For plugins discovered inside [get_instance](#) priority is not effective.

This class mostly handles tasks of finding, identifying, filtering and sorting ARC pluggable components. For loading shared libraries it uses functionality of [ModuleManager](#) class. So it is important to see documentation of [ModuleManager](#) in order to understand how this class works.

For more information also please check ARC HED documentation.

This class is thread-safe - its methods are protected from simultaneous use from multiple threads. Current thread protection implementation is suboptimal and will be revised in future.

10.248.2 Constructor & Destructor Documentation

10.248.2.1 Arc::PluginsFactory::PluginsFactory (XMLNode *cfg*)

Constructor - accepts configuration (not yet used) meant to tune loading of modules.

10.248.3 Member Function Documentation

10.248.3.1 static void Arc::PluginsFactory::FilterByKind (const std::string & *kind*, std::list< ModuleDesc > & *descs*) [static]

Filter list of modules by kind.

10.248.3.2 Plugin* Arc::PluginsFactory::get_instance (const std::string & *kind*, PluginArgument * *arg*, bool *search* = true)

These methods load module named lib'name', locate plugin constructor functions of specified 'kind' and 'name' (if specified) and call it. Supplied argument affects way plugin instance is created in plugin-specific way. If name of plugin is not specified then all plugins of specified kind are tried with supplied argument till valid instance is created. All loaded plugins are also registered in internal list of this instance of [PluginsFactory](#) class. If search is set to false then no attempt is made to find plugins in loadable modules. Only plugins already loaded with previous calls to [get_instance\(\)](#) and [load\(\)](#) are checked. Returns created instance or NULL if failed.

10.248.3.3 bool Arc::PluginsFactory::load (const std::string & *name*)

These methods load module named lib'name' and check if it contains ARC plugin(s) of specified 'kind' and 'name'. If there are no specified plugins or module does not contain any ARC plugins it is unloaded. All loaded plugins are also registered in internal list of this instance of [PluginsFactory](#) class. Returns true if any plugin was loaded.

10.248.3.4 void Arc::PluginsFactory::report (std::list< ModuleDesc > & *descs*)

Provides information about currently loaded modules and plugins.

10.248.3.5 bool Arc::PluginsFactory::scan (const std::string & *name*, ModuleDesc & *desc*)

Collect information about plugins stored in module(s) with specified names. Returns true if any of specified modules has plugins.

10.248.3.6 void Arc::PluginsFactory::TryLoad (bool *v*) [inline]

Specifies if loadable module may be loaded while looking for analyzing its content. If set to false only *.apd files are checked. Modules without corresponding *.apd will be ignored. Default is true;

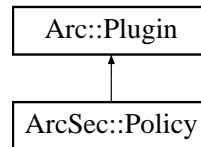
The documentation for this class was generated from the following file:

- [Plugin.h](#)

10.249 ArcSec::Policy Class Reference

Interface for containing and processing different types of policy.

#include <Policy.h> Inheritance diagram for ArcSec::Policy::



Public Member Functions

- [Policy](#) ([Arc::PluginArgument](#) *parg)
- [Policy](#) (const [Arc::XMLNode](#), [Arc::PluginArgument](#) *parg)
- [Policy](#) (const [Arc::XMLNode](#), [EvaluatorContext](#) *, [Arc::PluginArgument](#) *parg)
- virtual [operator bool](#) (void) const =0
- virtual [MatchResult](#) [match](#) ([EvaluationCtx](#) *) =0
- virtual [Result](#) [eval](#) ([EvaluationCtx](#) *) =0
- virtual void [addPolicy](#) ([Policy](#) *pl)
- virtual void [setEvaluatorContext](#) ([EvaluatorContext](#) *)
- virtual void [make_policy](#) ()
- virtual std::string [getEffect](#) () const =0
- virtual [EvalResult](#) & [getEvalResult](#) () =0
- virtual void [setEvalResult](#) ([EvalResult](#) &res) =0
- virtual const char * [getEvalName](#) () const =0
- virtual const char * [getName](#) () const =0

10.249.1 Detailed Description

Interface for containing and processing different types of policy. Basically, each policy object is a container which includes a few elements e.g., [ArcPolicySet](#) objects includes a few [ArcPolicy](#) objects; [ArcPolicy](#) object includes a few [ArcRule](#) objects. There is logical relationship between [ArcRules](#) or [ArcPolicies](#), which is called combining algorithm. According to algorithm, evaluation results from the elements are combined, and then the combined evaluation result is returned to the up-level.

10.249.2 Constructor & Destructor Documentation

10.249.2.1 ArcSec::Policy::Policy (const Arc::XMLNode, Arc::PluginArgument *parg) [inline]

Template constructor - creates policy based on XML document. If XML document is empty then empty policy is created. If it is not empty then it must be valid policy document - otherwise created object should be invalid.

10.249.2.2 ArcSec::Policy::Policy (const Arc::XMLNode, EvaluatorContext *, Arc::PluginArgument *parg) [inline]

Template constructor - creates policy based on XML document. If XML document is empty then empty policy is created. If it is not empty then it must be valid policy document - otherwise created object should be invalid. This constructor is based on the policy node and i the [EvaluatorContext](#) which includes the factory objects for combining algorithm and function

10.249.3 Member Function Documentation

10.249.3.1 virtual void ArcSec::Policy::addPolicy (Policy *pl) [inline, virtual]

Add a policy element to into "this" object

10.249.3.2 virtual Result ArcSec::Policy::eval (EvaluationCtx *) [pure virtual]

Evaluate policy For the <Rule> of [Arc](#), only get the "Effect" from rules; For the <Policy> of [Arc](#), combine the evaluation result from <Rule>; For the <Rule> of XACML, evaluate the <Condition> node by using information from request, and use the "Effect" attribute of <Rule>; For the <Policy> of XACML, combine the evaluation result from <Rule>

10.249.3.3 virtual std::string ArcSec::Policy::getEffect () const [pure virtual]

Get the "Effect" attribute

10.249.3.4 virtual const char* ArcSec::Policy::getEvalName () const [pure virtual]

Get the name of [Evaluator](#) which can evaluate this policy

10.249.3.5 virtual EvalResult& ArcSec::Policy::getEvalResult () [pure virtual]

Get eveluation result

10.249.3.6 virtual const char* ArcSec::Policy::getName () const [pure virtual]

Get the name of this policy

10.249.3.7 virtual void ArcSec::Policy::make_policy () [inline, virtual]

Parse XMLNode, and construct the low-level Rule object

10.249.3.8 virtual void ArcSec::Policy::setEvalResult (EvalResult &res) [pure virtual]

Set eveluation result

10.249.3.9 `virtual void ArcSec::Policy::setEvaluatorContext (EvaluatorContext *)` [`inline`,
`virtual`]

Set [Evaluator](#) Context for the usage in creating low-level policy object

The documentation for this class was generated from the following file:

- Policy.h

10.250 ArcSec::PolicyStore::PolicyElement Class Reference

The documentation for this class was generated from the following file:

- PolicyStore.h

10.251 ArcSec::PolicyParser Class Reference

A interface which will isolate the policy object from actual policy storage (files, urls, database).

```
#include <PolicyParser.h>
```

Public Member Functions

- virtual [Policy](#) * [parsePolicy](#) (const [Source](#) &source, std::string policyclassname, [EvaluatorContext](#) *ctx)

10.251.1 Detailed Description

A interface which will isolate the policy object from actual policy storage (files, urls, database). Parse the policy from policy source (e.g. files, urls, database, etc.).

10.251.2 Member Function Documentation

10.251.2.1 virtual [Policy](#)* [ArcSec::PolicyParser::parsePolicy](#) (const [Source](#) & *source*, std::string *policyclassname*, [EvaluatorContext](#) * *ctx*) [**virtual**]

Parse policy

Parameters:

- source* location of the policy
- policyclassname* name of the policy for ClassLoader
- ctx* [EvaluatorContext](#) which includes the **Factory

The documentation for this class was generated from the following file:

- PolicyParser.h

10.252 ArcSec::PolicyStore Class Reference

Storage place for policy objects.

```
#include <PolicyStore.h>
```

Data Structures

- class [PolicyElement](#)

Public Member Functions

- [PolicyStore](#) (const std::string &alg, const std::string &policyclassname, [EvaluatorContext](#) *ctx)

10.252.1 Detailed Description

Storage place for policy objects.

10.252.2 Constructor & Destructor Documentation

10.252.2.1 ArcSec::PolicyStore::PolicyStore (const std::string & *alg*, const std::string & *policyclassname*, [EvaluatorContext](#) * *ctx*)

Creates policy store with specified combining algorithm (alg - not used yet), policy name (policyclassname) and context (ctx)

The documentation for this class was generated from the following file:

- PolicyStore.h

10.253 AuthN::PrivateKeyInfoCodec Class Reference

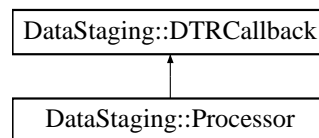
The documentation for this class was generated from the following file:

- nssprivkeyinfocodec.h

10.254 DataStaging::Processor Class Reference

The [Processor](#) performs pre- and post-transfer operations.

```
#include <arc/data-staging/Processor.h>Inheritance diagram for DataStaging::Processor::
```



Data Structures

- class **BulkThreadArgument**

Class used to pass information to spawned thread (for bulk operations).

- class **ThreadArgument**

Class used to pass information to spawned thread.

Public Member Functions

- [Processor](#) ()
- [~Processor](#) ()
- void [start](#) (void)
- void [stop](#) (void)
- virtual void [receiveDTR](#) ([DTR_ptr](#) dtr)

10.254.1 Detailed Description

The [Processor](#) performs pre- and post-transfer operations. The [Processor](#) takes care of everything that should happen before and after a transfer takes place. Calling [receiveDTR\(\)](#) spawns a thread to perform the required operation depending on the [DTR](#) state.

10.254.2 Member Function Documentation

10.254.2.1 virtual void DataStaging::Processor::receiveDTR ([DTR_ptr](#) dtr) [virtual]

Send a [DTR](#) to the [Processor](#). The [DTR](#) is sent to the [Processor](#) through this method when some long-latency processing is to be performed, eg contacting a remote service. The [Processor](#) spawns a thread to do the processing, and then returns. The thread pushes the [DTR](#) back to the scheduler when it is finished.

Implements [DataStaging::DTRCallback](#).

10.254.2.2 void DataStaging::Processor::start (void)

Start [Processor](#). This method actually does nothing. It is here only to make all classes of data staging to look alike. But it is better to call it before starting to use object because it may do something in the future.

10.254.2.3 void DataStaging::Processor::stop (void)

Stop [Processor](#). This method sends waits for all started threads to end and exits. Since threads are short-lived it is better to wait rather than interrupt them.

Referenced by `~Processor()`.

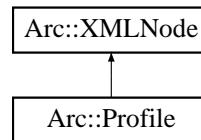
The documentation for this class was generated from the following file:

- `Processor.h`

10.255 Arc::Profile Class Reference

Class used to convert human-friendly ini-style configuration to XML.

`#include <arc/Profile.h>`Inheritance diagram for Arc::Profile::



Public Member Functions

- [Profile](#) (const std::string &filename)
- void [Evaluate](#) ([Config](#) &cfg, [IniConfig](#) ini)

10.255.1 Detailed Description

Class used to convert human-friendly ini-style configuration to XML.

The documentation for this class was generated from the following file:

- Profile.h

10.256 ArcCredential::PROXYCERTINFO_st Struct Reference

The documentation for this struct was generated from the following file:

- Proxycertinfo.h

10.257 ArcCredential::PROXYPOLICY_st Struct Reference

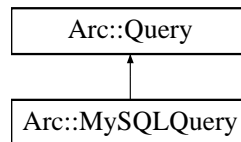
The documentation for this struct was generated from the following file:

- Proxycertinfo.h

10.258 Arc::Query Class Reference

Class representing a database query.

#include <arc/DBInterface.h> Inheritance diagram for Arc::Query::



Public Member Functions

- [Query \(\)](#)
- [Query \(Database *db\)](#)
- virtual [~Query \(\)](#)
- virtual int [get_num_columns \(\)](#)=0
- virtual int [get_num_rows \(\)](#)=0
- virtual bool [execute](#) (const std::string &sqlstr)=0
- virtual QueryRowResult [get_row](#) (int row_number) const =0
- virtual QueryRowResult [get_row](#) () const =0
- virtual std::string [get_row_field](#) (int row_number, std::string &field_name)=0
- virtual bool [get_array](#) (std::string &sqlstr, QueryArrayResult &result, std::vector< std::string > &arguments)=0

10.258.1 Detailed Description

Class representing a database query.

10.258.2 Constructor & Destructor Documentation

10.258.2.1 Arc::Query::Query (Database * db) [inline]

Constructor.

Parameters:

db The database object which will be used by [Query](#) class to get the database connection

10.258.3 Member Function Documentation

10.258.3.1 virtual bool Arc::Query::execute (const std::string & sqlstr) [pure virtual]

Execute the query.

Parameters:

sqlstr The sql sentence used to query

Implemented in [Arc::MySQLQuery](#).

10.258.3.2 virtual bool Arc::Query::get_array (std::string & *sqlstr*, QueryArrayResult & *result*, std::vector< std::string > & *arguments*) [pure virtual]

[Query](#) the database by using some parameters into sql sentence. An example sentence: "select table.value from table where table.name = ?"

Parameters:

sqlstr The sql sentence with some parameters marked with "?".

result The result in an array which includes all of the value in query result.

arguments The argument list which should exactly correspond with the parameters in the sql sentence.

Implemented in [Arc::MySQLQuery](#).

10.258.3.3 virtual QueryRowResult Arc::Query::get_row () const [pure virtual]

Get the value of one row in the query result. The row number will be automatically increased each time the method is called.

Implemented in [Arc::MySQLQuery](#).

10.258.3.4 virtual QueryRowResult Arc::Query::get_row (int *row_number*) const [pure virtual]

Get the value of one row in the query result.

Parameters:

row_number The number of the row

Returns:

A vector includes all the values in the row

Implemented in [Arc::MySQLQuery](#).

10.258.3.5 virtual std::string Arc::Query::get_row_field (int *row_number*, std::string & *field_name*) [pure virtual]

Get the value of one specific field in one specific row.

Parameters:

row_number The row number inside the query result

field_name The field name for the value which will be return

Returns:

The value of the specified filed in the specified row

Implemented in [Arc::MySQLQuery](#).

The documentation for this class was generated from the following file:

- DBInterface.h

10.259 Arc::Range< T > Class Template Reference

`template<class T> class Arc::Range< T >`

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.260 Arc::Register_Info_Type Struct Reference

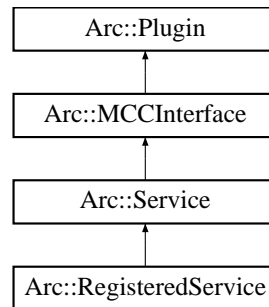
The documentation for this struct was generated from the following file:

- InfoRegister.h

10.261 Arc::RegisteredService Class Reference

[RegisteredService](#) - extension of [Service](#) performing self-registration.

#include <RegisteredService.h> Inheritance diagram for Arc::RegisteredService::



Public Member Functions

- [RegisteredService](#) ([Config](#) *, [PluginArgument](#) *)

10.261.1 Detailed Description

[RegisteredService](#) - extension of [Service](#) performing self-registration. [Service](#) is automatically added to registration framework. Registration information for service is obtained by calling its [RegistrationCollector\(\)](#) method. It is important to note that [RegistrationCollector\(\)](#) may be called anytime after [RegisteredService](#) constructor completed and hence even before actual constructor of inheriting class is complete. That must be taken into account while writing implementation of [RegistrationCollector\(\)](#) or object of [InfoRegisters](#) class must be used directly.

10.261.2 Constructor & Destructor Documentation

10.261.2.1 Arc::RegisteredService::RegisteredService (Config *, PluginArgument *)

Example constructor - Server takes at least it's configuration subtree

The documentation for this class was generated from the following file:

- `RegisteredService.h`

10.262 Arc::RegularExpression Class Reference

A regular expression class.

```
#include <arc/ArcRegex.h>
```

Public Member Functions

- [RegularExpression](#) ()
- [RegularExpression](#) (std::string pattern)
- [RegularExpression](#) (const [RegularExpression](#) ®ex)
- [~RegularExpression](#) ()
- [RegularExpression](#) & [operator=](#) (const [RegularExpression](#) ®ex)
- bool [isOk](#) ()
- bool [hasPattern](#) (std::string str)
- bool [match](#) (const std::string &str) const
- bool [match](#) (const std::string &str, std::list< std::string > &unmatched, std::list< std::string > &matched) const
- std::string [getPattern](#) () const

10.262.1 Detailed Description

A regular expression class. This class is a wrapper around the functions provided in regex.h.

10.262.2 Member Function Documentation

10.262.2.1 bool Arc::RegularExpression::match (const std::string & *str*, std::list< std::string > & *unmatched*, std::list< std::string > & *matched*) const

Returns true if this regex matches the string provided. Unmatched parts of the string are stored in 'unmatched'. Matched parts of the string are stored in 'matched'. The first entry in matched is the string that matched the regex, and the following entries are parenthesised elements of the regex.

The documentation for this class was generated from the following file:

- ArcRegex.h

10.263 Arc::RemoteLoggingType Class Reference

Remote logging.

```
#include <arc/compute/JobDescription.h>
```

Data Fields

- std::string [ServiceType](#)
- [URL Location](#)
- bool [optional](#)

10.263.1 Detailed Description

Remote logging. This class is used to specify a service which should be used to report logging information to, such as job resource usage.

10.263.2 Field Documentation

10.263.2.1 URL Arc::RemoteLoggingType::Location

[URL](#) of logging service. The Location [URL](#) specifies the [URL](#) of the service which job logging information should be sent to.

10.263.2.2 bool Arc::RemoteLoggingType::optional

Requirement satisfaction switch. The optional boolean specifies whether the requirement specified in the particular object is mandatory for job execution, or whether it be ignored.

10.263.2.3 std::string Arc::RemoteLoggingType::ServiceType

Type of logging service. The ServiceType string specifies the type of logging service. Some examples are "SGAS" (<http://www.sgas.se>) and "APEL" (<https://wiki.egi.eu/wiki/APEL>), however please refer to the particular execution service for a list of supported logging service types.

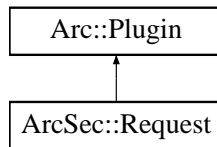
The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.264 ArcSec::Request Class Reference

Base class/Interface for request, includes a container for RequestItems and some operations.

#include <Request.h> Inheritance diagram for ArcSec::Request::



Public Member Functions

- virtual ReqItemList [getRequestItems](#) () const
- virtual void [setRequestItems](#) (ReqItemList)
- virtual void [addRequestItem](#) (Attrs &, Attrs &, Attrs &, Attrs &)
- virtual void [setAttributeFactory](#) (AttributeFactory *attributefactory)=0
- virtual void [make_request](#) ()=0
- virtual const char * [getEvalName](#) () const =0
- virtual const char * [getName](#) () const =0
- [Request](#) (Arc::PluginArgument *parg)
- [Request](#) (const Source &, Arc::PluginArgument *parg)

10.264.1 Detailed Description

Base class/Interface for request, includes a container for RequestItems and some operations. A [Request](#) object can has a few <subjects, actions, objects> tuples, i.e. [RequestItem](#) The [Request](#) class and any customized class which inherit from it, should be loadable, which means these classes can be dynamically loaded according to the configuration information, see the example configuration below: <Service name="pdp.service" id="pdp_service"> <pdp:PDPCfg> <.....> <pdp:[Request](#) name="arc.request" /> <.....> </pdp:PDPCfg> </Service>

There can be different types of subclass which inherit [Request](#), such like XACMLRequest, ArcRequest, GACLRequest

10.264.2 Constructor & Destructor Documentation

10.264.2.1 ArcSec::Request::Request (Arc::PluginArgument *parg) [inline]

Default constructor

10.264.2.2 ArcSec::Request::Request (const Source &, Arc::PluginArgument *parg) [inline]

Constructor: Parse request information from a xml stucture in memory

10.264.3 Member Function Documentation

10.264.3.1 `virtual void ArcSec::Request::addRequestItem (Attrs &, Attrs &, Attrs &, Attrs &) [inline, virtual]`

Add request tuple from non-XMLNode

10.264.3.2 `virtual const char* ArcSec::Request::getEvalName () const [pure virtual]`

Get the name of corresponding evaluator

10.264.3.3 `virtual const char* ArcSec::Request::getName () const [pure virtual]`

Get the name of this request

10.264.3.4 `virtual ReqItemList ArcSec::Request::getRequestItems () const [inline, virtual]`

Get all the [RequestItem](#) inside [RequestItem](#) container

10.264.3.5 `virtual void ArcSec::Request::make_request () [pure virtual]`

Create the objects included in [Request](#) according to the node attached to the [Request](#) object

10.264.3.6 `virtual void ArcSec::Request::setAttributeFactory (AttributeFactory * attributefactory) [pure virtual]`

Set the attribute factory for the usage of [Request](#)

10.264.3.7 `virtual void ArcSec::Request::setRequestItems (ReqItemList) [inline, virtual]`

Set the content of the container

The documentation for this class was generated from the following file:

- Request.h

10.265 ArcSec::RequestAttribute Class Reference

Wrapper which includes [AttributeValue](#) object which is generated according to date type of one spetic node in Request.xml.

```
#include <RequestAttribute.h>
```

Public Member Functions

- [RequestAttribute](#) ([Arc::XMLNode](#) &node, [AttributeFactory](#) *attrfactory)
- [RequestAttribute](#) & [duplicate](#) ([RequestAttribute](#) &)

10.265.1 Detailed Description

Wrapper which includes [AttributeValue](#) object which is generated according to date type of one spetic node in Request.xml.

10.265.2 Constructor & Destructor Documentation

10.265.2.1 ArcSec::RequestAttribute::RequestAttribute (Arc::XMLNode & node, AttributeFactory * attrfactory)

Constructor - create attribute value object according to the "Type" in the node <Attribute attributeid="urn:arc:subject:voms-attribute" type="string">urn:mace:shibboleth:examples</Attribute>

10.265.3 Member Function Documentation

10.265.3.1 RequestAttribute& ArcSec::RequestAttribute::duplicate (RequestAttribute &)

Duplicate the parameter into "this"

The documentation for this class was generated from the following file:

- RequestAttribute.h

10.266 ArcSec::RequestItem Class Reference

Interface for request item container, <subjects, actions, objects, ctxs> tuple.

```
#include <RequestItem.h>
```

Public Member Functions

- [RequestItem](#) ([Arc::XMLNode](#) &, [AttributeFactory](#) *)

10.266.1 Detailed Description

Interface for request item container, <subjects, actions, objects, ctxs> tuple.

10.266.2 Constructor & Destructor Documentation

10.266.2.1 ArcSec::RequestItem::RequestItem (Arc::XMLNode &, AttributeFactory *) [inline]

Constructor

Parameters:

node The XMLNode structure of the request item

attributefactory The [AttributeFactory](#) which will be used to generate [RequestAttribute](#)

The documentation for this class was generated from the following file:

- RequestItem.h

10.267 ArcSec::RequestTuple Class Reference

The documentation for this class was generated from the following file:

- EvaluationCtx.h

10.268 Arc::ResourceType Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.269 ArcSec::Response Class Reference

Container for the evaluation results.

```
#include <Response.h>
```

10.269.1 Detailed Description

Container for the evaluation results.

The documentation for this class was generated from the following file:

- Response.h

10.270 ArcSec::ResponseItem Class Reference

Evaluation result concerning one [RequestTuple](#).

```
#include <Response.h>
```

10.270.1 Detailed Description

Evaluation result concerning one [RequestTuple](#). Include the [RequestTuple](#), related XMLNode, the set of policy objects which give positive evaluation result, and the related XMLNode

The documentation for this class was generated from the following file:

- Response.h

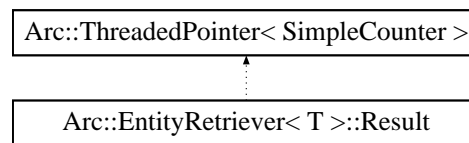
10.271 ArcSec::ResponseList Class Reference

The documentation for this class was generated from the following file:

- Response.h

10.272 Arc::EntityRetriever< T >::Result Class Reference

Inheritance diagram for Arc::EntityRetriever< T >::Result::



template<typename T> class Arc::EntityRetriever< T >::Result

The documentation for this class was generated from the following file:

- EntityRetriever.h

10.273 Arc::Run Class Reference

This class runs an external executable.

```
#include <arc/Run.h>
```

Public Member Functions

- [Run](#) (const std::string &cmdline)
- [Run](#) (const std::list< std::string > &argv)
- [~Run](#) (void)
- [operator bool](#) (void)
- bool [operator!](#) (void)
- bool [Start](#) (void)
- bool [Wait](#) (int timeout)
- bool [Wait](#) (void)
- int [Result](#) (void)
- bool [Running](#) (void)
- [Time RunTime](#) (void)
- [Time ExitTime](#) (void)
- int [ReadStdout](#) (int timeout, char *buf, int size)
- int [ReadStderr](#) (int timeout, char *buf, int size)
- int [WriteStdin](#) (int timeout, const char *buf, int size)
- void [AssignStdout](#) (std::string &str)
- void [AssignStderr](#) (std::string &str)
- void [AssignStdin](#) (std::string &str)
- void [KeepStdout](#) (bool keep=true)
- void [KeepStderr](#) (bool keep=true)
- void [KeepStdin](#) (bool keep=true)
- void [CloseStdout](#) (void)
- void [CloseStderr](#) (void)
- void [CloseStdin](#) (void)
- void [AssignInitializer](#) (void(*initializer_func)(void *), void *initializer_arg)
- void [AssignKicker](#) (void(*kicker_func)(void *), void *kicker_arg)
- void [AssignWorkingDirectory](#) (std::string &wd)
- void [AssignUserId](#) (int uid)
- void [AssignGroupId](#) (int gid)
- void [Kill](#) (int timeout)
- void [Abandon](#) (void)

Static Public Member Functions

- static void [AfterFork](#) (void)

10.273.1 Detailed Description

This class runs an external executable. It is possible to read from or write to its standard handles or to redirect them to std::string elements.

10.273.2 Member Function Documentation

10.273.2.1 void Arc::Run::Abandon (void)

Detach this object from running process. After calling this method instance is not associated with external process anymore. As result destructor will not kill process.

10.273.2.2 static void Arc::Run::AfterFork (void) [static]

Call this method after fork() in child process. It will reinitialize internal structures for new environment. Do not call it in any other case than defined.

10.273.2.3 void Arc::Run::AssignStderr (std::string & str)

Associate stderr handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

10.273.2.4 void Arc::Run::AssignStdin (std::string & str)

Associate stdin handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

10.273.2.5 void Arc::Run::AssignStdout (std::string & str)

Associate stdout handle of executable with string. This method must be called before [Start\(\)](#). str object must be valid as long as this object exists.

10.273.2.6 void Arc::Run::Kill (int timeout)

Kill running executable. First soft kill signal (SIGTERM) is sent to executable. If after timeout seconds executable is still running it's killed completely. Currently this method does not work for Windows OS

10.273.2.7 int Arc::Run::ReadStderr (int timeout, char * buf, int size)

Read from stderr handle of running executable. This method may be used while stderr is directed to string, but the result is unpredictable.

Parameters:

timeout upper limit for which method will block in milliseconds. Negative means infinite.

buf buffer to write the stderr to

size size of buf

Returns:

number of read bytes.

10.273.2.8 int Arc::Run::ReadStdout (int *timeout*, char * *buf*, int *size*)

Read from stdout handle of running executable. This method may be used while stdout is directed to string, but the result is unpredictable.

Parameters:

timeout upper limit for which method will block in milliseconds. Negative means infinite.

buf buffer to write the stdout to

size size of buf

Returns:

number of read bytes.

10.273.2.9 int Arc::Run::Result (void) [inline]

Returns exit code of execution. If child process was killed then exit code is -1. If code is compiled with support for detecting lost child process this code is -1 also if track of child was lost.

10.273.2.10 bool Arc::Run::Start (void)

Starts running executable. This method may be called only once.

Returns:

true if executable started without problems

10.273.2.11 bool Arc::Run::Wait (void)

Wait till execution finished.

Returns:

true if execution is complete, false if execution was not started.

10.273.2.12 bool Arc::Run::Wait (int *timeout*)

Wait till execution finished or till timeout seconds expires.

Returns:

true if execution is complete.

10.273.2.13 int Arc::Run::WriteStdin (int *timeout*, const char * *buf*, int *size*)

Write to stdin handle of running executable. This method may be used while stdin is directed to string, but the result is unpredictable.

Parameters:

timeout upper limit for which method will block in milliseconds. Negative means infinite.

buf buffer to read the stdin from

size size of buf

Returns:

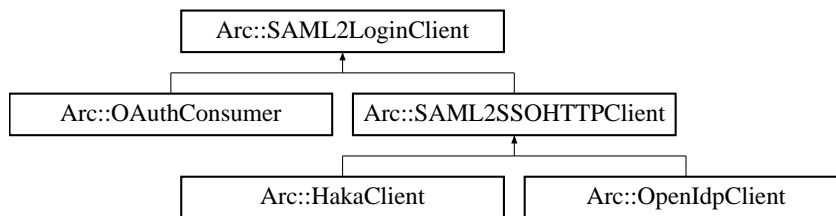
number of written bytes.

The documentation for this class was generated from the following file:

- Run.h

10.274 Arc::SAML2LoginClient Class Reference

Inheritance diagram for Arc::SAML2LoginClient::



Public Member Functions

- [SAML2LoginClient](#) (const [MCCCConfig](#) cfg, const [URL](#) url, std::list< std::string > idp_stack)
- virtual [MCC_Status processLogin](#) (const std::string username="", const std::string password="")=0
- [MCC_Status findSimpleSAMLInstallation](#) ()

10.274.1 Constructor & Destructor Documentation

10.274.1.1 Arc::SAML2LoginClient::SAML2LoginClient (const [MCCCConfig](#) cfg, const [URL](#) url, std::list< std::string > idp_stack)

list with the idp for nested wayf For example, Confusa can use betawayf.wayf.dk as an identity provider, which is itself only a wayf and shares the metadata with concrete service providers or even further nested wayfs. Since due to mutual authentication with metadata, we are obliged to follow the SSO redirects from WAYF to WAYF, the WAYFs are stored in a list.

10.274.2 Member Function Documentation

10.274.2.1 MCC_Status Arc::SAML2LoginClient::findSimpleSAMLInstallation ()

find the location of the simplesamlphp installation on the SP side Will be stored in (*sso_pages)[SimpleSAML]

10.274.2.2 virtual MCC_Status Arc::SAML2LoginClient::processLogin (const std::string username = "", const std::string password = "") [pure virtual]

Base interface for all login procedures

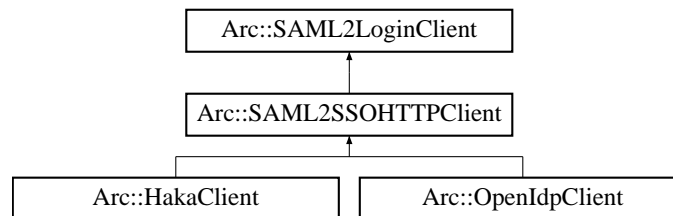
Implemented in [Arc::OAuthConsumer](#), and [Arc::SAML2SSOHTTPClient](#).

The documentation for this class was generated from the following file:

- SAML2LoginClient.h

10.275 Arc::SAML2SSOHTTPClient Class Reference

Inheritance diagram for Arc::SAML2SSOHTTPClient::



Public Member Functions

- [MCC_Status processLogin](#) (const std::string username, const std::string password)
- [MCC_Status parseDN](#) (std::string *dn)
- [MCC_Status approveCSR](#) (const std::string approve_page)
- [MCC_Status pushCSR](#) (const std::string b64_pub_key, const std::string pub_key_hash, std::string *approve_page)
- [MCC_Status storeCert](#) (const std::string cert_path, const std::string auth_token, const std::string b64_dn)

Protected Member Functions

- virtual [MCC_Status processIdPLogin](#) (const std::string username, const std::string password)=0
- virtual [MCC_Status processConsent](#) ()=0
- virtual [MCC_Status processIdP2Confusa](#) ()=0

10.275.1 Member Function Documentation

10.275.1.1 MCC_Status Arc::SAML2SSOHTTPClient::approveCSR (const std::string approve_page) [virtual]

Simulate click on the approve cert signing request link

Implements [Arc::SAML2LoginClient](#).

10.275.1.2 MCC_Status Arc::SAML2SSOHTTPClient::parseDN (std::string * dn) [virtual]

Parse the used DN from the Confusa about_you page

Implements [Arc::SAML2LoginClient](#).

10.275.1.3 virtual MCC_Status Arc::SAML2SSOHTTPClient::processConsent () [protected, pure virtual]

If the IdP has a consent module and the user has not saved her consent, this method will ask the user for consent to transmission of her data to Confusa

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

10.275.1.4 virtual MCC_Status Arc::SAML2SSOHTTPClient::processIdP2Confusa ()
[protected, pure virtual]

Redirects the user back from identity provider to the Confusa SP

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

10.275.1.5 virtual MCC_Status Arc::SAML2SSOHTTPClient::processIdPLogin (const std::string
username, const std::string password) [protected, pure virtual]

Actual identity provider parsers for next three methods implemented in subdirectory idp/

Parse identity provider login page and submit username and password in the provisioned way

Implemented in [Arc::HakaClient](#), and [Arc::OpenIdpClient](#).

10.275.1.6 MCC_Status Arc::SAML2SSOHTTPClient::processLogin (const std::string username,
const std::string password) [virtual]

Models complete SAML2 WebSSO authN flow with start -> WAYF -> Login -> (consent) -> start

Implements [Arc::SAML2LoginClient](#).

10.275.1.7 MCC_Status Arc::SAML2SSOHTTPClient::pushCSR (const std::string b64_pub_key,
const std::string pub_key_hash, std::string * approve_page) [virtual]

Send the cert signing request to Confusa for signing

Implements [Arc::SAML2LoginClient](#).

10.275.1.8 MCC_Status Arc::SAML2SSOHTTPClient::storeCert (const std::string cert_path,
const std::string auth_token, const std::string b64_dn) [virtual]

Download the signed certificate from Confusa and store it locally

Implements [Arc::SAML2LoginClient](#).

The documentation for this class was generated from the following file:

- SAML2LoginClient.h

10.276 Arc::SAMLToken Class Reference

Class for manipulating SAML Token [Profile](#).

```
#include <SAMLToken.h>
```

Public Types

- enum [SAMLVersion](#)

Public Member Functions

- [SAMLToken](#) (SOAPEnvelope &soap)
- [SAMLToken](#) (SOAPEnvelope &soap, const std::string &certfile, const std::string &keyfile, [SAMLVersion](#) saml_version=SAML2, [XMLNode](#) saml_assertion=[XMLNode](#)())
- [~SAMLToken](#) (void)
- [operator bool](#) (void)
- bool [Authenticate](#) (const std::string &cafile, const std::string &capath)
- bool [Authenticate](#) (void)

10.276.1 Detailed Description

Class for manipulating SAML Token [Profile](#). This class is for generating/consuming SAML Token profile. See WS-Security SAML Token [Profile](#) v1.1 (www.oasis-open.org/committees/wss) Currently this class is used by samltoken handler (will appears in src/hed/pdc/samltokensh/) It is not a must to directly called this class. If we need to use SAML Token functionality, we only need to configure the samltoken handler into service and client. Currently, only a minor part of the specification has been implemented.

About how to identify and reference security token for signing message, currently, only the "SAML Assertion Referenced from KeyInfo" (part 3.4.2 of WS-Security SAML Token [Profile](#) v1.1 specification) is supported, which means the implementation can only process SAML assertion "referenced from KeyInfo", and also can only generate SAML Token with SAML assertion "referenced from KeyInfo". More complete support need to implement.

About subject confirmation method, the implementation can process "hold-of-key" (part 3.5.1 of WS-Security SAML Token [Profile](#) v1.1 specification) subject subject confirmation method.

About SAML version, the implementation can process SAML assertion with SAML version 1.1 and 2.0; can only generate SAML assertion with SAML version 2.0.

In the SAML Token profile, for the hold-of-key subject confirmation method, there are three interaction parts: the attesting entity, the relying party and the issuing authority. In the hold-of-key subject confirmation method, it is the attesting entity's subject identity which will be inserted into the SAML assertion.

Firstly the attesting entity authenticates to issuing authority by using some authentication scheme such as WSS x509 Token profile (Alternatively the username/password authentication scheme or other different authentication scheme can also be used, unless the issuing authority can retrieve the key from a trusted certificate server after firmly establishing the subject's identity under the username/password scheme). So then issuing authority is able to make a definitive statement (sign a SAML assertion) about an act of authentication that has already taken place.

The attesting entity gets the SAML assertion and then signs the soap message together with the assertion by using its private key (the relevant certificate has been authenticated by issuing authority, and its relevant public key has been put into SubjectConfirmation element under saml assertion by issuing authority. Only

the actual owner of the saml assertion can do this, as only the subject possesses the private key paired with the public key in the assertion. This establishes an irrefutable connection between the author of the SOAP message and the assertion describing an authentication event.)

The relying party is supposed to trust the issuing authority. When it receives a message from the asserting entity, it will check the saml assertion based on its predetermined trust relationship with the SAML issuing authority, and check the signature of the soap message based on the public key in the saml assertion without directly trust relationship with attesting entity (subject owner).

10.276.2 Member Enumeration Documentation

10.276.2.1 enum Arc::SAMLToken::SAMLVersion

Since the specification SAMLVersion is for distinguishing two types of saml version. It is used as the parameter of constructor.

10.276.3 Constructor & Destructor Documentation

10.276.3.1 Arc::SAMLToken::SAMLToken (SOAPEnvelope & soap)

Constructor. Parse SAML Token information from SOAP header. SAML Token related information is extracted from SOAP header and stored in class variables. And then it the [SAMLToken](#) object will be used for authentication.

Parameters:

soap The SOAP message which contains the [SAMLToken](#) in the soap header

10.276.3.2 Arc::SAMLToken::SAMLToken (SOAPEnvelope & soap, const std::string & certfile, const std::string & keyfile, SAMLVersion saml_version = SAML2, XMLNode saml_assertion = XMLNode ())

Constructor. Add SAML Token information into the SOAP header. Generated token contains elements SAML token and signature, and is meant to be used for authentication on the consuming side. This constructor is for a specific SAML Token profile usage, in which the attesting entity signs the SAML assertion for itself (self-sign). This usage implicitly requires that the relying party trust the attesting entity. More general (requires issuing authority) usage will be provided by other constructor. And the under-developing SAML service will be used as the issuing authority.

Parameters:

soap The SOAP message to which the SAML Token will be inserted.

certfile The certificate file.

keyfile The key file which will be used to create signature.

samlversion The SAML version, only SAML2 is supported currently.

samlassertion The SAML assertion got from 3rd party, and used for protecting the SOAP message; If not present, then self-signed assertion will be generated.

10.276.3.3 Arc::SAMLToken::~~SAMLToken (void)

Deconstructor. Nothing to be done except finalizing the xmlsec library.

10.276.4 Member Function Documentation

10.276.4.1 `bool Arc::SAMLToken::Authenticate (void)`

Check signature by using the cert information in soap message

10.276.4.2 `bool Arc::SAMLToken::Authenticate (const std::string & cafile, const std::string & capath)`

Check signature by using the trusted certificates It is used by relying parting after calling [SAMLToken\(SOAPEnvelope& soap\)](#) This method will check the SAML assertion based on the trusted certificated specified as parameter *cafile* or *capath*; and also check the signature to soap message (the signature is generated by attesting entity by signing soap body together with SAML assertion) by using the public key inside SAML assetion.

Parameters:

cafile ca file

capath ca directory

10.276.4.3 `Arc::SAMLToken::operator bool (void)`

Returns true of constructor succeeded

The documentation for this class was generated from the following file:

- SAMLToken.h

10.277 Arc::ScalableTime< T > Class Template Reference

`template<class T> class Arc::ScalableTime< T >`

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.278 Arc::ScalableTime< int > Class Template Reference

`template<> class Arc::ScalableTime< int >`

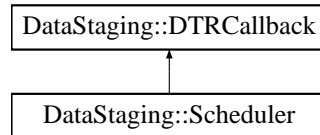
The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.279 DataStaging::Scheduler Class Reference

The [Scheduler](#) is the control centre of the data staging framework.

#include <arc/data-staging/Scheduler.h> Inheritance diagram for DataStaging::Scheduler::



Public Member Functions

- [Scheduler](#) ()
- [~Scheduler](#) ()
- void [SetSlots](#) (int pre_processor=0, int post_processor=0, int delivery=0, int emergency=0, int staged_prepared=0)
- void [AddURLMapping](#) (const [Arc::URL](#) &template_url, const [Arc::URL](#) &replacement_url, const [Arc::URL](#) &access_url=[Arc::URL](#)())
- void [SetURLMapping](#) (const [Arc::URLMap](#) &mapping=[Arc::URLMap](#)())
- void [SetPreferredPattern](#) (const std::string &pattern)
- void [SetTransferSharesConf](#) (const [TransferSharesConf](#) &share_conf)
- void [SetTransferParameters](#) (const [TransferParameters](#) ¶ms)
- void [SetDeliveryServices](#) (const std::vector< [Arc::URL](#) > &endpoints)
- void [SetRemoteSizeLimit](#) (unsigned long long int limit)
- void [SetDumpLocation](#) (const std::string &location)
- bool [start](#) (void)
- virtual void [receiveDTR](#) ([DTR_ptr](#) dtr)
- bool [cancelDTRs](#) (const std::string &jobid)
- bool [stop](#) ()

Static Public Member Functions

- static [Scheduler](#) * [getInstance](#) ()

10.279.1 Detailed Description

The [Scheduler](#) is the control centre of the data staging framework. The [Scheduler](#) manages a global list of DTRs and schedules when they should go into the next state or be sent to other processes. The [DTR](#) priority is used to decide each DTR's position in a queue.

10.279.2 Member Function Documentation

10.279.2.1 static Scheduler* DataStaging::Scheduler::getInstance () [static]

Get static instance of [Scheduler](#), to use one [DTR](#) instance with multiple generators. Configuration of [Scheduler](#) by Set* methods can only be done before [start\(\)](#) is called, so undetermined behaviour can result

from multiple threads simultaneously calling `Set*` then `start()`. It is safer to make sure that all threads use the same configuration (calling `start()` twice is harmless). It is also better to make sure that threads call `stop()` in a roughly coordinated way, i.e. all generators stop at the same time.

10.279.2.2 **virtual void DataStaging::Scheduler::receiveDTR (DTR_ptr *dtr*) [virtual]**

Callback method implemented from [DTRCallback](#). This method is called by the generator when it wants to pass a [DTR](#) to the scheduler and when other processes send a [DTR](#) back to the scheduler after processing.

Implements [DataStaging::DTRCallback](#).

10.279.2.3 **void DataStaging::Scheduler::SetPreferredPattern (const std::string & *pattern*)**

Set the preferred pattern for ordering replicas. This pattern will be used in the case of an index service URL with multiple physical replicas and allows sorting of those replicas in order of preference. It consists of one or more patterns separated by a pipe character (`|`) listed in order of preference. If the dollar character (`$`) is used at the end of a pattern, the pattern will be matched to the end of the hostname of the replica. Example: "srm://myhost.org|.uk\$|.ch\$"

10.279.2.4 **bool DataStaging::Scheduler::start (void)**

Start scheduling activity. This method must be called after all configuration parameters are set properly. [Scheduler](#) can be stopped either by calling `stop()` method or by destroying its instance.

10.279.2.5 **bool DataStaging::Scheduler::stop ()**

Tell the [Scheduler](#) to shut down all threads and exit. All active DTRs are cancelled and this method waits until they finish (all DTRs go to CANCELLED state)

Referenced by `~Scheduler()`.

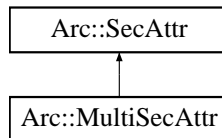
The documentation for this class was generated from the following file:

- Scheduler.h

10.280 Arc::SecAttr Class Reference

This is an abstract interface to a security attribute.

#include <SecAttr.h> Inheritance diagram for Arc::SecAttr::



Public Member Functions

- [SecAttr](#) ()
- bool [operator==](#) (const [SecAttr](#) &b) const
- bool [operator!=](#) (const [SecAttr](#) &b) const
- virtual [operator bool](#) () const
- virtual bool [Export](#) ([SecAttrFormat](#) format, std::string &val) const
- virtual bool [Export](#) ([SecAttrFormat](#) format, [XMLNode](#) &val) const
- virtual bool [Import](#) ([SecAttrFormat](#) format, const std::string &val)
- virtual std::string [get](#) (const std::string &id) const
- virtual std::list< std::string > [getAll](#) (const std::string &id) const

Static Public Attributes

- static [SecAttrFormat](#) [ARCAuth](#)
- static [SecAttrFormat](#) [XACML](#)
- static [SecAttrFormat](#) [SAML](#)
- static [SecAttrFormat](#) [GACL](#)

10.280.1 Detailed Description

This is an abstract interface to a security attribute. This class is meant to be inherited to implement security attributes. Depending on what data it needs to store inheriting classes may need to implement constructor and destructor. They must however override the equality and the boolean operators. The equality is meant to compare security attributes. The prototype implies that all attributes are comparable to all others. This behaviour should be modified as needed by using `dynamic_cast` operations. The boolean cast operation is meant to embody "nullness" if that is applicable to the particular type.

10.280.2 Member Function Documentation

10.280.2.1 virtual bool Arc::SecAttr::Export (SecAttrFormat *format*, XMLNode & *val*) const [virtual]

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute. XML node referenced by is turned into top level element of specified format.

Reimplemented in [Arc::MultiSecAttr](#).

10.280.2.2 virtual bool Arc::SecAttr::Export (SecAttrFormat *format*, std::string & *val*) const [virtual]

Convert internal structure into specified format. Returns false if format is not supported/suitable for this attribute.

10.280.2.3 virtual std::string Arc::SecAttr::get (const std::string & *id*) const [virtual]

Access to specific item of the security attribute. If there are few items of same id the first one is presented. It is meant to be used for tightly coupled SecHandlers and provides more effective interface than Export.

10.280.2.4 virtual std::list<std::string> Arc::SecAttr::getAll (const std::string & *id*) const [virtual]

Access to specific items of the security attribute. This method returns all items which have id assigned. It is meant to be used for tightly coupled SecHandlers and provides more effective interface than Export.

10.280.2.5 virtual bool Arc::SecAttr::Import (SecAttrFormat *format*, const std::string & *val*) [virtual]

Fills internal structure from external object of specified format. Returns false if failed to do. The usage pattern for this method is not defined and it is provided only to make class symmetric. Hence it's implementation is not required yet.

10.280.2.6 virtual Arc::SecAttr::operator bool () const [virtual]

This function should return false if the value is to be considered null, e.g. if it hasn't been set or initialized. In other cases it should return true.

Reimplemented in [Arc::MultiSecAttr](#).

10.280.2.7 bool Arc::SecAttr::operator!= (const SecAttr & *b*) const [inline]

This is a convenience function to allow the usage of "not equal" conditions and need not be overridden.

10.280.2.8 bool Arc::SecAttr::operator== (const SecAttr & *b*) const [inline]

This function should (in inheriting classes) return true if this and *b* are considered to represent same content. Identifying and restricting the type of *b* should be done using `dynamic_cast` operations. Currently it is not defined how comparison methods to be used. Hence their implementation is not required.

The documentation for this class was generated from the following file:

- SecAttr.h

10.281 Arc::SecAttrFormat Class Reference

Export/import format.

```
#include <SecAttr.h>
```

10.281.1 Detailed Description

Export/import format. Format is identified by textual identity string. Class description includes basic formats only. That list may be extended.

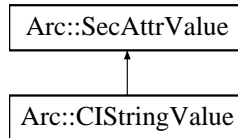
The documentation for this class was generated from the following file:

- SecAttr.h

10.282 Arc::SecAttrValue Class Reference

This is an abstract interface to a security attribute.

`#include <SecAttrValue.h>`Inheritance diagram for Arc::SecAttrValue::



Public Member Functions

- `bool operator== (SecAttrValue &b)`
- `bool operator!= (SecAttrValue &b)`
- `virtual operator bool ()`

10.282.1 Detailed Description

This is an abstract interface to a security attribute. This class is meant to be inherited to implement security attributes. Depending on what data it needs to store inheriting classes may need to implement constructor and destructor. They must however override the equality and the boolean operators. The equality is meant to compare security attributes. The prototype implies that all attributes are comparable to all others. This behaviour should be modified as needed by using `dynamic_cast` operations. The boolean cast operation is meant to embody "nullness" if that is applicable to the particular type.

10.282.2 Member Function Documentation

10.282.2.1 `virtual Arc::SecAttrValue::operator bool ()` [virtual]

This function should return false if the value is to be considered null, e g if it hasn't been set or initialized. In other cases it should return true.

Reimplemented in [Arc::CIStrngValue](#).

10.282.2.2 `bool Arc::SecAttrValue::operator!= (SecAttrValue & b)`

This is a convenience function to allow the usage of "not equal" conditions and need not be overridden.

10.282.2.3 `bool Arc::SecAttrValue::operator== (SecAttrValue & b)`

This function should (in inheriting classes) return true if this and b are considered to be the same. Identifying and restricting the type of b should be done using `dynamic_cast` operations.

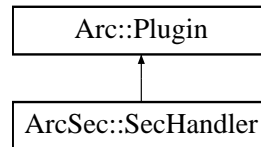
The documentation for this class was generated from the following file:

- `SecAttrValue.h`

10.283 ArcSec::SecHandler Class Reference

Base class for simple security handling plugins.

`#include <SecHandler.h>` Inheritance diagram for ArcSec::SecHandler::



10.283.1 Detailed Description

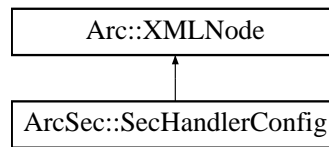
Base class for simple security handling plugins. This virtual class defines method `Handle()` which processes security related information/attributes in `Message` and optionally makes security decision. Instances of such classes are normally arranged in chains and are called on incoming and outgoing messages in various MCC and Service plugins. Return value of `Handle()` defines either processing should continue (true) or stop with error (false). Configuration of [SecHandler](#) is consumed during creation of instance through XML subtree fed to constructor.

The documentation for this class was generated from the following file:

- `SecHandler.h`

10.284 ArcSec::SecHandlerConfig Class Reference

`#include <SecHandler.h>`Inheritance diagram for ArcSec::SecHandlerConfig::



10.284.1 Detailed Description

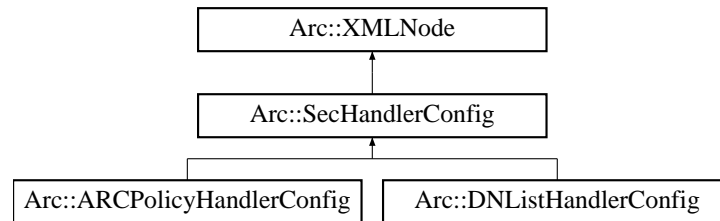
Helper class to create [Security](#) Handler configuration

The documentation for this class was generated from the following file:

- SecHandler.h

10.285 Arc::SecHandlerConfig Class Reference

Inheritance diagram for Arc::SecHandlerConfig::

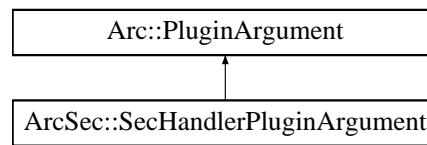


The documentation for this class was generated from the following file:

- ClientInterface.h

10.286 ArcSec::SecHandlerPluginArgument Class Reference

Inheritance diagram for ArcSec::SecHandlerPluginArgument::



The documentation for this class was generated from the following file:

- SecHandler.h

10.287 ArcSec::Security Class Reference

Common stuff used by security related slasses.

```
#include <Security.h>
```

10.287.1 Detailed Description

Common stuff used by security related slasses. This class is just a place where to put common stuff that is used by security related slasses. So far it only contains a logger.

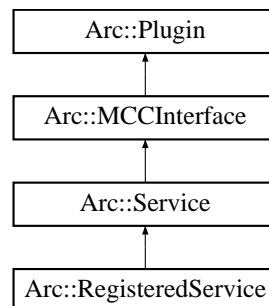
The documentation for this class was generated from the following file:

- Security.h

10.288 Arc::Service Class Reference

[Service](#) - last component in a [Message](#) Chain.

#include <Service.h> Inheritance diagram for Arc::Service::



Public Member Functions

- [Service](#) ([Config](#) *, [PluginArgument](#) *arg)
- virtual void [AddSecHandler](#) ([Config](#) *cfg, [ArcSec::SecHandler](#) *sechandler, const std::string &label="")
- virtual bool [RegistrationCollector](#) ([XMLNode](#) &doc)
- virtual std::string [getID](#) ()
- [operator bool](#) () const
- bool [operator!](#) () const

Protected Member Functions

- bool [ProcessSecHandlers](#) ([Message](#) &message, const std::string &label="") const

Protected Attributes

- std::map< std::string, std::list< [ArcSec::SecHandler](#) * > > [sechandlers_](#)
- bool [valid](#)

Static Protected Attributes

- static [Logger](#) [logger](#)

10.288.1 Detailed Description

[Service](#) - last component in a [Message](#) Chain. This class which defines interface and common functionality for every [Service](#) plugin. Interface is made of method [process\(\)](#) which is called by [Plexer](#) or [MCC](#) class. There is one [Service](#) object created for every service description processed by [Loader](#) class objects. Classes derived from [Service](#) class must implement [process\(\)](#) method of [MCCInterface](#). It is up to developer how internal state of service is stored and communicated to other services and external utilities. [Service](#) is free to expect any type of payload passed to it and generate any payload as well. Useful types depend on MCCs in chain which leads to that service. For example if service is expected to be linked to SOAP [MCC](#) it

must accept and generate messages with [PayloadSOAP](#) payload. Method [process\(\)](#) of class derived from [Service](#) class may be called concurrently in multiple threads. Developers must take that into account and write thread-safe implementation. Simple example of service is provided in `/src/tests/echo/echo.cpp` of source tree. The way to write client counterpart of corresponding service is undefined yet. For example see `/src/tests/echo/test.cpp`.

10.288.2 Constructor & Destructor Documentation

10.288.2.1 Arc::Service::Service (Config *, PluginArgument * arg)

Example constructor - Server takes at least its configuration subtree

10.288.3 Member Function Documentation

10.288.3.1 virtual void Arc::Service::AddSecHandler (Config * cfg, ArcSec::SecHandler * sechandler, const std::string & label = "") [virtual]

Add security components/handlers to this [MCC](#). For more information please see description of [MCC::AddSecHandler](#)

10.288.3.2 virtual std::string Arc::Service::getID () [inline, virtual]

[Service](#) may implement own service identifier gathering method. This method return identifier of service which is used for registering it Information Services.

10.288.3.3 Arc::Service::operator bool (void) const [inline]

Returns true if the [Service](#) is valid.

References valid.

10.288.3.4 bool Arc::Service::operator! (void) const [inline]

Returns true if the [Service](#) is not valid.

References valid.

10.288.3.5 bool Arc::Service::ProcessSecHandlers (Message & message, const std::string & label = "") const [protected]

Executes security handlers of specified queue. For more information please see description of [MCC::ProcessSecHandlers](#)

10.288.3.6 virtual bool Arc::Service::RegistrationCollector (XMLNode & doc) [virtual]

[Service](#) specific registration collector, used for generate service registrations. In implemented service this method should generate [GLUE2](#) document with part of service description which service wishes to advertise to Information Services.

10.288.4 Field Documentation

10.288.4.1 `Logger Arc::Service::logger` `[static, protected]`

[Logger](#) object used to print messages generated by this class.

10.288.4.2 `std::map<std::string,std::list<ArcSec::SecHandler*> > Arc::Service::sechandlers_` `[protected]`

Set of labelled authentication and authorization handlers. [MCC](#) calls sequence of handlers at specific point depending on associated identifier. in most aces those are "in" and "out" for incoming and outgoing messages correspondingly.

10.288.4.3 `bool Arc::Service::valid` `[protected]`

Is service valid? Services which are not valid should set this to false in their constructor.

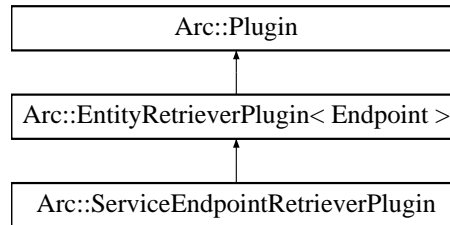
Referenced by operator `bool()`, and operator `!()`.

The documentation for this class was generated from the following file:

- `Service.h`

10.289 Arc::ServiceEndpointRetrieverPlugin Class Reference

Inheritance diagram for Arc::ServiceEndpointRetrieverPlugin::



The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

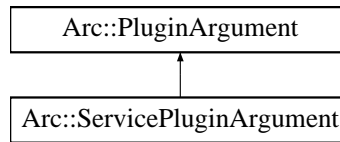
10.290 **Arc::ServiceEndpointRetrieverPluginTESTControl Class Reference**

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.291 Arc::ServicePluginArgument Class Reference

Inheritance diagram for Arc::ServicePluginArgument::



The documentation for this class was generated from the following file:

- Service.h

10.292 Arc::SharedMutex Class Reference

Mutex which allows shared and exclusive locking.

```
#include <arc/Thread.h>
```

Public Member Functions

- void [lockShared](#) (void)
- void [unlockShared](#) (void)
- bool [isLockShared](#) (void)
- void [lockExclusive](#) (void)
- void [unlockExclusive](#) (void)
- bool [isLockExclusive](#) (void)
- void [forceReset](#) (void)

10.292.1 Detailed Description

Mutex which allows shared and exclusive locking.

10.292.2 Member Function Documentation

10.292.2.1 void Arc::SharedMutex::forceReset (void) `[inline]`

This method is meant to be used only after fork. It resets state of all internal locks and variables.

The documentation for this class was generated from the following file:

- Thread.h

10.293 Arc::SimpleCondition Class Reference

Simple triggered condition.

```
#include <arc/Thread.h>
```

Public Member Functions

- void [lock](#) (void)
- void [unlock](#) (void)
- void [signal](#) (void)
- void [signal_nonblock](#) (void)
- void [broadcast](#) (void)
- void [wait](#) (void)
- void [wait_nonblock](#) (void)
- bool [wait](#) (int t)
- void [reset](#) (void)
- void [forceReset](#) (void)

10.293.1 Detailed Description

Simple triggered condition. Provides condition and semaphore objects in one element.

10.293.2 Member Function Documentation

10.293.2.1 void Arc::SimpleCondition::broadcast (void) [inline]

Signal about condition to all waiting threads. If there are no waiting threads, it works like [signal\(\)](#).

10.293.2.2 void Arc::SimpleCondition::forceReset (void) [inline]

This method is meant to be used only after fork. It resets state of all internal locks and variables.

10.293.2.3 void Arc::SimpleCondition::signal (void) [inline]

Signal about condition. This overrides [broadcast\(\)](#).

10.293.2.4 void Arc::SimpleCondition::signal_nonblock (void) [inline]

Signal about condition without using semaphore. Call it **only** with lock acquired.

10.293.2.5 bool Arc::SimpleCondition::wait (int t) [inline]

Wait for condition no longer than t milliseconds.

Returns:

false if timeout occurred

10.293.2.6 void Arc::SimpleCondition::wait_nonblock (void) [inline]

Wait for condition without using semaphor. Call it **only** with lock acquired.

The documentation for this class was generated from the following file:

- Thread.h

10.294 Arc::SimpleCounter Class Reference

Thread-safe counter with capability to wait for zero value.

```
#include <arc/Thread.h>
```

Public Member Functions

- virtual int [inc](#) (void)
- virtual int [dec](#) (void)
- virtual int [get](#) (void) const
- virtual int [set](#) (int v)
- virtual void [wait](#) (void) const
- virtual bool [wait](#) (int t) const
- virtual void [forceReset](#) (void)

10.294.1 Detailed Description

Thread-safe counter with capability to wait for zero value. It is extendible through re-implementation of virtual methods.

10.294.2 Member Function Documentation

10.294.2.1 virtual int Arc::SimpleCounter::dec (void) [virtual]

Decrement value of counter.

Returns:

new value. Does not go below 0 value.

10.294.2.2 virtual void Arc::SimpleCounter::forceReset (void) [inline, virtual]

This method is meant to be used only after fork. It resets state of all internal locks and variables.

10.294.2.3 virtual int Arc::SimpleCounter::get (void) const [virtual]

Returns:

current value of counter.

10.294.2.4 virtual int Arc::SimpleCounter::inc (void) [virtual]

Increment value of counter.

Returns:

new value.

10.294.2.5 `virtual int Arc::SimpleCounter::set (int v)` `[virtual]`

Set value of counter.

Returns:

new value.

10.294.2.6 `virtual bool Arc::SimpleCounter::wait (int t) const` `[virtual]`

Wait for zero condition no longer than *t* milliseconds. If *t* is negative - wait forever.

Returns:

false if timeout occurred.

The documentation for this class was generated from the following file:

- Thread.h

10.295 Arc::SlotRequirementType Class Reference

The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.296 Arc::SOAPMessage Class Reference

[Message](#) restricted to SOAP payload.

```
#include <SOAPMessage.h>
```

Public Member Functions

- [SOAPMessage](#) (void)
- [SOAPMessage](#) (long msg_ptr_addr)
- [SOAPMessage](#) ([Message](#) &msg)
- [~SOAPMessage](#) (void)
- SOAPEnvelope * [Payload](#) (void)
- void [Payload](#) (SOAPEnvelope *new_payload)
- [MessageAttributes](#) * [Attributes](#) (void)

10.296.1 Detailed Description

[Message](#) restricted to SOAP payload. This is a special [Message](#) intended to be used in language bindings for programming languages which are not flexible enough to support all kinds of Payloads. It is passed through chain of MCCs and works like the [Message](#) but can carry only SOAP content.

10.296.2 Constructor & Destructor Documentation

10.296.2.1 Arc::SOAPMessage::SOAPMessage (void) [inline]

Dummy constructor

10.296.2.2 Arc::SOAPMessage::SOAPMessage (long msg_ptr_addr)

Copy constructor. Used by language bindings

10.296.2.3 Arc::SOAPMessage::SOAPMessage (Message & msg)

Copy constructor. Ensures shallow copy.

10.296.2.4 Arc::SOAPMessage::~~SOAPMessage (void)

Destructor does not affect referred objects

10.296.3 Member Function Documentation

10.296.3.1 MessageAttributes* Arc::SOAPMessage::Attributes (void) [inline]

Returns a pointer to the current attributes object or NULL if no attributes object has been assigned.

10.296.3.2 void Arc::SOAPMessage::Payload (SOAPEnvelope * *new_payload*)

Replace payload with a COPY of new one

10.296.3.3 SOAPEnvelope* Arc::SOAPMessage::Payload (void)

Returns pointer to current payload or NULL if no payload assigned.

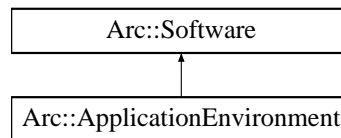
The documentation for this class was generated from the following file:

- SOAPMessage.h

10.297 Arc::Software Class Reference

Used to represent software (names and version) and comparison.

#include <arc/compute/Software.h> Inheritance diagram for Arc::Software::



Public Types

- enum [ComparisonOperatorEnum](#) {
[NOTEQUAL](#) = 0, [EQUAL](#) = 1, [GREATERTHAN](#) = 2, [LESSTHAN](#) = 3,
[GREATERTHANOREQUAL](#) = 4, [LESSTHANOREQUAL](#) = 5 }
- typedef bool(Software::* [ComparisonOperator](#))(const [Software](#) &) const

Public Member Functions

- [Software](#) ()
- [Software](#) (const std::string &name_version)
- [Software](#) (const std::string &name, const std::string &version)
- [Software](#) (const std::string &family, const std::string &name, const std::string &version)
- bool [empty](#) () const
- bool [operator==](#) (const [Software](#) &sw) const
- bool [operator!=](#) (const [Software](#) &sw) const
- bool [operator>](#) (const [Software](#) &sw) const
- bool [operator<](#) (const [Software](#) &sw) const
- bool [operator>=](#) (const [Software](#) &sw) const
- bool [operator<=](#) (const [Software](#) &sw) const
- std::string [operator\(\)](#) () const
- [operator](#) std::string (void) const
- const std::string & [getFamily](#) () const
- const std::string & [getName](#) () const
- const std::string & [getVersion](#) () const

Static Public Member Functions

- static [ComparisonOperator](#) [convert](#) (const [ComparisonOperatorEnum](#) &co)
- static std::string [toString](#) ([ComparisonOperator](#) co)

Static Public Attributes

- static const std::string [VERSIONTOKENS](#)

Friends

- `std::ostream & operator<< (std::ostream &out, const Software &sw)`

10.297.1 Detailed Description

Used to represent software (names and version) and comparison. The [Software](#) class is used to represent the name of a piece of software internally. Generally software are identified by a name and possibly a version number. Some software can also be categorized by type or family (compilers, operating system, etc.). A software object can be compared to other software objects using the comparison operators contained in this class. The basic usage of this class is to test if some specified software requirement ([SoftwareRequirement](#)) are fulfilled, by using the comparability of the class.

Internally the [Software](#) object is represented by a family and name identifier, and the software version is tokenized at the characters defined in `VERSIONTOKENS`, and stored as a list of tokens.

10.297.2 Member Typedef Documentation

10.297.2.1 `typedef bool(Software::* Arc::Software::ComparisonOperator)(const Software &) const`

Definition of a comparison operator method pointer. This `typedef` defines a comparison operator method pointer.

See also:

```
operator==,  
operator!=,  
operator>,  
operator<,  
operator>=,  
operator<=,  
ComparisonOperatorEnum.
```

10.297.3 Member Enumeration Documentation

10.297.3.1 `enum Arc::Software::ComparisonOperatorEnum`

Comparison operator enum. The [ComparisonOperatorEnum](#) enumeration is a 1-1 correspondance between the defined comparison method operators ([Software::ComparisonOperator](#)), and can be used in circumstances where method pointers are not supported.

Enumerator:

```
NOTEQUAL see operator!=  
EQUAL see operator==  
GREATERTHAN see operator>  
LESSTHAN see operator<  
GREATERTHANOREQUAL see operator>=  
LESSTHANOREQUAL see operator<=
```

10.297.4 Constructor & Destructor Documentation

10.297.4.1 `Arc::Software::Software () [inline]`

Dummy constructor. This constructor creates a empty object.

10.297.4.2 `Arc::Software::Software (const std::string & name_version)`

Create a [Software](#) object. Create a [Software](#) object from a single string composed of a name and a version part. The created object will contain a empty family part. The name and version part of the string will be split at the first occurrence of a dash (-) which is followed by a digit (0-9). If the string does not contain such a pattern, the passed string will be taken to be the name and version will be empty.

Parameters:

name_version should be a string composed of the name and version of the software to represent.

10.297.4.3 `Arc::Software::Software (const std::string & name, const std::string & version)`

Create a [Software](#) object. Create a [Software](#) object with the specified name and version. The family part will be left empty.

Parameters:

name the software name to represent.

version the software version to represent.

10.297.4.4 `Arc::Software::Software (const std::string & family, const std::string & name, const std::string & version)`

Create a [Software](#) object. Create a [Software](#) object with the specified family, name and version.

Parameters:

family the software family to represent.

name the software name to represent.

version the software version to represent.

10.297.5 Member Function Documentation

10.297.5.1 `static ComparisonOperator Arc::Software::convert (const ComparisonOperatorEnum & co) [static]`

Convert a [ComparisonOperatorEnum](#) value to a comparison method pointer. The passed [ComparisonOperatorEnum](#) will be converted to a comparison method pointer defined by the [Software::ComparisonOperator](#) typedef.

This static method is not defined in language bindings created with Swig, since method pointers are not supported by Swig.

Parameters:

co a [ComparisonOperatorEnum](#) value.

Returns:

A method pointer to a comparison method is returned.

10.297.5.2 bool Arc::Software::empty () const [inline]

Indicates whether the object is empty.

Returns:

true if the name of this object is empty, otherwise false.

10.297.5.3 const std::string& Arc::Software::getFamily () const [inline]

Get family.

Returns:

The family the represented software belongs to is returned.

10.297.5.4 const std::string& Arc::Software::getName () const [inline]

Get name.

Returns:

The name of the represented software is returned.

10.297.5.5 const std::string& Arc::Software::getVersion () const [inline]

Get version.

Returns:

The version of the represented software is returned.

10.297.5.6 Arc::Software::operator std::string (void) const [inline]

Cast to string. This casting operator behaves exactly as [operator\(\)\(\)](#) does. The cast is used like (std::string) <software-object>.

See also:

[operator\(\)\(\)](#).

References [operator\(\)\(\)](#).

10.297.5.7 bool Arc::Software::operator!= (const Software & *sw*) const [inline]

Inequality operator. The behaviour of the inequality operator is just opposite that of the equality operator ([operator==\(\)](#)).

Parameters:

sw is the RHS [Software](#) object.

Returns:

`true` when the two objects are unequal, otherwise `false`.

References [operator==\(\)](#).

10.297.5.8 std::string Arc::Software::operator() () const

Get string representation. Returns the string representation of this object, which is 'family'-'name'-'version'.

Returns:

The string representation of this object is returned.

See also:

[operator std::string\(\)](#).

Referenced by [operator std::string\(\)](#).

10.297.5.9 bool Arc::Software::operator< (const Software & *sw*) const [inline]

Less-than operator. The behaviour of this less-than operator is equivalent to the greater-than operator ([operator>\(\)](#)) with the LHS and RHS swapped.

Parameters:

sw is the RHS object.

Returns:

`true` if the LHS is less than the RHS, otherwise `false`.

See also:

[operator>\(\)](#).

10.297.5.10 bool Arc::Software::operator<= (const Software & *sw*) const [inline]

Less-than or equal operator. The LHS object is greater than or equal to the RHS object if the LHS equal the RHS ([operator==\(\)](#)) or if the LHS is greater than the RHS ([operator>\(\)](#)).

Parameters:

sw is the RHS object.

Returns:

`true` if the LHS is less than or equal the RHS, otherwise `false`.

See also:

[operator==\(\)](#),
[operator<\(\)](#).

10.297.5.11 bool Arc::Software::operator==(const Software & sw) const [inline]

Equality operator. Two [Software](#) objects are equal only if they are of the same family, have the same name and is of same version. This operator can also be represented by the [Software::EQUAL ComparisonOperatorEnum](#) value.

Parameters:

`sw` is the RHS [Software](#) object.

Returns:

`true` when the two objects equals, otherwise `false`.

Referenced by [operator!==\(\(\)\)](#).

10.297.5.12 bool Arc::Software::operator>(const Software & sw) const

Greater-than operator. For the LHS object to be greater than the RHS object they must first share the same family and name. If the version of the LHS is empty or the LHS and RHS versions equal then LHS is not greater than RHS. If the LHS version is not empty while the RHS is then LHS is greater than RHS. If both versions are non empty and not equal then, the first version token of each object is compared and if they are identical, the two next version tokens will be compared. If not identical, the two tokens will be parsed as integers, and if parsing fails the LHS is not greater than the RHS. If parsing succeeds and the integers equals, the two next tokens will be compared, otherwise the comparison is resolved by the integer comparison.

If the LHS contains more version tokens than the RHS, and the comparison have not been resolved at the point of equal number of tokens, then if the additional tokens contains a token which cannot be parsed to a integer the LHS is not greater than the RHS. If the parsed integer is not 0 then the LHS is greater than the RHS. If the rest of the additional tokens are 0, the LHS is not greater than the RHS.

If the RHS contains more version tokens than the LHS and comparison have not been resolved at the point of equal number of tokens, or simply if comparison have not been resolved at the point of equal number of tokens, then the LHS is not greater than the RHS.

Parameters:

`sw` is the RHS object.

Returns:

`true` if the LHS is greater than the RHS, otherwise `false`.

10.297.5.13 `bool Arc::Software::operator>= (const Software & sw) const` `[inline]`

Greater-than or equal operator. The LHS object is greater than or equal to the RHS object if the LHS equal the RHS (`operator==(())`) or if the LHS is greater than the RHS (`operator>()`).

Parameters:

sw is the RHS object.

Returns:

`true` if the LHS is greater than or equal the RHS, otherwise `false`.

See also:

`operator==(())`,
`operator>()`.

10.297.5.14 `static std::string Arc::Software::toString (ComparisonOperator co)` `[static]`

Convert `Software::ComparisonOperator` to a string. This method is not available in language bindings created by Swig, since method pointers are not supported by Swig.

Parameters:

co is a `Software::ComparisonOperator`.

Returns:

The string representation of the passed `Software::ComparisonOperator` is returned.

10.297.6 Friends And Related Function Documentation**10.297.6.1** `std::ostream& operator<< (std::ostream & out, const Software & sw)` `[friend]`

Write `Software` string representation to a `std::ostream`. Write the string representation of a `Software` object to a `std::ostream`.

Parameters:

out is a `std::ostream` to write the string representation of the `Software` object to.

sw is the `Software` object to write to the `std::ostream`.

Returns:

The passed `std::ostream` *out* is returned.

10.297.7 Field Documentation**10.297.7.1** `const std::string Arc::Software::VERSIONTOKENS` `[static]`

Tokens used to split version string. This string constant specifies which tokens will be used to split the version string.

The documentation for this class was generated from the following file:

- [Software.h](#)

10.298 Arc::SoftwareRequirement Class Reference

Class used to express and resolve version requirements on software.

```
#include <arc/compute/Software.h>
```

Public Member Functions

- [SoftwareRequirement](#) ()
- [SoftwareRequirement](#) (const [Software](#) &sw, [Software::ComparisonOperator](#) swComOp)
- [SoftwareRequirement](#) (const [Software](#) &sw, [Software::ComparisonOperatorEnum](#) co=Software::EQUAL)
- [SoftwareRequirement](#) & operator= (const [SoftwareRequirement](#) &sr)
- [SoftwareRequirement](#) (const [SoftwareRequirement](#) &sr)
- void [add](#) (const [Software](#) &sw, [Software::ComparisonOperator](#) swComOp)
- void [add](#) (const [Software](#) &sw, [Software::ComparisonOperatorEnum](#) co)
- bool [isSatisfied](#) (const [Software](#) &sw) const
- bool [isSatisfied](#) (const std::list< [Software](#) > &swList) const
- bool [isSatisfied](#) (const std::list< [ApplicationEnvironment](#) > &swList) const
- bool [selectSoftware](#) (const [Software](#) &sw)
- bool [selectSoftware](#) (const std::list< [Software](#) > &swList)
- bool [selectSoftware](#) (const std::list< [ApplicationEnvironment](#) > &swList)
- bool [isResolved](#) () const
- bool [empty](#) () const
- void [clear](#) ()
- const std::list< [Software](#) > & [getSoftwareList](#) () const
- const std::list< [Software::ComparisonOperator](#) > & [getComparisonOperatorList](#) () const

10.298.1 Detailed Description

Class used to express and resolve version requirements on software. A requirement in this class is defined as a pair composed of a [Software](#) object and either a [Software::ComparisonOperator](#) method pointer or equally a [Software::ComparisonOperatorEnum](#) enum value. A [SoftwareRequirement](#) object can contain multiple of such requirements, and then it can specified if all these requirements should be satisfied, or if it is enough to satisfy only one of them. The requirements can be satisfied by a single [Software](#) object or a list of either [Software](#) or [ApplicationEnvironment](#) objects, by using the method [isSatisfied\(\)](#). This class also contain a number of methods ([selectSoftware\(\)](#)) to select [Software](#) objects which are satisfying the requirements, and in this way resolving requirements.

10.298.2 Constructor & Destructor Documentation

10.298.2.1 Arc::SoftwareRequirement::SoftwareRequirement () [inline]

Create a empty [SoftwareRequirement](#) object. The created [SoftwareRequirement](#) object will contain no requirements.

10.298.2.2 Arc::SoftwareRequirement::SoftwareRequirement (const Software & *sw*, Software::ComparisonOperator *swComOp*)

Create a [SoftwareRequirement](#) object. The created [SoftwareRequirement](#) object will contain one requirement specified by the [Software](#) object *sw*, and the [Software::ComparisonOperator](#) *swComOp*.

This constructor is not available in language bindings created by Swig, since method pointers are not supported by Swig, see [SoftwareRequirement\(const Software&, Software::ComparisonOperatorEnum\)](#) instead.

Parameters:

sw is the [Software](#) object of the requirement to add.

swComOp is the [Software::ComparisonOperator](#) of the requirement to add.

10.298.2.3 Arc::SoftwareRequirement::SoftwareRequirement (const Software & *sw*, Software::ComparisonOperatorEnum *co* = Software::EQUAL)

Create a [SoftwareRequirement](#) object. The created [SoftwareRequirement](#) object will contain one requirement specified by the [Software](#) object *sw*, and the [Software::ComparisonOperatorEnum](#) *co*.

Parameters:

sw is the [Software](#) object of the requirement to add.

co is the [Software::ComparisonOperatorEnum](#) of the requirement to add.

10.298.2.4 Arc::SoftwareRequirement::SoftwareRequirement (const SoftwareRequirement & *sr*) [inline]

Copy constructor. Create a [SoftwareRequirement](#) object from another [SoftwareRequirement](#) object.

Parameters:

sr is the [SoftwareRequirement](#) object to make a copy of.

10.298.3 Member Function Documentation

10.298.3.1 void Arc::SoftwareRequirement::add (const Software & *sw*, Software::ComparisonOperatorEnum *co*)

Add a [Software](#) object a corresponding comparison operator to this object. Adds software name and version to list of requirements and associates the comparison operator with it (equality by default).

Parameters:

sw is the [Software](#) object to add as part of a requirement.

co is the [Software::ComparisonOperatorEnum](#) value to add as part of a requirement, the default enum will be [Software::EQUAL](#).

10.298.3.2 void Arc::SoftwareRequirement::add (const Software & *sw*, Software::ComparisonOperator *swComOp*)

Add a [Software](#) object a corresponding comparison operator to this object. Adds software name and version to list of requirements and associates the comparison operator with it (equality by default).

This method is not available in language bindings created by Swig, since method pointers are not supported by Swig, see [add\(const Software&, Software::ComparisonOperatorEnum\)](#) instead.

Parameters:

sw is the [Software](#) object to add as part of a requirement.

swComOp is the [Software::ComparisonOperator](#) method pointer to add as part of a requirement, the default operator will be [Software::operator==\(\)](#).

10.298.3.3 void Arc::SoftwareRequirement::clear () [inline]

Clear the object. The requirements in this object will be cleared when invoking this method.

10.298.3.4 bool Arc::SoftwareRequirement::empty () const [inline]

Test if the object is empty.

Returns:

`true` if this object do no contain any requirements, otherwise `false`.

10.298.3.5 const std::list<Software::ComparisonOperator>& Arc::SoftwareRequirement::getComparisonOperatorList () const [inline]

Get list of comparison operators.

Returns:

The list of internally stored comparison operators is returned.

See also:

[Software::ComparisonOperator](#),
[getSoftwareList](#).

10.298.3.6 const std::list<Software>& Arc::SoftwareRequirement::getSoftwareList () const [inline]

Get list of [Software](#) objects.

Returns:

The list of internally stored [Software](#) objects is returned.

See also:

[Software](#),
[getComparisonOperatorList](#).

10.298.3.7 bool Arc::SoftwareRequirement::isResolved () const

Indicates whether requirements have been resolved or not. If specified that only one requirement has to be satisfied, then for this object to be resolved it can only contain one requirement and it has use the equal operator ([Software::operator==](#)).

If specified that all requirements has to be satisfied, then for this object to be resolved each requirement must have a [Software](#) object with a unique family/name composition, i.e. no other requirements have a [Software](#) object with the same family/name composition, and each requirement must use the equal operator ([Software::operator==](#)).

If this object has been resolved then `true` is returned when invoking this method, otherwise `false` is returned.

Returns:

`true` if this object have been resolved, otherwise `false`.

10.298.3.8 bool Arc::SoftwareRequirement::isSatisfied (const std::list< ApplicationEnvironment > & swList) const

Test if requirements are satisfied. This method behaves in exactly the same way as the [isSatisfied\(const Software&\) const](#) method does.

Parameters:

swList is the list of [ApplicationEnvironment](#) objects which should be used to try satisfy the requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

[isSatisfied\(const Software&\) const](#),
[isSatisfied\(const std::list<Software>&\) const](#),
[selectSoftware\(const std::list<ApplicationEnvironment>&\)](#),
[isResolved\(\) const](#).

10.298.3.9 bool Arc::SoftwareRequirement::isSatisfied (const std::list< Software > & swList) const [inline]

Test if requirements are satisfied. Returns `true` if stored requirements are satisfied by software specified in *swList*, otherwise `false` is returned.

Note that if all requirements must be satisfied and multiple requirements exist having identical name and family all these requirements should be satisfied by a single [Software](#) object.

Parameters:

swList is the list of [Software](#) objects which should be used to try satisfy the requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

[isSatisfied\(const Software&\) const](#),
[isSatisfied\(const std::list<ApplicationEnvironment>&\) const](#),
[selectSoftware\(const std::list<Software>&\)](#),
[isResolved\(\) const](#).

10.298.3.10 bool Arc::SoftwareRequirement::isSatisfied (const Software & sw) const [inline]

Test if requirements are satisfied. Returns `true` if the requirements are satisfied by the specified [Software](#) *sw*, otherwise `false` is returned.

Parameters:

sw is the [Software](#) which should satisfy the requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

[isSatisfied\(const std::list<Software>&\) const](#),
[isSatisfied\(const std::list<ApplicationEnvironment>&\) const](#),
[selectSoftware\(const Software&\)](#),
[isResolved\(\) const](#).

References [isSatisfied\(\)](#).

Referenced by [isSatisfied\(\)](#).

10.298.3.11 SoftwareRequirement& Arc::SoftwareRequirement::operator= (const SoftwareRequirement & sr)

Assignment operator. Set this object equal to that of the passed [SoftwareRequirement](#) object *sr*.

Parameters:

sr is the [SoftwareRequirement](#) object to set object equal to.

10.298.3.12 bool Arc::SoftwareRequirement::selectSoftware (const std::list<ApplicationEnvironment > & swList)

Select software. This method behaves exactly as the [selectSoftware\(const std::list<Software>&\)](#) method does.

Parameters:

swList is a list of [ApplicationEnvironment](#) objects used to satisfy requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

```
selectSoftware(const Software&),  
selectSoftware(const std::list<Software>&),  
isSatisfied(const std::list<ApplicationEnvironment>&) const,  
isResolved() const.
```

10.298.3.13 bool Arc::SoftwareRequirement::selectSoftware (const std::list< Software > & swList)

Select software. If the passed list of [Software](#) objects *swList* do not satisfy the requirements `false` is returned and this object is not modified. If however the list of [Software](#) objects *swList* do satisfy the requirements `true` is returned and the [Software](#) objects satisfying the requirements will replace these with the equality operator ([Software::operator==](#)) used as the comparator for the new requirements.

Note that if all requirements must be satisfied and multiple requirements exist having identical name and family all these requirements should be satisfied by a single [Software](#) object and it will replace all these requirements.

Parameters:

swList is a list of [Software](#) objects used to satisfy requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

```
selectSoftware(const Software&),  
selectSoftware(const std::list<ApplicationEnvironment>&),  
isSatisfied(const std::list<Software>&) const,  
isResolved() const.
```

10.298.3.14 bool Arc::SoftwareRequirement::selectSoftware (const Software & sw) [inline]

Select software. If the passed [Software](#) *sw* do not satisfy the requirements `false` is returned and this object is not modified. If however the [Software](#) object *sw* do satisfy the requirements `true` is returned and the requirements are set to equal the *sw* [Software](#) object.

Parameters:

sw is the [Software](#) object used to satisfy requirements.

Returns:

`true` if requirements are satisfied, otherwise `false`.

See also:

[selectSoftware\(const std::list<Software>&\),](#)
[selectSoftware\(const std::list<ApplicationEnvironment>&\),](#)
[isSatisfied\(const Software&\) const,](#)
[isResolved\(\) const.](#)

References [selectSoftware\(\)](#).

Referenced by [selectSoftware\(\)](#).

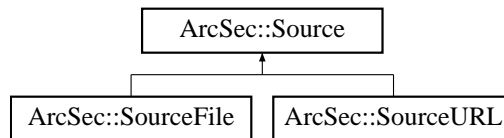
The documentation for this class was generated from the following file:

- [Software.h](#)

10.299 ArcSec::Source Class Reference

Acquires and parses XML document from specified source.

#include <Source.h>Inheritance diagram for ArcSec::Source::



Public Member Functions

- [Source](#) (const [Source](#) &s)
- [Source](#) ([Arc::XMLNode](#) xml)
- [Source](#) (std::istream &stream)
- [Source](#) ([Arc::URL](#) &url)
- [Source](#) (const std::string &str)
- [Arc::XMLNode Get](#) (void) const
- [operator bool](#) (void)

10.299.1 Detailed Description

Acquires and parses XML document from specified source. This class is to be used to provide easy way to specify different sources for XML Authorization Policies and Requests.

10.299.2 Constructor & Destructor Documentation

10.299.2.1 ArcSec::Source::Source (const Source & s) [inline]

Copy constructor. Use this constructor only for temporary objects. Parsed XML document is still owned by copied source and hence lifetime of created object should not exceed that of copied one.

10.299.2.2 ArcSec::Source::Source (Arc::XMLNode xml)

Use XML subtree referred by xml. There is no copy of xml made. Hence lifetime of this object should not exceed that of xml.

10.299.2.3 ArcSec::Source::Source (Arc::URL & url)

Fetch XML document from specified url and parse it. This constructor is not implemented yet.

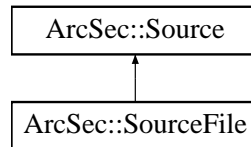
The documentation for this class was generated from the following file:

- Source.h

10.300 ArcSec::SourceFile Class Reference

Convenience class for obtaining XML document from file.

`#include <Source.h>`Inheritance diagram for ArcSec::SourceFile::



Public Member Functions

- [SourceFile](#) (const [SourceFile](#) &s)
- [SourceFile](#) (const char *name)
- [SourceFile](#) (const std::string &name)

10.300.1 Detailed Description

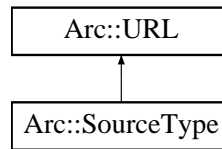
Convenience class for obtaining XML document from file.

The documentation for this class was generated from the following file:

- Source.h

10.301 Arc::SourceType Class Reference

Inheritance diagram for Arc::SourceType::



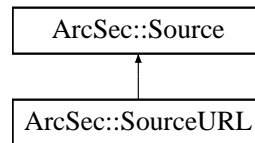
The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.302 ArcSec::SourceURL Class Reference

Convenience class for obtaining XML document from remote URL.

#include <Source.h>Inheritance diagram for ArcSec::SourceURL::



Public Member Functions

- [SourceURL](#) (const [SourceURL](#) &s)
- [SourceURL](#) (const char *url)
- [SourceURL](#) (const std::string &url)

10.302.1 Detailed Description

Convenience class for obtaining XML document from remote URL.

The documentation for this class was generated from the following file:

- Source.h

10.303 DataStaging::DataDeliveryComm::Status Struct Reference

Plain C struct to pass information from executing process back to main thread.

```
#include <DataDeliveryComm.h>
```

Data Fields

- [CommStatusType](#) `commstatus`
- `time_t` `timestamp`
- [DTRStatus::DTRStatusType](#) `status`
- [DTRErrorStatus::DTRErrorStatusType](#) `error`
- [DTRErrorStatus::DTRErrorLocation](#) `error_location`
- `char` [error_desc](#) [1024]
- `unsigned int` [streams](#)
- `unsigned long long int` [transferred](#)
- `unsigned long long int` [offset](#)
- `unsigned long long int` [size](#)
- `unsigned int` [speed](#)
- `char` [checksum](#) [128]

10.303.1 Detailed Description

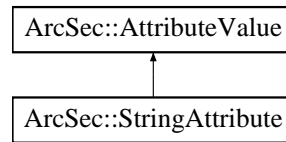
Plain C struct to pass information from executing process back to main thread.

The documentation for this struct was generated from the following file:

- `DataDeliveryComm.h`

10.304 ArcSec::StringAttribute Class Reference

Inheritance diagram for ArcSec::StringAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.304.1 Member Function Documentation

10.304.1.1 virtual std::string ArcSec::StringAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.304.1.2 virtual std::string ArcSec::StringAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.304.1.3 virtual std::string ArcSec::StringAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- StringAttribute.h

10.305 Arc::SubmissionStatus Class Reference

The documentation for this class was generated from the following file:

- SubmissionStatus.h

10.306 Arc::Submitter Class Reference

Data Structures

- class **ConsumerWrapper**

The documentation for this class was generated from the following file:

- Submitter.h

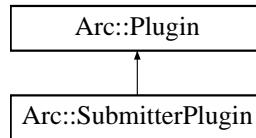
10.307 Arc::SubmitterPlugin Class Reference

Base class for the SubmitterPlugins.

```
#include <arc/compute/SubmitterPlugin.h>
Arc::SubmitterPlugin::
```

diagram

for



Public Member Functions

- virtual [SubmissionStatus](#) [Submit](#) (const [JobDescription](#) &j, const [ExecutionTarget](#) &et, [EntityConsumer](#)< [Job](#) > &jc)
- virtual [SubmissionStatus](#) [Submit](#) (const std::list< [JobDescription](#) > &jobdesc, const [ExecutionTarget](#) &et, [EntityConsumer](#)< [Job](#) > &jc, std::list< const [JobDescription](#) * > ¬Submitted)=0
- virtual bool [Migrate](#) (const std::string &jobid, const [JobDescription](#) &jobdesc, const [ExecutionTarget](#) &et, bool forcemigration, [Job](#) &job)

10.307.1 Detailed Description

Base class for the SubmitterPlugins. [SubmitterPlugin](#) is the base class for Grid middleware specialized [SubmitterPlugin](#) objects. The class submits job(s) to the computing resource it represents and uploads (needed by the job) local input files.

10.307.2 Member Function Documentation

10.307.2.1 virtual bool [Arc::SubmitterPlugin::Migrate](#) (const std::string &*jobid*, const [JobDescription](#) &*jobdesc*, const [ExecutionTarget](#) &*et*, bool *forcemigration*, [Job](#) &*job*) [**virtual**]

Migrate job. This virtual method should be overridden by plugins which should be capable of migrating jobs. The active job which should be migrated is pointed to by the [URL](#) *jobid*, and is represented by the [JobDescription](#) *jobdesc*. The forcemigration boolean specifies if the migration should succeed if the active job cannot be terminated. The protected method `AddJob` can be used to save job information. This method should return the [URL](#) of the migrated job. In case migration fails an empty [URL](#) should be returned.

10.307.2.2 virtual [SubmissionStatus](#) [Arc::SubmitterPlugin::Submit](#) (const std::list< [JobDescription](#) > &*jobdesc*, const [ExecutionTarget](#) &*et*, [EntityConsumer](#)< [Job](#) > &*jc*, std::list< const [JobDescription](#) * > &*notSubmitted*) [**pure virtual**]

Submit job. This virtual method should be overridden by plugins which should be capable of submitting jobs, defined in the [JobDescription](#) *jobdesc*, to the [ExecutionTarget](#) *et*. The protected convenience method `AddJob` can be used to save job information. This method should return the [URL](#) of the submitted job. In case submission fails an empty [URL](#) should be returned.

10.307.2.3 `virtual SubmissionStatus Arc::SubmitterPlugin::Submit (const JobDescription & j,
const ExecutionTarget & et, EntityConsumer< Job > & jc) [inline, virtual]`

Submit a single job description. Convenience method for submitting single job description, it simply calls the [SubmitterPlugin::Submit](#) method taking a list of job descriptions.

Parameters:

- j* [JobDescription](#) object to be submitted.
- et* [ExecutionTarget](#) to submit the job description to.
- jc* callback object used to add [Job](#) object of newly submitted job to.

Returns:

a bool indicating whether job submission succeeded or not.

References [Submit\(\)](#).

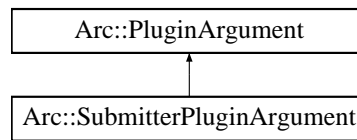
Referenced by [Submit\(\)](#).

The documentation for this class was generated from the following file:

- [SubmitterPlugin.h](#)

10.308 Arc::SubmitterPluginArgument Class Reference

Inheritance diagram for Arc::SubmitterPluginArgument::



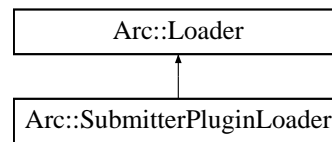
The documentation for this class was generated from the following file:

- [SubmitterPlugin.h](#)

10.309 Arc::SubmitterPluginLoader Class Reference

```
#include <arc/compute/SubmitterPlugin.h>
Arc::SubmitterPluginLoader::
```

diagram for



Public Member Functions

- [SubmitterPluginLoader](#) ()
- [~SubmitterPluginLoader](#) ()
- [SubmitterPlugin * load](#) (const std::string &name, const [UserConfig](#) &usercfg)

10.309.1 Detailed Description

Class responsible for loading [SubmitterPlugin](#) plugins The [SubmitterPlugin](#) objects returned by a [SubmitterPluginLoader](#) must not be used after the [SubmitterPluginLoader](#) is destroyed.

10.309.2 Constructor & Destructor Documentation

10.309.2.1 Arc::SubmitterPluginLoader::SubmitterPluginLoader ()

Constructor Creates a new [SubmitterPluginLoader](#).

10.309.2.2 Arc::SubmitterPluginLoader::~~SubmitterPluginLoader ()

Destructor Calling the destructor destroys all [SubmitterPlugins](#) loaded by the [SubmitterPluginLoader](#) instance.

10.309.3 Member Function Documentation

10.309.3.1 SubmitterPlugin* Arc::SubmitterPluginLoader::load (const std::string & name, const UserConfig & usercfg)

Load a new [SubmitterPlugin](#)

Parameters:

name The name of the [SubmitterPlugin](#) to load.

usercfg The [UserConfig](#) object for the new [SubmitterPlugin](#).

Returns:

A pointer to the new [SubmitterPlugin](#) (NULL on error).

The documentation for this class was generated from the following file:

- [SubmitterPlugin.h](#)

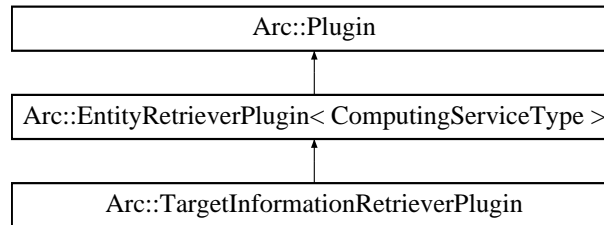
10.310 Arc::SubmitterPluginTestACCControl Class Reference

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.311 Arc::TargetInformationRetrieverPlugin Class Reference

Inheritance diagram for Arc::TargetInformationRetrieverPlugin::



The documentation for this class was generated from the following file:

- [EntityRetrieverPlugin.h](#)

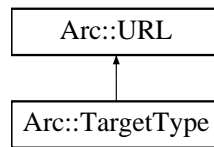
10.312 **Arc::TargetInformationRetrieverPluginTESTControl** Class Reference

The documentation for this class was generated from the following file:

- [TestACCControl.h](#)

10.313 Arc::TargetType Class Reference

Inheritance diagram for Arc::TargetType::



The documentation for this class was generated from the following file:

- [JobDescription.h](#)

10.314 Arc::TCPSec Class Reference

The documentation for this class was generated from the following file:

- ClientInterface.h

10.315 Arc::EntityRetriever< T >::ThreadArg Class Reference

template<typename T> class Arc::EntityRetriever< T >::ThreadArg

The documentation for this class was generated from the following file:

- EntityRetriever.h

10.316 Arc::ThreadDataItem Class Reference

Base class for per-thread object.

```
#include <arc/Thread.h>
```

Public Member Functions

- [ThreadDataItem](#) (void)
- [ThreadDataItem](#) (std::string &key)
- [ThreadDataItem](#) (const std::string &key)
- void [Attach](#) (std::string &key)
- void [Attach](#) (const std::string &key)
- virtual void [Dup](#) (void)

Static Public Member Functions

- static [ThreadDataItem](#) * [Get](#) (const std::string &key)

10.316.1 Detailed Description

Base class for per-thread object. Classes inherited from this one are attached to current thread under specified key and destroyed only when thread ends or object is replaced by another one with same key.

10.316.2 Constructor & Destructor Documentation

10.316.2.1 Arc::ThreadDataItem::ThreadDataItem (void)

Dummy constructor which does nothing. To make object usable one of the [Attach\(...\)](#) methods must be used.

10.316.2.2 Arc::ThreadDataItem::ThreadDataItem (std::string & key)

Creates instance and attaches it to current thread under key. If supplied key is empty random one is generated and stored in key variable.

10.316.3 Member Function Documentation

10.316.3.1 void Arc::ThreadDataItem::Attach (const std::string & key)

Attaches object to current thread under key. This method must be used only if object was created using dummy constructor.

10.316.3.2 void Arc::ThreadDataItem::Attach (std::string & key)

Attaches object to current thread under key. If supplied key is empty random one is generated and stored in key variable. This method must be used only if object was created using dummy constructor.

10.316.3.3 virtual void Arc::ThreadDataItem::Dup (void) [virtual]

Creates copy of object. This method is called when a new thread is created from the current thread. It is called in the new thread, so the new object - if created - gets attached to the new thread. If the object is not meant to be inherited by new threads then this method should do nothing.

10.316.3.4 static ThreadDataItem* Arc::ThreadDataItem::Get (const std::string & key) [static]

Retrieves object attached to thread under key.

Returns:

NULL if no such object.

The documentation for this class was generated from the following file:

- Thread.h

10.317 Arc::ThreadedPointer< T > Class Template Reference

Wrapper for pointer with automatic destruction and multiple references.

```
#include <arc/Thread.h>
```

Public Member Functions

- T & [operator*](#) (void) const
- T * [operator->](#) (void) const
- [operator bool](#) (void) const
- bool [operator!](#) (void) const
- bool [operator==](#) (const [ThreadedPointer](#) &p) const
- bool [operator!=](#) (const [ThreadedPointer](#) &p) const
- bool [operator<](#) (const [ThreadedPointer](#) &p) const
- T * [Ptr](#) (void) const
- T * [Release](#) (void)
- unsigned int [Holders](#) (void)
- unsigned int [WaitOutOfRange](#) (unsigned int minThr, unsigned int maxThr)
- unsigned int [WaitOutOfRange](#) (unsigned int minThr, unsigned int maxThr, int timeout)
- unsigned int [WaitInRange](#) (unsigned int minThr, unsigned int maxThr)
- unsigned int [WaitInRange](#) (unsigned int minThr, unsigned int maxThr, int timeout)

10.317.1 Detailed Description

template<typename T> class Arc::ThreadedPointer< T >

Wrapper for pointer with automatic destruction and multiple references. See for [CountedPointer](#) for description. Differently from [CountedPointer](#) this class provides thread safe destruction of referred object. But the instance of [ThreadedPointer](#) itself is not thread safe. Hence it is advisable to use different instances in different threads.

10.317.2 Member Function Documentation

10.317.2.1 **template<typename T> T* Arc::ThreadedPointer< T >::Release (void) [inline]**

Release referred object so that it can be passed to other container. After [Release\(\)](#) is called referred object is will not be destroyed automatically anymore.

10.317.2.2 **template<typename T> unsigned int Arc::ThreadedPointer< T >::WaitInRange (unsigned int *minThr*, unsigned int *maxThr*, int *timeout*) [inline]**

Waits till number of [ThreadedPointer](#) instances \geq *minThr* and \leq *maxThr*. Waits no longer than *timeout* milliseconds. If *timeout* is negative - wait forever. Returns current number of instances.

**10.317.2.3 template<typename T> unsigned int Arc::ThreadedPointer< T >::WaitOutOfRange
 (unsigned int *minThr*, unsigned int *maxThr*, int *timeout*) [inline]**

Waits till number of [ThreadedPointer](#) instances \leq minThr or \geq maxThr. Waits no longer than timeout milliseconds. If timeout is negative - wait forever. Returns current number of instances.

The documentation for this class was generated from the following file:

- Thread.h

10.318 Arc::ThreadInitializer Class Reference

This class initializes the glibmm thread system.

```
#include <Thread.h>
```

Public Member Functions

- [ThreadInitializer](#) (void)
- void [forceReset](#) (void)
- void [waitExit](#) (void)

10.318.1 Detailed Description

This class initializes the glibmm thread system.

10.318.2 Member Function Documentation

10.318.2.1 void Arc::ThreadInitializer::forceReset (void)

This method is meant to be used only after fork. It resets state of all internal locks and variables.

10.318.2.2 void Arc::ThreadInitializer::waitExit (void)

Wait for all known threads to exit. It can be used before exiting application to make sure no concurrent threads are running during cleanup.

The documentation for this class was generated from the following file:

- Thread.h

10.319 Arc::ThreadRegistry Class Reference

A set of conditions, mutexes, etc. conveniently exposed to monitor running child threads and to wait till they exit.

```
#include <arc/Thread.h>
```

Public Member Functions

- void [RegisterThread](#) (void)
- void [UnregisterThread](#) (void)
- bool [WaitOrCancel](#) (int timeout)
- bool [WaitForExit](#) (int timeout=-1)
- void [RequestCancel](#) (void)
- void [forceReset](#) (void)

10.319.1 Detailed Description

A set of conditions, mutexes, etc. conveniently exposed to monitor running child threads and to wait till they exit. There are no protections against race conditions, so use it carefully.

10.319.2 Member Function Documentation

10.319.2.1 void Arc::ThreadRegistry::forceReset (void) `[inline]`

This method is meant to be used only after fork. It resets state of all internal locks and variables.

10.319.2.2 bool Arc::ThreadRegistry::WaitForExit (int *timeout* = -1)

Wait for registered threads to exit. Leave after timeout milliseconds if failed.

Returns:

true if all registered threads reported their exit.

10.319.2.3 bool Arc::ThreadRegistry::WaitOrCancel (int *timeout*)

Wait for timeout milliseconds or cancel request.

Returns:

true if cancel request received.

The documentation for this class was generated from the following file:

- Thread.h

10.320 Arc::Time Class Reference

A class for storing and manipulating times.

```
#include <arc/DateTime.h>
```

Public Member Functions

- [Time](#) ()
- [Time](#) (time_t)
- [Time](#) (time_t time, uint32_t nanosec)
- [Time](#) (const std::string &)
- [Time](#) & [operator=](#) (time_t)
- [Time](#) & [operator=](#) (const [Time](#) &)
- [Time](#) & [operator=](#) (const char *)
- [Time](#) & [operator=](#) (const std::string &)
- void [SetTime](#) (time_t)
- void [SetTime](#) (time_t time, uint32_t nanosec)
- time_t [GetTime](#) () const
- time_t [GetTimeNanoseconds](#) () const
- [operator std::string](#) () const
- std::string [str](#) (const [TimeFormat](#) &=time_format) const
- bool [operator<](#) (const [Time](#) &) const
- bool [operator>](#) (const [Time](#) &) const
- bool [operator<=](#) (const [Time](#) &) const
- bool [operator>=](#) (const [Time](#) &) const
- bool [operator==](#) (const [Time](#) &) const
- bool [operator!=](#) (const [Time](#) &) const
- [Time](#) [operator+](#) (const [Period](#) &) const
- [Time](#) [operator-](#) (const [Period](#) &) const
- [Period](#) [operator-](#) (const [Time](#) &) const

Static Public Member Functions

- static void [SetFormat](#) (const [TimeFormat](#) &)
- static [TimeFormat](#) [GetFormat](#) ()

Static Public Attributes

- static const int [YEAR](#) = 31536000
- static const int [MONTH](#) = 2592000
- static const int [WEEK](#) = 604800
- static const int [DAY](#) = 86400
- static const int [HOUR](#) = 3600
- static const time_t [UNDEFINED](#) = (time_t)(-1)

10.320.1 Detailed Description

A class for storing and manipulating times. [Time](#) represents a moment of time (eg midnight on 1st Jan 2000), whereas [Period](#) represents a length of time (eg 2 mins and 30.1 seconds).

See also:

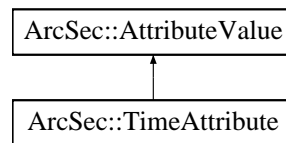
[Period](#)

The documentation for this class was generated from the following file:

- [DateTime.h](#)

10.321 ArcSec::TimeAttribute Class Reference

#include <DateTimeAttribute.h> Inheritance diagram for ArcSec::TimeAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.321.1 Detailed Description

Format: HHMMSSZ HH:MM:SS HH:MM:SS+HH:MM HH:MM:SSZ

10.321.2 Member Function Documentation

10.321.2.1 virtual std::string ArcSec::TimeAttribute::encode () [virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.321.2.2 virtual std::string ArcSec::TimeAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.321.2.3 virtual std::string ArcSec::TimeAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- DateTimeAttribute.h

10.322 Arc::TimedMutex Class Reference

Mutex which allows a timeout on locking.

```
#include <arc/Thread.h>
```

Public Member Functions

- bool [lock](#) (int t=-1)
- bool [trylock](#) (void)
- bool [unlock](#) (void)
- void [forceReset](#) (void)

10.322.1 Detailed Description

Mutex which allows a timeout on locking.

10.322.2 Member Function Documentation

10.322.2.1 void Arc::TimedMutex::forceReset (void) [inline]

This method is meant to be used only after fork. It resets state of all internal locks and variables.

10.322.2.2 bool Arc::TimedMutex::lock (int t = -1) [inline]

Lock mutex, but wait no longer than t milliseconds.

Returns:

false if timeout occurred.

Referenced by trylock().

The documentation for this class was generated from the following file:

- Thread.h

10.323 DataStaging::TransferParameters Class Reference

Represents limits and properties of a [DTR](#) transfer. These generally apply to all DTRs.

```
#include <arc/data-staging/DTR.h>
```

Public Member Functions

- [TransferParameters](#) ()

Data Fields

- unsigned long long int [min_average_bandwidth](#)
- unsigned int [max_inactivity_time](#)
- unsigned long long int [min_current_bandwidth](#)
- unsigned int [averaging_time](#)

10.323.1 Detailed Description

Represents limits and properties of a [DTR](#) transfer. These generally apply to all DTRs.

10.323.2 Field Documentation

10.323.2.1 unsigned int DataStaging::TransferParameters::max_inactivity_time

Maximum inactivity time in sec. If transfer stops for longer than this time it will be killed.

10.323.2.2 unsigned long long int DataStaging::TransferParameters::min_average_bandwidth

Minimum average bandwidth in bytes/sec. If the average bandwidth used over the whole transfer drops below this level the transfer will be killed.

10.323.2.3 unsigned long long int DataStaging::TransferParameters::min_current_bandwidth

Minimum current bandwidth in bytes/sec. If bandwidth averaged over the previous `averaging_time` seconds is less than `min_current_bandwidth` the transfer will be killed (allows transfers which slow down to be killed quicker).

The documentation for this class was generated from the following file:

- `DTR.h`

10.324 DataStaging::TransferShares Class Reference

[TransferShares](#) is used to implement fair-sharing and priorities.

```
#include <arc/data-staging/TransferShares.h>
```

Public Member Functions

- [TransferShares](#) ()
- [TransferShares](#) (const [TransferSharesConf](#) &shares_conf)
- [~TransferShares](#) ()
- void [set_shares_conf](#) (const [TransferSharesConf](#) &share_conf)
- void [calculate_shares](#) (int TotalNumberOfSlots)
- void [increase_transfer_share](#) (const std::string &ShareToIncrease)
- void [decrease_transfer_share](#) (const std::string &ShareToDecrease)
- void [decrease_number_of_slots](#) (const std::string &ShareToDecrease)
- bool [can_start](#) (const std::string &ShareToStart)
- std::map< std::string, int > [active_shares](#) () const

10.324.1 Detailed Description

[TransferShares](#) is used to implement fair-sharing and priorities. [TransferShares](#) defines the algorithm used to prioritise and share transfers among different users or groups. Configuration information on the share type and reference shares is held in a [TransferSharesConf](#) instance. The [Scheduler](#) uses [TransferShares](#) to determine which DTRs in the queue for each process go first. The calculation is based on the configuration and the currently active shares (the DTRs already in the process). [can_start\(\)](#) is the method called by the [Scheduler](#) to determine whether a particular share has an available slot in the process.

10.324.2 Member Function Documentation

10.324.2.1 void DataStaging::TransferShares::calculate_shares (int *TotalNumberOfSlots*)

Calculate how many slots to assign to each active share. This method is called each time the [Scheduler](#) loops to calculate the number of slots to assign to each share, based on the current number of active shares and the shares' relative priorities.

10.324.2.2 void DataStaging::TransferShares::decrease_number_of_slots (const std::string & *ShareToDecrease*)

Decrease by one the number of slots available to the given share. Called when there is a slot already used by this share to reduce the number available.

The documentation for this class was generated from the following file:

- [TransferShares.h](#)

10.325 DataStaging::TransferSharesConf Class Reference

[TransferSharesConf](#) describes the configuration of [TransferShares](#).

```
#include <arc/data-staging/TransferShares.h>
```

Public Types

- enum [ShareType](#) {
[USER](#), [VO](#), [GROUP](#), [ROLE](#),
[NONE](#) }

Public Member Functions

- [TransferSharesConf](#) (const std::string &type, const std::map< std::string, int > &ref_shares)
- [TransferSharesConf](#) ()
- void [set_share_type](#) (const std::string &type)
- void [set_reference_share](#) (const std::string &RefShare, int Priority)
- void [set_reference_shares](#) (const std::map< std::string, int > &shares)
- bool [is_configured](#) (const std::string &ShareToCheck)
- int [get_basic_priority](#) (const std::string &ShareToCheck)
- std::string [conf](#) () const
- std::string [extract_share_info](#) ([DTR_ptr](#) DTRToExtract)

10.325.1 Detailed Description

[TransferSharesConf](#) describes the configuration of [TransferShares](#). It allows reference shares to be defined with certain priorities. An instance of this class is used when creating a [TransferShares](#) object.

10.325.2 Member Enumeration Documentation

10.325.2.1 enum DataStaging::TransferSharesConf::ShareType

The criterion for assigning a share to a [DTR](#).

Enumerator:

- USER*** Shares are defined per DN of the user's proxy.
- VO*** Shares are defined per VOMS VO of the user's proxy.
- GROUP*** Shares are defined per VOMS group of the user's proxy.
- ROLE*** Shares are defined per VOMS role of the user's proxy.
- NONE*** No share criterion - all DTRs will be assigned to a single share.

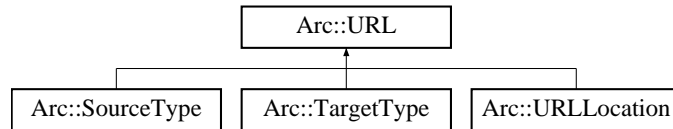
The documentation for this class was generated from the following file:

- [TransferShares.h](#)

10.326 Arc::URL Class Reference

Class to represent general URLs.

`#include <arc/URL.h>`Inheritance diagram for Arc::URL::



Public Types

- enum [Scope](#)

Public Member Functions

- [URL](#) ()
- [URL](#) (const std::string &url, bool encoded=false)
- virtual [~URL](#) ()
- void [URIDecode](#) (void)
- const std::string & [Protocol](#) () const
- void [ChangeProtocol](#) (const std::string &newprot)
- const std::string & [Username](#) () const
- const std::string & [Passwd](#) () const
- const std::string & [Host](#) () const
- void [ChangeHost](#) (const std::string &newhost)
- int [Port](#) () const
- void [ChangePort](#) (int newport)
- const std::string & [Path](#) () const
- std::string [FullPath](#) () const
- std::string [FullPathURIEncoded](#) () const
- void [ChangePath](#) (const std::string &newpath)
- void [ChangeFullPath](#) (const std::string &newpath, bool encoded=false)
- const std::map< std::string, std::string > & [HTTPOptions](#) () const
- const std::string & [HTTPOption](#) (const std::string &option, const std::string &undefined="") const
- bool [AddHTTPOption](#) (const std::string &option, const std::string &value, bool overwrite=true)
- void [RemoveHTTPOption](#) (const std::string &option)
- const std::list< std::string > & [LDAPAttributes](#) () const
- void [AddLDAPAttribute](#) (const std::string &attribute)
- [Scope](#) [LDAPScope](#) () const
- void [ChangeLDAPScope](#) (const [Scope](#) newscope)
- const std::string & [LDAPFilter](#) () const
- void [ChangeLDAPFilter](#) (const std::string &newfilter)
- const std::map< std::string, std::string > & [Options](#) () const
- const std::string & [Option](#) (const std::string &option, const std::string &undefined="") const
- const std::map< std::string, std::string > & [MetaDataOptions](#) () const
- const std::string & [MetaDataOption](#) (const std::string &option, const std::string &undefined="") const

- bool [AddOption](#) (const std::string &option, const std::string &value, bool overwrite=true)
- bool [AddOption](#) (const std::string &option, bool overwrite=true)
- void [AddMetaDataOption](#) (const std::string &option, const std::string &value, bool overwrite=true)
- void [AddLocation](#) (const [URLLocation](#) &location)
- const std::list< [URLLocation](#) > & [Locations](#) () const
- const std::map< std::string, std::string > & [CommonLocOptions](#) () const
- const std::string & [CommonLocOption](#) (const std::string &option, const std::string &undefined="") const
- void [RemoveOption](#) (const std::string &option)
- void [RemoveMetaDataOption](#) (const std::string &option)
- virtual std::string [str](#) (bool encode=false) const
- virtual std::string [plainstr](#) (bool encode=false) const
- virtual std::string [fullstr](#) (bool encode=false) const
- virtual std::string [ConnectionURL](#) () const
- bool [operator<](#) (const [URL](#) &url) const
- bool [operator==](#) (const [URL](#) &url) const
- [operator bool](#) () const
- bool [operator!](#) () const
- bool [StringMatches](#) (const std::string &str) const
- std::map< std::string, std::string > [ParseOptions](#) (const std::string &optstring, char separator, bool encoded=false)

Static Public Member Functions

- static std::string [OptionString](#) (const std::map< std::string, std::string > &options, char separator, bool encode=false)
- static std::string [URIEncode](#) (const std::string &str)
- static std::string [URIDecode](#) (const std::string &str)

Protected Member Functions

- void [ParsePath](#) (bool encoded=false)

Static Protected Member Functions

- static std::string [BaseDN2Path](#) (const std::string &)
- static std::string [Path2BaseDN](#) (const std::string &)

Protected Attributes

- std::string [protocol](#)
- std::string [username](#)
- std::string [passwd](#)
- std::string [host](#)
- bool [ip6addr](#)
- int [port](#)
- std::string [path](#)
- std::map< std::string, std::string > [httpoptions](#)
- std::map< std::string, std::string > [metadataoptions](#)

- `std::list< std::string >` [ldapattributes](#)
- [Scope](#) `ldapscope`
- `std::string` [ldapfilter](#)
- `std::map< std::string, std::string >` [urloptions](#)
- `std::list< URLLocation >` [locations](#)
- `std::map< std::string, std::string >` [commonlocoptions](#)
- `bool` [valid](#)

Friends

- `std::ostream &` [operator<<](#) (`std::ostream &out`, `const URL &u`)

10.326.1 Detailed Description

Class to represent general URLs. The [URL](#) is split into protocol, hostname, port and path. This class tries to follow RFC 3986 for splitting URLs, at least for protocol + host part. It also accepts local file paths which are converted to `file:path`. The usual system dependent file paths are supported. Relative paths are converted to absolute paths by prepending them with current working directory path. A file path can't start from # symbol. If the string representation of [URL](#) starts from '@' then it is treated as path to a file containing a list of URLs.

A [URL](#) is parsed in the following way:

```
[protocol:] [//[username:passwd@] [host] [:port]] [;urloptions[;...]] [/path[?httpoption[&...]] [:metadataoption[&...]]]
```

The 'protocol' and 'host' parts are treated as case-insensitive and to avoid confusion are converted to lowercase in constructor. Note that 'path' is always converted to absolute path in the constructor. The meaning of 'absolute' may depend upon [URL](#) type. For generic [URL](#) and local POSIX file paths that means the path starts from / like

```
/path/to/file
```

For Windows paths the absolute path may look like

```
C:\path\to\file
```

It is important to note that path still can be empty. For referencing a local file using an absolute path on a POSIX filesystem one may use either

```
file:///path/to/file or file:/path/to/file
```

The relative path will look like

```
file:to/file
```

For local Windows files possible URLs are

```
%file:C:\path\to\file or %file:to\file
```

URLs representing LDAP resources have a different structure of options following the 'path' part:

```
ldap://host[:port] [;urloptions[;...]] [/path[?attributes[?scope[?filter]]]]
```

For LDAP URLs paths are converted from /key1=value1/.../keyN=valueN notation to keyN=valueN,...,key1=value1 and hence path does not contain a leading /. If an LDAP [URL](#) initially had its path in the second notation, the leading / is treated as a separator only and is stripped.

URLs of indexing services optionally may have locations specified before the 'host' part

```
protocol://[location[;location[;...]]@[host][:port]...
```

The structure of the 'location' element is protocol specific.

10.326.2 Member Function Documentation

10.326.2.1 `bool Arc::URL::AddHTTPOption (const std::string & option, const std::string & value, bool overwrite = true)`

Adds a HTP option with the given value.

Returns:

false if overwrite is false and option already exists, true otherwise.

10.326.2.2 `bool Arc::URL::AddOption (const std::string & option, bool overwrite = true)`

Adds a [URL](#) option where option has the format "name=value".

Returns:

false if overwrite is true and option already exists or if option does not have the correct format. Returns true otherwise.

10.326.2.3 `bool Arc::URL::AddOption (const std::string & option, const std::string & value, bool overwrite = true)`

Adds a [URL](#) option with the given value. Note that some compilers may interpret `AddOption("name", "value")` as a call to `AddOption(const std::string&, bool)` so it is recommended to use explicit string types when calling this method.

Returns:

false if overwrite is false and option already exists, true otherwise.

10.326.2.4 `const std::string& Arc::URL::CommonLocOption (const std::string & option, const std::string & undefined = "") const`

Returns the value of a common location option.

Parameters:

option The option whose value is returned.

undefined This value is returned if the common location option is not defined.

10.326.2.5 std::string Arc::URL::FullPathURIEncoded () const

Returns the path and all options, URI-encoded according to RFC 3986. Forward slashes (‘/’) in the path are not encoded but are encoded in the options.

10.326.2.6 const std::string& Arc::URL::HTTPOption (const std::string & *option*, const std::string & *undefined* = "") const

Returns the value of an HTTP option.

Parameters:

option The option whose value is returned.

undefined This value is returned if the HTTP option is not defined.

10.326.2.7 const std::string& Arc::URL::MetaDataOption (const std::string & *option*, const std::string & *undefined* = "") const

Returns the value of a metadata option.

Parameters:

option The option whose value is returned.

undefined This value is returned if the metadata option is not defined.

10.326.2.8 const std::string& Arc::URL::Option (const std::string & *option*, const std::string & *undefined* = "") const

Returns the value of a [URL](#) option.

Parameters:

option The option whose value is returned.

undefined This value is returned if the [URL](#) option is not defined.

10.326.2.9 static std::string Arc::URL::OptionString (const std::map< std::string, std::string > & *options*, char *separator*, bool *encode* = false) [static]

Returns a string representation of the options given in the options map.

Parameters:

encode if set to true then options are encoded according to RFC 3986

10.326.2.10 void Arc::URL::RemoveHTTPOption (const std::string & *option*)

Removes a HTTP option if exists.

Parameters:

option The option to remove.

10.326.2.11 void Arc::URL::RemoveMetaDataOption (const std::string & *option*)

Remove a metadata option if exists.

Parameters:

option The option to remove.

10.326.2.12 void Arc::URL::RemoveOption (const std::string & *option*)

Removes a [URL](#) option if exists.

Parameters:

option The option to remove.

10.326.2.13 static std::string Arc::URL::URIDecode (const std::string & *str*) [static]

Perform decoding according to RFC 3986. This simply calls [Arc::uri_unencode\(\)](#).

10.326.2.14 void Arc::URL::URIDecode (void)

Perform decoding of stored [URL](#) parts according to RFC 3986. This method is supposed to be used only if for some reason [URL](#) constructor was called with encoded=false for [URL](#) which was encoded. Use it only once.

10.326.2.15 static std::string Arc::URL::URIEncode (const std::string & *str*) [static]

Perform encoding according to RFC 3986. This simply calls [Arc::uri_encode\(\)](#).

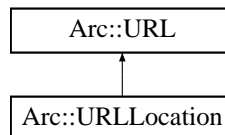
The documentation for this class was generated from the following file:

- [URL.h](#)

10.327 Arc::URLLocation Class Reference

Class to hold a resolved [URL](#) location.

`#include <arc/URL.h>`Inheritance diagram for Arc::URLLocation::



Public Member Functions

- [URLLocation](#) (const std::string &url="")
- [URLLocation](#) (const std::string &url, const std::string &name)
- [URLLocation](#) (const [URL](#) &url)
- [URLLocation](#) (const [URL](#) &url, const std::string &name)
- [URLLocation](#) (const std::map< std::string, std::string > &options, const std::string &name)
- virtual [~URLLocation](#) ()
- const std::string & [Name](#) () const
- virtual std::string [str](#) () const
- virtual std::string [fullstr](#) () const

Protected Attributes

- std::string [name](#)

10.327.1 Detailed Description

Class to hold a resolved [URL](#) location. It is specific to file indexing service registrations.

The documentation for this class was generated from the following file:

- [URL.h](#)

10.328 Arc::User Class Reference

Platform independent representation of system user.

```
#include <arc/User.h>
```

Public Member Functions

- [User](#) ()
- [User](#) (const std::string &name, const std::string &group="")
- [User](#) (int uid, int gid=-1)
- [operator bool](#) () const
- bool [operator!](#) () const
- const std::string & [Name](#) (void) const
- const std::string & [Home](#) (void) const
- int [get_uid](#) (void) const
- int [get_gid](#) (void) const
- bool [operator==](#) (const std::string &n)
- int [check_file_access](#) (const std::string &path, int flags) const
- bool [SwitchUser](#) () const

10.328.1 Detailed Description

Platform independent representation of system user.

10.328.2 Constructor & Destructor Documentation

10.328.2.1 Arc::User::User (const std::string & *name*, const std::string & *group* = "")

Construct user from username and optional group name. If group is not specified it is determined automatically.

10.328.2.2 Arc::User::User (int *uid*, int *gid* = -1)

Construct user from uid and optional gid. If gid is not specified it is determined automatically.

10.328.3 Member Function Documentation

10.328.3.1 int Arc::User::check_file_access (const std::string & *path*, int *flags*) const

Check if this [User](#) has the rights specified by flags on the given path.

Returns:

0 if [User](#) has the rights

10.328.3.2 bool Arc::User::SwitchUser () const

Change the owner of the current process. Internally this method calls `setuid()` and `setgid()` with this User's values. It can be used in the initializer of [Arc::Run](#) to switch the owner of a child process just after `fork()`. To temporarily change the owner of a thread in a multi-threaded environment [UserSwitch](#) should be used instead.

Returns:

true if switch succeeded.

The documentation for this class was generated from the following file:

- User.h

10.329 Arc::UserConfig Class Reference

User configuration class

```
#include <arc/UserConfig.h>
```

Public Member Functions

- [UserConfig](#) ([initializeCredentialsType](#) initializeCredentials=[initializeCredentialsType](#)())
- [UserConfig](#) (const std::string &conffile, [initializeCredentialsType](#) initializeCredentials=[initializeCredentialsType](#)(), bool loadSysConfig=true)
- [UserConfig](#) (const std::string &conffile, const std::string &jfile, [initializeCredentialsType](#) initializeCredentials=[initializeCredentialsType](#)(), bool loadSysConfig=true)
- [UserConfig](#) (const long int &ptraddr)
- bool [InitializeCredentials](#) ([initializeCredentialsType](#) initializeCredentials)
- bool [CredentialsFound](#) () const
- bool [LoadConfigurationFile](#) (const std::string &conffile, bool ignoreJobListFile=true)
- bool [SaveToFile](#) (const std::string &filename) const
- void [ApplyToConfig](#) ([BaseConfig](#) &ccfg) const
- [operator bool](#) () const
- bool [operator!](#) () const
- bool [JobListFile](#) (const std::string &path)
- const std::string & [JobListFile](#) () const
- bool [Timeout](#) (int newTimeout)
- int [Timeout](#) () const
- bool [Verbosity](#) (const std::string &newVerbosity)
- const std::string & [Verbosity](#) () const
- bool [Broker](#) (const std::string &name)
- bool [Broker](#) (const std::string &name, const std::string &argument)
- const std::pair< std::string, std::string > & [Broker](#) () const
- bool [Bartender](#) (const std::vector< [URL](#) > &urls)
- void [AddBartender](#) (const [URL](#) &url)
- const std::vector< [URL](#) > & [Bartender](#) () const
- bool [VOMSESPath](#) (const std::string &path)
- const std::string & [VOMSESPath](#) ()
- bool [UserName](#) (const std::string &name)
- const std::string & [UserName](#) () const
- bool [Password](#) (const std::string &newPassword)
- const std::string & [Password](#) () const
- bool [ProxyPath](#) (const std::string &newProxyPath)
- const std::string & [ProxyPath](#) () const
- bool [CertificatePath](#) (const std::string &newCertificatePath)
- const std::string & [CertificatePath](#) () const
- bool [KeyPath](#) (const std::string &newKeyPath)
- const std::string & [KeyPath](#) () const
- bool [KeyPassword](#) (const std::string &newKeyPassword)
- const std::string & [KeyPassword](#) () const
- bool [KeySize](#) (int newKeySize)
- int [KeySize](#) () const
- bool [CACertificatePath](#) (const std::string &newCACertificatePath)

- const std::string & [CACertificatePath](#) () const
- bool [CACertificatesDirectory](#) (const std::string &newCACertificatesDirectory)
- const std::string & [CACertificatesDirectory](#) () const
- bool [CertificateLifeTime](#) (const [Period](#) &newCertificateLifeTime)
- const [Period](#) & [CertificateLifeTime](#) () const
- bool [SLCS](#) (const [URL](#) &newSLCS)
- const [URL](#) & [SLCS](#) () const
- bool [StoreDirectory](#) (const std::string &newStoreDirectory)
- const std::string & [StoreDirectory](#) () const
- bool [JobDownloadDirectory](#) (const std::string &newDownloadDirectory)
- const std::string & [JobDownloadDirectory](#) () const
- bool [IdPName](#) (const std::string &name)
- const std::string & [IdPName](#) () const
- bool [OverlayFile](#) (const std::string &path)
- const std::string & [OverlayFile](#) () const
- bool [UtilsDirPath](#) (const std::string &dir)
- const std::string & [UtilsDirPath](#) () const
- void [SetUser](#) (const [User](#) &u)
- const [User](#) & [GetUser](#) () const
- bool [InfoInterface](#) (const std::string &infointerface_)
- const std::string & [InfoInterface](#) () const
- bool [SubmissionInterface](#) (const std::string &submissioninterface_)
- const std::string & [SubmissionInterface](#) () const
- const std::list< std::string > & [RejectDiscoveryURLs](#) () const
- void [AddRejectDiscoveryURLs](#) (const std::list< std::string > &urls)
- void [ClearRejectDiscoveryURLs](#) ()
- const std::list< std::string > & [RejectManagementURLs](#) () const
- [ConfigEndpoint](#) [GetService](#) (const std::string &alias)
- std::list< [ConfigEndpoint](#) > [GetServicesInGroup](#) (const std::string &group, [ConfigEndpoint::Type](#) type=[ConfigEndpoint::ANY](#))
- std::list< [ConfigEndpoint](#) > [GetDefaultServices](#) ([ConfigEndpoint::Type](#) type=[ConfigEndpoint::ANY](#))
- std::list< [ConfigEndpoint](#) > [GetServices](#) (const std::string &groupOrAlias, [ConfigEndpoint::Type](#) type=[ConfigEndpoint::ANY](#))
- std::map< std::string, [ConfigEndpoint](#) > [GetAllConfiguredServices](#) ()

Static Public Attributes

- static const std::string [ARCUSERDIRECTORY](#)
- static const std::string [SYSCONFIG](#)
- static const std::string [SYSCONFIGARCLOC](#)
- static const std::string [DEFAULTCONFIG](#)
- static const std::string [EXAMPLECONFIG](#)
- static const int [DEFAULT_TIMEOUT](#) = 20
- static const std::string [DEFAULT_BROKER](#)

10.329.1 Detailed Description

User configuration class This class provides a container for a selection of various attributes/parameters which can be configured to needs of the user, and can be read by implementing instances or programs. The class can be used in two ways. One can create a object from a configuration file, or simply set the desired attributes by using the setter method, associated with every setable attribute. The list of attributes which can be configured in this class are:

- certificatepath / [CertificatePath\(const std::string&\)](#)
- keypath / [KeyPath\(const std::string&\)](#)
- proxypath / [ProxyPath\(const std::string&\)](#)
- cacertificatesdirectory / [CACertificatesDirectory\(const std::string&\)](#)
- cacertificatepath / [CACertificatePath\(const std::string&\)](#)
- timeout / [Timeout\(int\)](#)
- joblist / [JobListFile\(const std::string&\)](#)
- verbosity / [Verbosity\(const std::string&\)](#)
- brokername / [Broker\(const std::string&\)](#) or [Broker\(const std::string&, const std::string&\)](#)
- brokerarguments / [Broker\(const std::string&\)](#) or [Broker\(const std::string&, const std::string&\)](#)
- bartender / [Bartender\(const std::list<URL>&\)](#)
- vomsserverpath / [VOMSESPath\(const std::string&\)](#)
- username / [UserName\(const std::string&\)](#)
- password / [Password\(const std::string&\)](#)
- keypassword / [KeyPassword\(const std::string&\)](#)
- keysize / [KeySize\(int\)](#)
- certificatelifetime / [CertificateLifeTime\(const Period&\)](#)
- slcs / [SLCS\(const URL&\)](#)
- storedirectory / [StoreDirectory\(const std::string&\)](#)
- jobdownloadaddirectory / [JobDownloadDirectory\(const std::string&\)](#)
- idpname / [IdPName\(const std::string&\)](#)
- submissioninterface / [SubmissionInterface\(const std::string&\)](#)
- infointerface / [InfoInterface\(const std::string&\)](#)

where the first term is the name of the attribute used in the configuration file, and the second term is the associated setter method (for more information about a given attribute see the description of the setter method).

The configuration file should have a INI-style format and the [IniConfig](#) class will thus be used to parse the file. The above mentioned attributes should be placed in the common section.

Besides the options above, the configuration file can contain information about services (service registries and computing elements). Each service has to be put in its on section. Each service has an alias, which is a

short name. The name of the section consists of the word ‘registry’ for service registries and ‘computing’ for computing elements, then contains a slash and the alias of the service. e.g. ‘[registry/index1]’ or ‘[computing/testce]’. In a service section the possible options are the following:

- url: is the url of the service
- default: if yes, then this service will be used if no other is specified
- group: assigns the service to a group with a given name

For computing elements the following additional options exist:

- infointerface: the [GLUE2](#) InterfaceName of the local information system
- submissioninterface: the [GLUE2](#) InterfaceName to the job submission interface

For a service registry the following additional option exist:

- registryinterface: the [GLUE2](#) InterfaceName of the service registry interface

These services can be accessed by the [GetService](#), [GetServices](#), [GetDefaultServices](#), [GetServicesInGroup](#) methods, which return [ConfigEndpoint](#) object(s). The [ConfigEndpoint](#) objects contain the [URL](#) and the [InterfaceNames](#) of the services.

The [UserConfig](#) class also provides a method [InitializeCredentials\(\)](#) for locating user credentials by searching in different standard locations. The [CredentialsFound\(\)](#) method can be used to test if locating the credentials succeeded.

10.329.2 Constructor & Destructor Documentation

10.329.2.1 Arc::UserConfig::UserConfig (initializeCredentialsType *initializeCredentials* = initializeCredentialsType ())

Create a [UserConfig](#) object. The [UserConfig](#) object created by this constructor initializes only default values, and if specified by the *initializeCredentials* boolean credentials will be tried initialized using the [InitializeCredentials\(\)](#) method. The object is only non-valid if initialization of credentials fails which can be checked with the [operator bool\(\)](#) method.

Parameters:

initializeCredentials is a optional boolean indicating if the [InitializeCredentials\(\)](#) method should be invoked, the default is `true`.

See also:

[InitializeCredentials\(\)](#)
[operator bool\(\)](#)

10.329.2.2 Arc::UserConfig::UserConfig (const std::string & *conffile*, initializeCredentialsType *initializeCredentials* = initializeCredentialsType (), bool *loadSysConfig* = `true`)

Create a [UserConfig](#) object. The [UserConfig](#) object created by this constructor will, if specified by the *loadSysConfig* boolean, first try to load the system configuration file by invoking the [LoadConfigurationFile\(\)](#) method, and if this fails a **WARNING** is reported. Then the configuration file passed will be

tried loaded using the before mentioned method, and if this fails an [ERROR](#) is reported, and the created object will be non-valid. Note that if the passed file path is empty the example configuration will be tried copied to the default configuration file path specified by DEFAULTCONFIG. If the example file cannot be copied one or more [WARNING](#) messages will be reported and no configuration will be loaded. If loading the configurations file succeeded and if *initializeCredentials* is `true` then credentials will be initialized using the [InitializeCredentials\(\)](#) method, and if no valid credentials are found the created object will be non-valid.

Parameters:

conffile is the path to a INI-configuration file.

initializeCredentials is a boolean indicating if credentials should be initialized, the default is `true`.

loadSysConfig is a boolean indicating if the system configuration file should be loaded aswell, the default is `true`.

See also:

[LoadConfigurationFile\(const std::string&, bool\)](#)
[InitializeCredentials\(\)](#)
[operator bool\(\)](#)
[SYSCONFIG](#)
[EXAMPLECONFIG](#)

10.329.2.3 [Arc::UserConfig::UserConfig](#) (const std::string & *conffile*, const std::string & *jfile*, initializeCredentialsType *initializeCredentials* = initializeCredentialsType (), bool *loadSysConfig* = `true`)

Create a [UserConfig](#) object. The [UserConfig](#) object created by this constructor does only differ from the [UserConfig\(const std::string&, bool, bool\)](#) constructor in that it is possible to pass the path of the job list file directly to this constructor. If the job list file *joblistfile* is empty, the behaviour of this constructor is exactly the same as the before mentioned, otherwise the job list file will be initilized by invoking the setter method [JobListFile\(const std::string&\)](#). If it fails the created object will be non-valid, otherwise the specified configuration file *conffile* will be loaded with the *ignoreJobListFile* argument set to `true`.

Parameters:

conffile is the path to a INI-configuration file

jfile is the path to a (non-)existing job list file.

initializeCredentials is a boolean indicating if credentials should be initialized, the default is `true`.

loadSysConfig is a boolean indicating if the system configuration file should be loaded aswell, the default is `true`.

See also:

[JobListFile\(const std::string&\)](#)
[LoadConfigurationFile\(const std::string&, bool\)](#)
[InitializeCredentials\(\)](#)
[operator bool\(\)](#)

10.329.2.4 Arc::UserConfig::UserConfig (const long int & ptraddr)

Language binding constructor. The passed long int should be a pointer address to a [UserConfig](#) object, and this address is then casted into this [UserConfig](#) object.

Parameters:

ptraddr is an memory address to a [UserConfig](#) object.

10.329.3 Member Function Documentation

10.329.3.1 void Arc::UserConfig::AddBartender (const URL & url) [inline]

Set bartenders, used to contact Chelonia. Takes as input a Bartender [URL](#) and adds this to the list of bartenders.

Parameters:

url is a [URL](#) to be added to the list of bartenders.

See also:

[Bartender\(const std::list<URL>&\)](#)
[Bartender\(\) const](#)

10.329.3.2 void Arc::UserConfig::AddRejectDiscoveryURLs (const std::list< std::string > & urls) [inline]

Add list of URLs to ignored at service discovery. The passed list of strings will be added to the internal reject list and they should represent URLs which should be ignored when doing service discovery.

Parameters:

urls list of string representing URLs to ignore at service discovery

10.329.3.3 void Arc::UserConfig::ApplyToConfig (BaseConfig & ccfg) const

Apply credentials to [BaseConfig](#). This methods sets the [BaseConfig](#) credentials to the credentials contained in this object. It also passes user defined configuration overlay if any.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\)](#)
[BaseConfig](#)

Parameters:

ccfg a [BaseConfig](#) object which will configured with the credentials of this object.

10.329.3.4 `const std::vector<URL>& Arc::UserConfig::Bartender () const` `[inline]`

Get bartenders. Returns a list of Bartender URLs

Returns:

The list of bartender [URL](#) objects is returned.

See also:

[Bartender\(const std::list<URL>&\)](#)
[AddBartender\(const URL&\)](#)

10.329.3.5 `bool Arc::UserConfig::Bartender (const std::vector< URL > & urls)` `[inline]`

Set bartenders, used to contact Chelonia. Takes as input a vector of Bartender URLs.

The attribute associated with this setter method is 'bartender'.

Parameters:

urls is a list of [URL](#) object to be set as bartenders.

Returns:

This method always returns `true`.

See also:

[AddBartender\(const URL&\)](#)
[Bartender\(\) const](#)

10.329.3.6 `const std::pair<std::string, std::string>& Arc::UserConfig::Broker () const` `[inline]`

Get the broker and corresponding arguments. The returned pair contains the broker name as the first component and the argument as the second.

See also:

[Broker\(const std::string&\)](#)
[Broker\(const std::string&, const std::string&\)](#)
[DEFAULT_BROKER](#)

10.329.3.7 `bool Arc::UserConfig::Broker (const std::string & name, const std::string & argument)` `[inline]`

Set broker to use in target matching. As opposed to the [Broker\(const std::string&\)](#) method this method sets broker name and arguments directly from the passed two arguments.

Two attributes are associated with this setter method 'brokername' and 'brokerarguments'.

Parameters:

name is the name of the broker.

argument is the arguments of the broker.

Returns:

This method always returns `true`.

See also:

[Broker](#)
[Broker\(const std::string&\)](#)
[Broker\(\) const](#)
[DEFAULT_BROKER](#)

10.329.3.8 bool Arc::UserConfig::Broker (const std::string & name)

Set broker to use in target matching. The string passed to this method should be in the format:

`< name > [:< argument >]`

where the `<name>` is the name of the broker and cannot contain any `'.'`, and the optional `<argument>` should contain arguments which should be passed to the broker.

Two attributes are associated with this setter method `'brokername'` and `'brokerarguments'`.

Parameters:

name the broker name and argument specified in the format given above.

Returns:

This method always returns `true`.

See also:

[Broker](#)
[Broker\(const std::string&, const std::string&\)](#)
[Broker\(\) const](#)
[DEFAULT_BROKER](#)

10.329.3.9 const std::string& Arc::UserConfig::CACertificatePath () const [inline]

Get path to CA-certificate. Retrieve the path to the file containing CA-certificate. This configuration parameter is deprecated.

Returns:

The path to the CA-certificate is returned.

See also:

[CACertificatePath\(const std::string&\)](#)

10.329.3.10 `bool Arc::UserConfig::CACertificatePath (const std::string & newCACertificatePath) [inline]`

Set CA-certificate path. The path to the file containing CA-certificate will be set when calling this method. This configuration parameter is deprecated - use CACertificatesDirectory instead. Only arcslcs uses it.

The attribute associated with this setter method is 'cacertificatepath'.

Parameters:

newCACertificatePath is the path to the CA-certificate.

Returns:

This method always returns `true`.

See also:

[CACertificatePath\(\) const](#)

10.329.3.11 `const std::string& Arc::UserConfig::CACertificatesDirectory () const [inline]`

Get path to CA-certificate directory. Retrieve the path to the CA-certificate directory.

Returns:

The path to the CA-certificate directory is returned.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CACertificatesDirectory\(const std::string&\)](#)

10.329.3.12 `bool Arc::UserConfig::CACertificatesDirectory (const std::string & newCACertificatesDirectory) [inline]`

Set path to CA-certificate directory. The path to the directory containing CA-certificates will be set when calling this method. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'cacertificatesdirectory'.

Parameters:

newCACertificatesDirectory is the path to the CA-certificate directory.

Returns:

This method always returns `true`.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CACertificatesDirectory\(\) const](#)

10.329.3.13 `const Period& Arc::UserConfig::CertificateLifeTime () const` `[inline]`

Get certificate life time. Gets lifetime of user certificate which will be obtained from Short Lived Credentials [Service](#).

Returns:

The certificate life time is returned as a [Period](#) object.

See also:

[CertificateLifeTime\(const Period&\)](#)

10.329.3.14 `bool Arc::UserConfig::CertificateLifeTime (const Period & newCertificateLifeTime)` `[inline]`

Set certificate life time. Sets lifetime of user certificate which will be obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'certificatelifetime'.

Parameters:

newCertificateLifeTime is the life time of a certificate, as a [Period](#) object.

Returns:

This method always returns `true`.

See also:

[CertificateLifeTime\(\) const](#)

10.329.3.15 `const std::string& Arc::UserConfig::CertificatePath () const` `[inline]`

Get path to certificate. The path to the certificate is returned when invoking this method.

Returns:

The certificate path is returned.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CertificatePath\(const std::string&\)](#)
[KeyPath\(\) const](#)

10.329.3.16 `bool Arc::UserConfig::CertificatePath (const std::string & newCertificatePath)` `[inline]`

Set path to certificate. The path to user certificate will be set by this method. The path to the corresponding key can be set with the [KeyPath\(const std::string&\)](#) method. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'certificatepath'.

Parameters:

newCertificatePath is the path to the new certificate.

Returns:

This method always returns `true`.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[CertificatePath\(\) const](#)
[KeyPath\(const std::string&\)](#)

10.329.3.17 void Arc::UserConfig::ClearRejectDiscoveryURLs () [inline]

Clear the rejected service discovery URLs. Clears the list of strings representing URLs which should be ignored during service discovery.

10.329.3.18 bool Arc::UserConfig::CredentialsFound () const [inline]

Validate credential location. Valid credentials consists of a combination of a path to existing CA-certificate directory and either a path to existing proxy or a path to existing user key/certificate pair. If valid credentials are found this method returns `true`, otherwise `false` is returned.

Returns:

`true` if valid credentials are found, otherwise `false`.

See also:

[InitializeCredentials\(\)](#)

10.329.3.19 std::list<ConfigEndpoint> Arc::UserConfig::GetDefaultServices (ConfigEndpoint::Type type = ConfigEndpoint::ANY)

Get the services flagged as default filtered by type. Return all the services which had 'default=yes' in their configuration, if they have the given type.

Parameters:

← *type* is REGISTRY or COMPUTING if only those services are needed, or ANY if all

Returns:

a list of [ConfigEndpoint](#) objects, the default services, empty list if there are no default service, or no services matched the filter

10.329.3.20 ConfigEndpoint Arc::UserConfig::GetService (const std::string & alias)

Get the [ConfigEndpoint](#) for the service with the given alias. Each service in the configuration file has its own section, and the name of the section contains the type of the service ('registry' or 'computing'), and the alias of the service (separated by a slash).

Parameters:

← *alias* is the alias of the service

Returns:

the [ConfigEndpoint](#) generated from the service with the given alias.

10.329.3.21 std::list<ConfigEndpoint> Arc::UserConfig::GetServices (const std::string & groupOrAlias, ConfigEndpoint::Type type = ConfigEndpoint::ANY)

Get one or more service with the given alias or in the given group filtered by type. This is a convenience method for querying the configured services by both the name of a group or an alias of a service. If the name is a name of a group then all the services in the group will be returned (filtered by type). If there is no such group, then a service with the given alias is returned in a single item list (but only if it matches the filter).

Parameters:

← *groupOrAlias* is either a name of a group or an alias of a service

← *type* is REGISTRY or COMPUTING if only those services are needed, or ANY if all

Returns:

a list of [ConfigEndpoint](#) objects, the found services, empty list if no such group and no such alias or no services matched the filter

10.329.3.22 std::list<ConfigEndpoint> Arc::UserConfig::GetServicesInGroup (const std::string & group, ConfigEndpoint::Type type = ConfigEndpoint::ANY)

Get the services in a given group filtered by type. All services of the given group are returned if they match the type filter.

Parameters:

← *group* is the name of the group

← *type* is REGISTRY or COMPUTING if only those services are needed, or ANY if all

Returns:

a list of [ConfigEndpoint](#) objects, the services in the group, empty list if no such group, or no services matched the filter

10.329.3.23 `const User& Arc::UserConfig::GetUser () const [inline]`

Get [User](#) for filesystem access.

Returns:

The user identity to use for file system access

See also:

[SetUser\(const User&\)](#)

10.329.3.24 `const std::string& Arc::UserConfig::IdPName () const [inline]`

Get IdP name. Gets Identity Provider name (Shibboleth) to which user belongs.

Returns:

The IdP name

See also:

[IdPName\(const std::string&\)](#)

10.329.3.25 `bool Arc::UserConfig::IdPName (const std::string & name) [inline]`

Set IdP name. Sets Identity Provider name (Shibboleth) to which user belongs. It is used for contacting Short Lived Certificate [Service](#).

The attribute associated with this setter method is 'idpname'.

Parameters:

name is the new IdP name.

Returns:

This method always returns `true`.

See also:**10.329.3.26** `const std::string& Arc::UserConfig::InfoInterface () const [inline]`

Get the default local information system interface.

Returns:

the [GLUE2](#) InterfaceName string specifying the default local information system interface

See also:

[InfoInterface\(const std::string&\)](#)

10.329.3.27 bool Arc::UserConfig::InfoInterface (const std::string & *infointerface_*) [inline]

Set the default local information system interface. For services which does not specify a local information system interface, this default will be used.

If a local information system interface is given, the computing element will be only queried using this interface.

Parameters:

infointerface_ is a string specifying a [GLUE2](#) InterfaceName

Returns:

This method always returns `true`.

10.329.3.28 bool Arc::UserConfig::InitializeCredentials (initializeCredentialsType *initializeCredentials*)

Initialize user credentials. The location of the user credentials will be tried located when calling this method and stored internally when found. The method searches in different locations. Depending on value of `initializeCredentials` this method behaves differently. Following is an explanation for `RequireCredentials`. For less strict values see information below. First the user proxy or the user key/certificate pair is tried located in the following order:

- Proxy path specified by the environment variable `X509_USER_PROXY`. If value is set and corresponding file does not exist it considered to be an error and no other locations are tried. If found no more proxy paths are tried.
- Current proxy path as passed to the constructor, explicitly set using the setter method [ProxyPath\(const std::string&\)](#) or read from configuration by constructor or `LoadConfiguationFile()` method. If value is set and corresponding file does not exist it considered to be an error and no other locations are tried. If found no more proxy paths are tried.
- Proxy path made of `x509up_u` token concatenated with the user numerical ID located in the OS temporary directory. It is NOT an error if corresponding file does not exist and processing continues.
- Key/certificate paths specified by the environment variables `X509_USER_KEY` and `X509_USER_CERT`. If values are set and corresponding files do not exist it considered to be an error and no other locations are tried. Error message is suppressed if proxy was previously found.
- Current key/certificate paths passed to the constructor or explicitly set using the setter methods [Key-Path\(const std::string&\)](#) and [CertificatePath\(const std::string&\)](#) or read from configuration by constructor or `LoadConfiguationFile()` method. If values are set and corresponding files do not exist it is an error and no other locations are tried. Error message is suppressed if proxy was previously found.
- Key/certificate paths `~/.arc/usercert.pem` and `~/.arc/userkey.pem` respectively are tried. It is not an error if not found.
- Key/certificate paths `~/.globus/usercert.pem` and `~/.globus/userkey.pem` respectively are tried. It is not an error if not found.
- Key/certificate paths created by concatenation of ARC installation location and `/etc/arc/usercert.pem` and `/etc/arc/userkey.pem` respectively are tried. It is not an error if not found.
- Key/certificate located in current working directory are tried.

- If neither proxy nor key/certificate files are found this is considered to be an error.

Along with the proxy and key/certificate pair, the path of the directory containing CA certificates is also located. The presence of directory will be checked in the following order and first found is accepted:

- Path specified by the X509_CERT_DIR environment variable. It is an error if value is set and directory does not exist.
- Current path explicitly specified by using the setter method [CACertificatesDirectory\(\)](#) or read from configuration by constructor or LoadConfiguartionFile() method. It is an error if value is set and directory does not exist.
- Path ~/.globus/certificates. It is not an error if it does not exist.
- Path created by concatenating the ARC installation location and /etc/certificates. It is not an error if it does not exist.
- Path created by concatenating the ARC installation location and /share/certificates. It is not an error if it does not exist.
- Path /etc/grid-security/certificates.

It is an error if none of the directories above exist.

In case of initializeCredentials == TryCredentials method behaves same way like in case RequireCredentials except it does not report errors through its [Logger](#) object and does not return false.

If NotTryCredentials is used method does not check for presence of credentials. It behaves like if corresponding files are always present.

And in case of SkipCredentials method does nothing.

All options with SkipCA* prefix behaves similar to those without prefix except the path of the directory containing CA certificates is completely ignored.

See also:

[CredentialsFound\(\)](#)
[ProxyPath\(const std::string&\)](#)
[KeyPath\(const std::string&\)](#)
[CertificatePath\(const std::string&\)](#)
[CACertificatesDirectory\(const std::string&\)](#)

10.329.3.29 `const std::string& Arc::UserConfig::JobDownloadDirectory () const [inline]`

Get download directory. returns directory which will be used to download the job directory using arcget command.

The attribute associated with the method is 'jobdownloadaddirectory'.

Returns:

This method returns the job download directory.

See also:

10.329.3.30 `bool Arc::UserConfig::JobDownloadDirectory (const std::string & newDownloadDirectory) [inline]`

Set download directory. Sets directory which will be used to download the job directory using `arcget` command.

The attribute associated with this setter method is 'jobdownloaddirectory'.

Parameters:

newDownloadDirectory is the path to the download directory.

Returns:

This method always returns `true`.

See also:

10.329.3.31 `const std::string& Arc::UserConfig::JobListFile () const [inline]`

Get a reference to the path of the job list file. The job list file is used to store and fetch information about submitted computing jobs to computing services. This method will return the path to the specified job list file.

Returns:

The path to the job list file is returned.

See also:

[JobListFile\(const std::string&\)](#)

10.329.3.32 `bool Arc::UserConfig::JobListFile (const std::string & path)`

Set path to job list file. The method takes a path to a file which will be used as the job list file for storing and reading job information. If the specified path *path* does not exist a empty job list file will be tried created. If creating the job list file in any way fails *false* will be returned and a [ERROR](#) message will be reported. Otherwise *true* is returned. If the directory containing the file does not exist, it will be tried created. The method will also return *false* if the file is not a regular file.

The attribute associated with this setter method is 'joblist'.

Parameters:

path the path to the job list file.

Returns:

If the job list file is a regular file or if it can be created *true* is returned, otherwise *false* is returned.

See also:

[JobListFile\(\) const](#)

10.329.3.33 `const std::string& Arc::UserConfig::KeyPassword () const` `[inline]`

Get password for generated key. Get password to be used to encode private key of credentials obtained from Short Lived Credentials [Service](#).

Returns:

The key password is returned.

See also:

[KeyPassword\(const std::string&\)](#)
[KeyPath\(\) const](#)
[KeySize\(\) const](#)

10.329.3.34 `bool Arc::UserConfig::KeyPassword (const std::string & newKeyPassword)`
`[inline]`

Set password for generated key. Set password to be used to encode private key of credentials obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'keypassword'.

Parameters:

newKeyPassword is the new password to the key.

Returns:

This method always returns `true`.

See also:

[KeyPassword\(\) const](#)
[KeyPath\(const std::string&\)](#)
[KeySize\(int\)](#)

10.329.3.35 `const std::string& Arc::UserConfig::KeyPath () const` `[inline]`

Get path to key. The path to the key is returned when invoking this method.

Returns:

The path to the user key is returned.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[KeyPath\(const std::string&\)](#)
[CertificatePath\(\) const](#)
[KeyPassword\(\) const](#)
[KeySize\(\) const](#)

10.329.3.36 bool Arc::UserConfig::KeyPath (const std::string & *newKeyPath*) [inline]

Set path to key. The path to user key will be set by this method. The path to the corresponding certificate can be set with the [CertificatePath\(const std::string&\)](#) method. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'keypath'.

Parameters:

newKeyPath is the path to the new key.

Returns:

This method always returns `true`.

See also:

[InitializeCredentials\(\)](#)
[CredentialsFound\(\) const](#)
[KeyPath\(\) const](#)
[CertificatePath\(const std::string&\)](#)
[KeyPassword\(const std::string&\)](#)
[KeySize\(int\)](#)

10.329.3.37 int Arc::UserConfig::KeySize () const [inline]

Get key size. Get size/strengt of private key of credentials obtained from Short Lived Credentials [Service](#).

Returns:

The key size, as an integer, is returned.

See also:

[KeySize\(int\)](#)
[KeyPath\(\) const](#)
[KeyPassword\(\) const](#)

10.329.3.38 bool Arc::UserConfig::KeySize (int *newKeySize*) [inline]

Set key size. Set size/strengt of private key of credentials obtained from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'keysize'.

Parameters:

newKeySize is the size, an an integer, of the key.

Returns:

This method always returns `true`.

See also:

[KeySize\(\) const](#)
[KeyPath\(const std::string&\)](#)
[KeyPassword\(const std::string&\)](#)

10.329.3.39 `bool Arc::UserConfig::LoadConfigurationFile (const std::string & conffile, bool ignoreJobListFile = true)`

Load specified configuration file. The configuration file passed is parsed by this method by using the [IniConfig](#) class. If the parsing is unsuccessful a [WARNING](#) is reported.

The format of the configuration file should follow that of INI, and every attribute present in the file is only allowed once (except the 'rejectmanagement' and 'rejectdiscovery' attributes), otherwise a [WARNING](#) will be reported. For the list of allowed attributes see the detailed description of [UserConfig](#).

Parameters:

*conf*file is the path to the configuration file.

ignoreJobListFile is a optional boolean which indicates whether the joblistfile attribute in the configuration file should be ignored. Default is to ignored it (`true`).

Returns:

If loading the configuration file succeeds `true` is returned, otherwise `false` is returned.

See also:

[SaveToFile\(\)](#)

10.329.3.40 `Arc::UserConfig::operator bool (void) const [inline]`

Check for validity. The validity of an object created from this class can be checked using this casting operator. An object is valid if the constructor did not encounter any errors.

See also:

[operator!\(\)](#)

10.329.3.41 `bool Arc::UserConfig::operator! (void) const [inline]`

Check for non-validity. See [operator bool\(\)](#) for a description.

See also:

[operator bool\(\)](#)

10.329.3.42 `const std::string& Arc::UserConfig::OverlayFile () const [inline]`

Get path to configuration overlay file.

Returns:

The overlay file path

See also:

[OverlayFile\(const std::string&\)](#)

10.329.3.43 bool Arc::UserConfig::OverlayFile (const std::string & *path*) [inline]

Set path to configuration overlay file. Content of specified file is a backdoor to configuration XML generated from information stored in this class. The content of file is passed to [BaseConfig](#) class in ApplyToConfig(BaseConfig&) then merged with internal configuration XML representation. This feature is meant for quick prototyping/testing/tuning of functionality without rewriting code. It is meant for developers and most users won't need it.

The attribute associated with this setter method is 'overlayfile'.

Parameters:

path is the new overlay file path.

Returns:

This method always returns `true`.

See also:**10.329.3.44 const std::string& Arc::UserConfig::Password () const [inline]**

Get password. Get password which is used for requesting credentials from Short Lived Credentials [Service](#).

Returns:

The password is returned.

See also:

[Password\(const std::string&\)](#)

10.329.3.45 bool Arc::UserConfig::Password (const std::string & *newPassword*) [inline]

Set password. Set password which is used for requesting credentials from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'password'.

Parameters:

newPassword is the new password to set.

Returns:

This method always returns `true`.

See also:

[Password\(\) const](#)

10.329.3.46 `const std::string& Arc::UserConfig::ProxyPath () const` `[inline]`

Get path to user proxy. Retrieve path to user proxy.

Returns:

Returns the path to the user proxy.

See also:

[ProxyPath\(const std::string&\)](#)

10.329.3.47 `bool Arc::UserConfig::ProxyPath (const std::string & newProxyPath)` `[inline]`

Set path to user proxy. This method will set the path of the user proxy. Note that the [InitializeCredentials\(\)](#) method will also try to set this path, by searching in different locations.

The attribute associated with this setter method is 'proxypath'

Parameters:

newProxyPath is the path to a user proxy.

Returns:

This method always returns `true`.

See also:

[InitializeCredentials\(\)](#)

[CredentialsFound\(\)](#)

[ProxyPath\(\) const](#)

10.329.3.48 `const std::list<std::string>& Arc::UserConfig::RejectDiscoveryURLs () const` `[inline]`

Get the list of rejected service discovery URLs. This list is populated by the (possibly multiple) 'rejectdiscovery' configuration options. A service registry should not be queried if its [URL](#) matches any string in this list.

Returns:

a list of rejected service discovery URLs

10.329.3.49 `const std::list<std::string>& Arc::UserConfig::RejectManagementURLs () const` `[inline]`

Get the list of rejected job management URLs. This list is populated by the (possibly multiple) 'rejectmanagement' configuration options. Those jobs should not be managed, that reside on a computing element with a matching [URL](#).

Returns:

a list of rejected job management URLs

10.329.3.50 bool Arc::UserConfig::SaveToFile (const std::string &filename) const

Save to INI file. This method will save the object data as a INI file. The saved file can be loaded with the LoadConfigurationFile method.

Parameters:

filename the name of the file which the data will be saved to.

Returns:

false if unable to get handle on file, otherwise true is returned.

See also:

[LoadConfigurationFile\(\)](#)

10.329.3.51 void Arc::UserConfig::SetUser (const User &u) [inline]

Set [User](#) for filesystem access. Sometimes it is desirable to use the identity of another user when accessing the filesystem. This user can be specified through this method. By default this user is the same as the user running the process.

Parameters:

u [User](#) identity to use

10.329.3.52 const URL& Arc::UserConfig::SLCS () const [inline]

Get the [URL](#) to the Short Lived Certificate [Service](#) (SLCS).

Returns:

The SLCS is returned.

See also:

[SLCS\(const URL&\)](#)

10.329.3.53 bool Arc::UserConfig::SLCS (const URL &newSLCS) [inline]

Set the [URL](#) to the Short Lived Certificate [Service](#) (SLCS). The attribute associated with this setter method is 'slcs'.

Parameters:

newSLCS is the [URL](#) to the SLCS

Returns:

This method always returns true.

See also:

[SLCS\(\) const](#)

10.329.3.54 `const std::string& Arc::UserConfig::StoreDirectory () const` `[inline]`

Get store directory. Sets directory which is used to store credentials obtained from Short Lived [Credential](#) Service.

Returns:

The path to the store directory is returned.

See also:

[StoreDirectory\(const std::string&\)](#)

10.329.3.55 `bool Arc::UserConfig::StoreDirectory (const std::string & newStoreDirectory)` `[inline]`

Set store directory. Sets directory which will be used to store credentials obtained from Short Lived [Credential](#) Service.

The attribute associated with this setter method is 'storedirectory'.

Parameters:

newStoreDirectory is the path to the store directory.

Returns:

This method always returns `true`.

See also:**10.329.3.56** `const std::string& Arc::UserConfig::SubmissionInterface () const` `[inline]`

Get the default submission interface.

Returns:

the [GLUE2](#) InterfaceName string specifying the default submission interface

See also:

[SubmissionInterface\(const std::string&\)](#)

10.329.3.57 `bool Arc::UserConfig::SubmissionInterface (const std::string & submissioninterface_)` `[inline]`

Set the default submission interface. For services which does not specify a submission interface this default submission interface will be used.

If a submission interface is given, then all the jobs will be submitted to this interface, no other job submission interfaces of the computing element will be tried.

Parameters:

submissioninterface_ is a string specifying a [GLUE2](#) InterfaceName

Returns:

This method always returns `true`.

10.329.3.58 int Arc::UserConfig::Timeout () const [inline]

Get timeout. Returns the timeout in seconds.

Returns:

timeout in seconds.

See also:

[Timeout\(int\)](#)
[DEFAULT_TIMEOUT](#)

10.329.3.59 bool Arc::UserConfig::Timeout (int newTimeout)

Set timeout. When communicating with a service the timeout specifies how long, in seconds, the communicating instance should wait for a response. If the response have not been recieved before this period in time, the connection is typically dropped, and an error will be reported.

This method will set the timeout to the specified integer. If the passed integer is less than or equal to 0 then `false` is returned and the timeout will not be set, otherwise `true` is returned and the timeout will be set to the new value.

The attribute associated with this setter method is 'timeout'.

Parameters:

newTimeout the new timeout value in seconds.

Returns:

`false` in case *newTimeout* <= 0, otherwise `true`.

See also:

[Timeout\(\) const](#)
[DEFAULT_TIMEOUT](#)

10.329.3.60 const std::string& Arc::UserConfig::UserName () const [inline]

Get user-name. Get username which is used for requesting credentials from Short Lived Credentials [Service](#).

Returns:

The username is returned.

See also:

[UserName\(const std::string&\)](#)

10.329.3.61 `bool Arc::UserConfig::UserName (const std::string & name) [inline]`

Set user-name for SLCS. Set username which is used for requesting credentials from Short Lived Credentials [Service](#).

The attribute associated with this setter method is 'username'.

Parameters:

name is the name of the user.

Returns:

This method always return true.

See also:

[UserName\(\) const](#)

10.329.3.62 `const std::string& Arc::UserConfig::UtilsDirPath () const [inline]`

Get path to directory storing utility files for DataPoints.

Returns:

The utils dir path

See also:

[UtilsDirPath\(const std::string&\)](#)

10.329.3.63 `bool Arc::UserConfig::UtilsDirPath (const std::string & dir)`

Set path to directory storing utility files for DataPoints. Some DataPoints can store information on remote services in local files. This method sets the path to the directory containing these files. For example arc* tools set it to ARCUSERDIRECTORY and A-REX sets it to the control directory. The directory is created if it does not exist.

Parameters:

dir is the new utils dir path.

Returns:

This method always returns `true`.

10.329.3.64 `const std::string& Arc::UserConfig::Verbosity () const [inline]`

Get the user selected level of verbosity. The string representation of the verbosity level specified by the user is returned when calling this method. If the user have not specified the verbosity level the empty string will be referenced.

Returns:

the verbosity level, or empty if it has not been set.

See also:

[Verbosity\(const std::string&\)](#)

10.329.3.65 bool Arc::UserConfig::Verbosity (const std::string & *newVerbosity*)

Set verbosity. The verbosity will be set when invoking this method. If the string passed cannot be parsed into a corresponding LogLevel, using the function a [WARNING](#) is reported and `false` is returned, otherwise `true` is returned.

The attribute associated with this setter method is 'verbosity'.

Returns:

`true` in case the verbosity could be set to a allowed LogLevel, otherwise `false`.

See also:

[Verbosity\(\) const](#)

10.329.3.66 const std::string& Arc::UserConfig::VOMSESPath ()

Get path to file containing VOMS configuration. Get path to file which contains list of VOMS services and associated configuration parameters.

Returns:

The path to VOMS configuration file is returned.

See also:

[VOMSESPath\(const std::string&\)](#)

10.329.3.67 bool Arc::UserConfig::VOMSESPath (const std::string & *path*) [inline]

Set path to file containing VOMS configuration. Set path to file which contains list of VOMS services and associated configuration parameters needed to contact those services. It is used by arcproxy.

The attribute associated with this setter method is 'vomsserverpath'.

Parameters:

path the path to VOMS configuration file

Returns:

This method always return true.

See also:

[VOMSESPath\(\) const](#)

10.329.4 Field Documentation

10.329.4.1 `const std::string Arc::UserConfig::ARCUSERDIRECTORY` `[static]`

Path to ARC user home directory. The *ARCUSERDIRECTORY* variable is the path to the ARC home directory of the current user. This path is created using the [User::Home\(\)](#) method.

See also:

[User::Home\(\)](#)

10.329.4.2 `const std::string Arc::UserConfig::DEFAULT_BROKER` `[static]`

Default broker. The *DEFAULT_BROKER* specifies the name of the broker which should be used in case no broker is explicitly chosen.

See also:

[Broker](#)
[Broker\(const std::string&\)](#)
[Broker\(const std::string&, const std::string&\)](#)
[Broker\(\) const](#)

10.329.4.3 `const int Arc::UserConfig::DEFAULT_TIMEOUT = 20` `[static]`

Default timeout in seconds. The *DEFAULT_TIMEOUT* specifies interval which will be used in case no timeout interval have been explicitly specified. For a description about timeout see [Timeout\(int\)](#).

See also:

[Timeout\(int\)](#)
[Timeout\(\) const](#)

10.329.4.4 `const std::string Arc::UserConfig::DEFAULTCONFIG` `[static]`

Path to default configuration file. The *DEFAULTCONFIG* variable is the path to the default configuration file used in case no configuration file have been specified. The path is created from the *ARCUSERDIRECTORY* object.

10.329.4.5 `const std::string Arc::UserConfig::EXAMPLECONFIG` `[static]`

Path to example configuration. The *EXAMPLECONFIG* variable is the path to the example configuration file.

10.329.4.6 `const std::string Arc::UserConfig::SYSCONFIG` `[static]`

Path to system configuration. The *SYSCONFIG* variable is the path to the system configuration file. This variable is only equal to *SYSCONFIGARCLOC* if ARC is installed in the root (highly unlikely).

10.329.4.7 `const std::string Arc::UserConfig::SYSCONFIGARCLOC` `[static]`

Path to system configuration at ARC location. The *SYSCONFIGARCLOC* variable is the path to the system configuration file which reside at the ARC installation location.

The documentation for this class was generated from the following file:

- UserConfig.h

10.330 Arc::UsernameToken Class Reference

Interface for manipulation of WS-Security according to Username Token [Profile](#).

```
#include <UsernameToken.h>
```

Public Types

- enum [PasswordType](#)

Public Member Functions

- [UsernameToken](#) (SOAPEnvelope &soap)
- [UsernameToken](#) (SOAPEnvelope &soap, const std::string &username, const std::string &password, const std::string &uid, [PasswordType](#) pwdtype)
- [UsernameToken](#) (SOAPEnvelope &soap, const std::string &username, const std::string &id, bool mac, int iteration)
- [operator bool](#) (void)
- std::string [Username](#) (void)
- bool [Authenticate](#) (const std::string &password, std::string &derived_key)
- bool [Authenticate](#) (std::istream &password, std::string &derived_key)

10.330.1 Detailed Description

Interface for manipulation of WS-Security according to Username Token [Profile](#).

10.330.2 Member Enumeration Documentation

10.330.2.1 enum Arc::UsernameToken::PasswordType

SOAP header element

10.330.3 Constructor & Destructor Documentation

10.330.3.1 Arc::UsernameToken::UsernameToken (SOAPEnvelope & *soap*)

Link to existing SOAP header and parse Username Token information. Username Token related information is extracted from SOAP header and stored in class variables.

10.330.3.2 Arc::UsernameToken::UsernameToken (SOAPEnvelope & *soap*, const std::string & *username*, const std::string & *password*, const std::string & *uid*, PasswordType *pwdtype*)

Add Username Token information into the SOAP header. Generated token contains elements Username and Password and is meant to be used for authentication.

Parameters:

soap the SOAP message

username <wsse:Username>...</wsse:Username> - if empty it is entered interactively from stdin

password <wsse:Password Type="...">...</wsse:Password> - if empty it is entered interactively from stdin

uid <wsse:[UsernameToken](#) wsu:ID="...">

pwdtype <wsse:Password Type="...">...</wsse:Password>

10.330.3.3 Arc::UsernameToken::UsernameToken (SOAPEnvelope & soap, const std::string & username, const std::string & id, bool mac, int iteration)

Add Username Token information into the SOAP header. Generated token contains elements Username and Salt and is meant to be used for deriving Key Derivation.

Parameters:

soap the SOAP message

username <wsse:Username>...</wsse:Username>

mac if derived key is meant to be used for [Message](#) Authentication Code

iteration <wsse11:Iteration>...</wsse11:Iteration>

10.330.4 Member Function Documentation

10.330.4.1 bool Arc::UsernameToken::Authenticate (std::istream & password, std::string & derived_key)

Checks parsed token against password stored in specified stream. If token is meant to be used for deriving a key then key is returned in derived_key

10.330.4.2 bool Arc::UsernameToken::Authenticate (const std::string & password, std::string & derived_key)

Checks parsed/generated token against specified password. If token is meant to be used for deriving a key then key is returned in derived_key. In that case authentication is performed outside of [UsernameToken](#) class using obtained derived_key.

10.330.4.3 Arc::UsernameToken::operator bool (void)

Returns true of constructor succeeded

10.330.4.4 std::string Arc::UsernameToken::Username (void)

Returns username associated with this instance

The documentation for this class was generated from the following file:

- UsernameToken.h

10.331 Arc::UserSwitch Class Reference

Class for temporary switching of user id.

```
#include <arc/User.h>
```

Public Member Functions

- [UserSwitch](#) (int uid, int gid)
- [~UserSwitch](#) (void)
- [operator bool](#) (void)

10.331.1 Detailed Description

Class for temporary switching of user id. If this class is created, the user identity is switched to the provided uid and gid. Due to an internal lock there will be only one valid instance of this class. Any attempt to create another instance will block until the first one is destroyed. If uid and gid are set to 0 then the user identity is not switched, but the lock is applied anyway. The lock has a dual purpose. The first and most important is to protect communication with the underlying operating system which may depend on user identity. For that it is advisable for code which talks to the operating system to acquire a valid instance of this class. Care must be taken not to hold that instance too long as that may block other code in a multithreaded environment. The other purpose of this lock is to provide a workaround for a glibc bug in `__nptl_setxid`. This bug causes lockup of `seteuid()` function if racing with `fork`. To avoid this problem the lock mentioned above is used by the [Run](#) class while spawning a new process.

The documentation for this class was generated from the following file:

- User.h

10.332 Arc::VOMSACInfo Class Reference

The documentation for this class was generated from the following file:

- VOMSUtil.h

10.333 Arc::VOMSTrustList Class Reference

```
#include <VOMSUtil.h>
```

Public Member Functions

- [VOMSTrustList](#) (const std::vector< std::string > &encoded_list)
- [VOMSTrustList](#) (const std::vector< VOMSTrustChain > &chains, const std::vector< VOMSTrustRegex > ®exs)
- VOMSTrustChain & [AddChain](#) (const VOMSTrustChain &chain)
- VOMSTrustChain & [AddChain](#) (void)
- [RegularExpression](#) & [AddRegex](#) (const VOMSTrustRegex ®)

10.333.1 Detailed Description

Stores definitions for making decision if VOMS server is trusted

10.333.2 Constructor & Destructor Documentation

10.333.2.1 Arc::VOMSTrustList::VOMSTrustList (const std::vector< std::string > &encoded_list)

Creates chain lists and regexps from plain list. List is made of chunks delimited by elements containing pattern "NEXT CHAIN". Each chunk with more than one element is converted into one instance of VOMSTrustChain. Chunks with single element are converted to VOMSTrustChain if element does not have special symbols. Otherwise it is treated as regular expression. Those symbols are '^','\$' and '*'. Trusted chains can be configured in two ways: one way is: <tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=host/arthur.hep.lu.se</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>----NEXT CHAIN--- </tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/OU=computers/CN=voms.cern.ch</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/CN=CERN Trusted Certification Authority</tls:VOMSCertTrustDN> </tls:VOMSCertTrustDNChain> the other way is: <tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=host/arthur.hep.lu.se</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/O=Grid/O=NorduGrid/CN=NorduGrid Certification Authority</tls:VOMSCertTrustDN> </tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDNChain> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/OU=computers/CN=voms.cern.ch</tls:VOMSCertTrustDN> <tls:VOMSCertTrustDN>/DC=ch/DC=cern/CN=CERN Trusted Certification Authority</tls:VOMSCertTrustDN> </tls:VOMSCertTrustDNChain> each chunk is supposed to contain a suit of DN of trusted certificate chain, in which the first DN is the DN of the certificate (cert0) which is used to sign the Attribute Certificate (AC), the second DN is the DN of the issuer certificate(cert1) which is used to sign cert0. So if there are one or more intermediate issuers, then there should be 3 or more than 3 DNs in this chunk (considering cert0 and the root certificate, plus the intermediate certificate) .

10.333.2.2 Arc::VOMSTrustList::VOMSTrustList (const std::vector< VOMSTrustChain > &chains, const std::vector< VOMSTrustRegex > ®exs)

Creates chain lists and regexps from those specified in arguments. See [AddChain\(\)](#) and [AddRegex\(\)](#) for more information.

10.333.3 Member Function Documentation

10.333.3.1 VOMSTrustChain& Arc::VOMSTrustList::AddChain (void)

Adds empty chain of trusted DNs to list.

10.333.3.2 VOMSTrustChain& Arc::VOMSTrustList::AddChain (const VOMSTrustChain & *chain*)

Adds chain of trusted DNs to list. During verification each signature of AC is checked against all stored chains. DNs of chain of certificate used for signing AC are compared against DNs stored in these chains one by one. If needed DN of issuer of last certificate is checked too. Comparison succeeds if DNs in at least one stored chain are same as those in certificate chain. Comparison stops when all DNs in stored chain are compared. If there are more DNs in stored chain than in certificate chain then comparison fails. Empty stored list matches any certificate chain. Taking into account that certificate chains are verified down to trusted CA anyway, having more than one DN in stored chain seems to be useless. But such feature may be found useful by some very strict sysadmins. ??? IMO, DN list here is not only for authentication, it is also kind of ACL, which means the AC consumer only trusts those DNs which issues AC.

10.333.3.3 RegularExpression& Arc::VOMSTrustList::AddRegex (const VOMSTrustRegex & *reg*)

Adds regular expression to list. During verification each signature of AC is checked against all stored regular expressions. DN of signing certificate must match at least one of stored regular expressions.

The documentation for this class was generated from the following file:

- VOMSUtil.h

10.334 Arc::WatchdogChannel Class Reference

This class is meant to be used in code which provides "I'm alive" ticks to watchdog.

```
#include <arc/Watchdog.h>
```

Public Member Functions

- [WatchdogChannel](#) (int timeout)
- [~WatchdogChannel](#) (void)
- void [Kick](#) (void)

10.334.1 Detailed Description

This class is meant to be used in code which provides "I'm alive" ticks to watchdog.

10.334.2 Constructor & Destructor Documentation

10.334.2.1 Arc::WatchdogChannel::WatchdogChannel (int *timeout*)

Defines watchdog kicking source with specified timeout. Code must call [Kick\(\)](#) method of this instance to keep watchdog from timeouting. If object is destroyed watchdog does not monitor it anymore. Although timeout is specified in seconds real time resolution of watchdog is about 1 minute.

The documentation for this class was generated from the following file:

- Watchdog.h

10.335 Arc::WatchdogListener Class Reference

This class is meant to provide interface for Watchdog executor part.

```
#include <arc/Watchdog.h>
```

Public Member Functions

- bool [Listen](#) (void)
- bool [Listen](#) (int limit, bool &error)

10.335.1 Detailed Description

This class is meant to provide interface for Watchdog executor part.

10.335.2 Member Function Documentation

10.335.2.1 bool Arc::WatchdogListener::Listen (int *limit*, bool & *error*)

Similar to [Listen\(\)](#) but forces method to exit after limit seconds. If limit passed false is returned. If method is exited due to internal error then error argument is filled with true.

10.335.2.2 bool Arc::WatchdogListener::Listen (void)

Waits till timeout occurs and then returns true. If any error occurs it returns false and watchdog is normally not usable anymore.

The documentation for this class was generated from the following file:

- Watchdog.h

10.336 Arc::WSAEndpointReference Class Reference

Interface for manipulation of WS-Addressing [Endpoint](#) Reference.

```
#include <WSA.h>
```

Public Member Functions

- [WSAEndpointReference](#) ([XMLNode](#) epr)
- [WSAEndpointReference](#) (const [WSAEndpointReference](#) &wsa)
- [WSAEndpointReference](#) (const std::string &address)
- [WSAEndpointReference](#) (void)
- [~WSAEndpointReference](#) (void)
- std::string [Address](#) (void) const
- bool [hasAddress](#) (void) const
- void [Address](#) (const std::string &uri)
- [WSAEndpointReference](#) & [operator=](#) (const std::string &address)
- [XMLNode](#) [ReferenceParameters](#) (void)
- [XMLNode](#) [MetaData](#) (void)
- [operator](#) [XMLNode](#) (void)

10.336.1 Detailed Description

Interface for manipulation of WS-Addressing [Endpoint](#) Reference. It works on [Endpoint](#) Reference stored in XML tree. No information is stored in this object except reference to corresponding XML subtree.

10.336.2 Constructor & Destructor Documentation

10.336.2.1 Arc::WSAEndpointReference::WSAEndpointReference ([XMLNode](#) *epr*)

Link to top level EPR XML node Linking to existing EPR in XML tree

10.336.2.2 Arc::WSAEndpointReference::WSAEndpointReference (const [WSAEndpointReference](#) & *wsa*)

Copy constructor

10.336.2.3 Arc::WSAEndpointReference::WSAEndpointReference (const std::string & *address*)

Creating independent EPR - not implemented

10.336.2.4 Arc::WSAEndpointReference::WSAEndpointReference (void)

Dummy constructor - creates invalid instance

10.336.2.5 Arc::WSAEndpointReference::~~WSAEndpointReference (void)

Destructor. All empty elements of EPR XML are destroyed here too

10.336.3 Member Function Documentation

10.336.3.1 void Arc::WSAEndpointReference::Address (const std::string & *uri*)

Assigns new Address value. If EPR had no Address element it is created.

10.336.3.2 std::string Arc::WSAEndpointReference::Address (void) const

Returns Address ([URL](#)) encoded in EPR

10.336.3.3 bool Arc::WSAEndpointReference::hasAddress (void) const

Returns true if Address is defined

10.336.3.4 XMLNode Arc::WSAEndpointReference::MetaData (void)

Access to MetaData element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no MetaData element it is created.

10.336.3.5 Arc::WSAEndpointReference::operator XMLNode (void)

Returns reference to EPR top XML node

10.336.3.6 WSAEndpointReference& Arc::WSAEndpointReference::operator= (const std::string & *address*)

Same as Address(uri)

10.336.3.7 XMLNode Arc::WSAEndpointReference::ReferenceParameters (void)

Access to ReferenceParameters element of EPR. Obtained XML element should be manipulated directly in application-dependent way. If EPR had no ReferenceParameters element it is created.

The documentation for this class was generated from the following file:

- WSA.h

10.337 Arc::WSAHeader Class Reference

Interface for manipulation WS-Addressing information in SOAP header.

```
#include <WSA.h>
```

Public Member Functions

- [WSAHeader](#) (SOAPEnvelope &soap)
- [WSAHeader](#) (const std::string &action)
- std::string [To](#) (void) const
- bool [hasTo](#) (void) const
- void [To](#) (const std::string &uri)
- [WSAEndpointReference From](#) (void)
- [WSAEndpointReference ReplyTo](#) (void)
- [WSAEndpointReference FaultTo](#) (void)
- std::string [Action](#) (void) const
- bool [hasAction](#) (void) const
- void [Action](#) (const std::string &uri)
- std::string [MessageID](#) (void) const
- bool [hasMessageID](#) (void) const
- void [MessageID](#) (const std::string &uri)
- std::string [RelatesTo](#) (void) const
- bool [hasRelatesTo](#) (void) const
- void [RelatesTo](#) (const std::string &uri)
- std::string [RelationshipType](#) (void) const
- bool [hasRelationshipType](#) (void) const
- void [RelationshipType](#) (const std::string &uri)
- [XMLNode ReferenceParameter](#) (int n)
- [XMLNode ReferenceParameter](#) (const std::string &name)
- [XMLNode NewReferenceParameter](#) (const std::string &name)
- [operator XMLNode](#) (void)

Static Public Member Functions

- static bool [Check](#) (SOAPEnvelope &soap)

Protected Attributes

- bool [header_allocated_](#)

10.337.1 Detailed Description

Interface for manipulation WS-Addressing information in SOAP header. It works on [Endpoint](#) Reference stored in XML tree. No information is stored in this object except reference to corresponding XML subtree.

10.337.2 Constructor & Destructor Documentation

10.337.2.1 Arc::WSAHeader::WSAHeader (SOAPEnvelope & *soap*)

Linking to a header of existing SOAP message

10.337.2.2 Arc::WSAHeader::WSAHeader (const std::string & *action*)

Creating independent SOAP header - not implemented

10.337.3 Member Function Documentation

10.337.3.1 void Arc::WSAHeader::Action (const std::string & *uri*)

Set content of Action element of SOAP Header. If such element does not exist it's created.

10.337.3.2 std::string Arc::WSAHeader::Action (void) const

Returns content of Action element of SOAP Header.

10.337.3.3 static bool Arc::WSAHeader::Check (SOAPEnvelope & *soap*) [static]

Tells if specified SOAP message has WSA header

10.337.3.4 WSAEndpointReference Arc::WSAHeader::FaultTo (void)

Returns FaultTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulated.

10.337.3.5 WSAEndpointReference Arc::WSAHeader::From (void)

Returns From element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulated.

10.337.3.6 bool Arc::WSAHeader::hasAction (void) const

Returns true if Action element is defined.

10.337.3.7 bool Arc::WSAHeader::hasMessageID (void) const

Returns true if MessageID element is defined.

10.337.3.8 bool Arc::WSAHeader::hasRelatesTo (void) const

Returns true if RelatesTo element is defined.

10.337.3.9 bool Arc::WSAHeader::hasRelationshipType (void) const

Returns true if RelationshipType element is defined.

10.337.3.10 bool Arc::WSAHeader::hasTo (void) const

Returns true if To element is defined.

10.337.3.11 void Arc::WSAHeader::MessageID (const std::string & uri)

Set content of MessageID element of SOAP Header. If such element does not exist it's created.

10.337.3.12 std::string Arc::WSAHeader::MessageID (void) const

Returns content of MessageID element of SOAP Header.

10.337.3.13 XMLNode Arc::WSAHeader::NewReferenceParameter (const std::string & name)

Creates new ReferenceParameter element with specified name. Returns reference to created element.

10.337.3.14 Arc::WSAHeader::operator XMLNode (void)

Returns reference to SOAP Header - not implemented

10.337.3.15 XMLNode Arc::WSAHeader::ReferenceParameter (const std::string & name)

Returns first ReferenceParameter element with specified name

10.337.3.16 XMLNode Arc::WSAHeader::ReferenceParameter (int n)

Return n-th ReferenceParameter element

10.337.3.17 void Arc::WSAHeader::RelatesTo (const std::string & uri)

Set content of RelatesTo element of SOAP Header. If such element does not exist it's created.

10.337.3.18 std::string Arc::WSAHeader::RelatesTo (void) const

Returns content of RelatesTo element of SOAP Header.

10.337.3.19 void Arc::WSAHeader::RelationshipType (const std::string & uri)

Set content of RelationshipType element of SOAP Header. If such element does not exist it's created.

10.337.3.20 std::string Arc::WSAHeader::RelationshipType (void) const

Returns content of RelationshipType element of SOAP Header.

10.337.3.21 WSAEndpointReference Arc::WSAHeader::ReplyTo (void)

Returns ReplyTo element of SOAP Header. If such element does not exist it's created. Obtained element may be manipulated.

10.337.3.22 void Arc::WSAHeader::To (const std::string & uri)

Set content of To element of SOAP Header. If such element does not exist it's created.

10.337.3.23 std::string Arc::WSAHeader::To (void) const

Returns content of To element of SOAP Header.

10.337.4 Field Documentation**10.337.4.1 bool Arc::WSAHeader::header_allocated_ [protected]**

SOAP header element

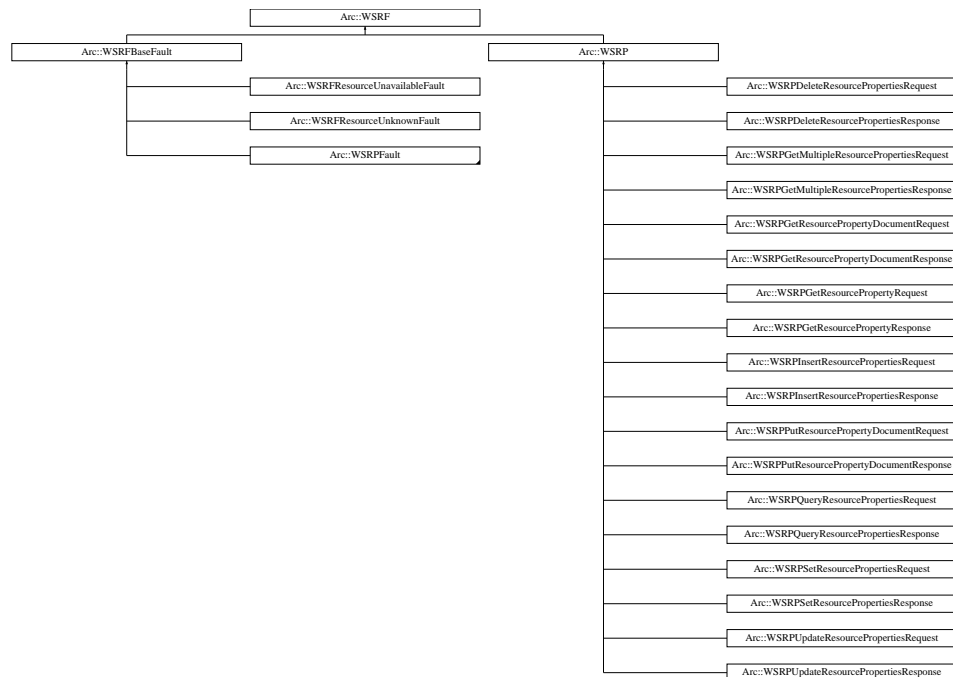
The documentation for this class was generated from the following file:

- WSA.h

10.338 Arc::WSRF Class Reference

Base class for every [WSRF](#) message.

#include <WSRF.h> Inheritance diagram for Arc::WSRF::



Public Member Functions

- [WSRF](#) (SOAPEnvelope &soap, const std::string &action="")
- [WSRF](#) (bool fault=false, const std::string &action="")
- virtual SOAPEnvelope & [SOAP](#) (void)
- virtual [operator bool](#) (void)

Protected Member Functions

- void [set_namespaces](#) (void)

Protected Attributes

- bool [allocated_](#)
- bool [valid_](#)

10.338.1 Detailed Description

Base class for every [WSRF](#) message. This class is not intended to be used directly. Use it like reference while passing through unknown [WSRF](#) message or use classes derived from it.

10.338.2 Constructor & Destructor Documentation

10.338.2.1 Arc::WSRF::WSRF (SOAPEnvelope & *soap*, const std::string & *action* = "")

Constructor - creates object out of supplied SOAP tree.

10.338.2.2 Arc::WSRF::WSRF (bool *fault* = false, const std::string & *action* = "")

Constructor - creates new [WSRF](#) object

10.338.3 Member Function Documentation

10.338.3.1 virtual Arc::WSRF::operator bool (void) [inline, virtual]

Returns true if instance is valid

References `valid_`.

10.338.3.2 void Arc::WSRF::set_namespaces (void) [protected]

true if object represents valid [WSRF](#) message set WS Resource namespaces and default prefixes in SOAP message

Reimplemented in [Arc::WSRP](#), and [Arc::WSRFBBaseFault](#).

10.338.3.3 virtual SOAPEnvelope& Arc::WSRF::SOAP (void) [inline, virtual]

Direct access to underlying SOAP element

10.338.4 Field Documentation

10.338.4.1 bool Arc::WSRF::allocated_ [protected]

Associated SOAP message - it's SOAP message after all

10.338.4.2 bool Arc::WSRF::valid_ [protected]

true if `soap_` needs to be deleted in destructor

Referenced by operator `bool()`.

The documentation for this class was generated from the following file:

- `WSRF.h`

10.339 Arc::WSRFBBaseFault Class Reference

Base class for [WSRF](#) fault messages.

#include <WSRFBBaseFault.h> Inheritance diagram for Arc::WSRFBBaseFault::



Public Member Functions

- [WSRFBBaseFault](#) (SOAPEnvelope &soap)
- [WSRFBBaseFault](#) (const std::string &type)

Protected Member Functions

- void [set_namespaces](#) (void)

10.339.1 Detailed Description

Base class for [WSRF](#) fault messages. Use classes inherited from it for specific faults.

10.339.2 Constructor & Destructor Documentation

10.339.2.1 Arc::WSRFBBaseFault::WSRFBBaseFault (SOAPEnvelope & soap)

Constructor - creates object out of supplied SOAP tree.

10.339.2.2 Arc::WSRFBBaseFault::WSRFBBaseFault (const std::string & type)

Constructor - creates new [WSRF](#) fault

10.339.3 Member Function Documentation

10.339.3.1 void Arc::WSRFBBaseFault::set_namespaces (void) [protected]

set WS-ResourceProperties namespaces and default prefixes in SOAP message

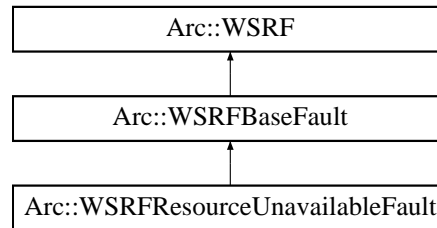
Reimplemented from [Arc::WSRF](#).

The documentation for this class was generated from the following file:

- WSRFBBaseFault.h

10.340 Arc::WSRFResourceUnavailableFault Class Reference

Inheritance diagram for Arc::WSRFResourceUnavailableFault::

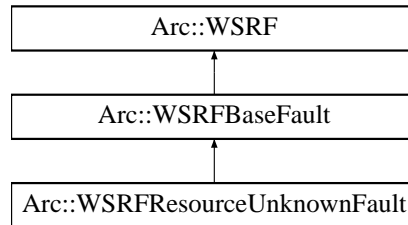


The documentation for this class was generated from the following file:

- WSRFBaseFault.h

10.341 Arc::WSRFResourceUnknownFault Class Reference

Inheritance diagram for Arc::WSRFResourceUnknownFault::



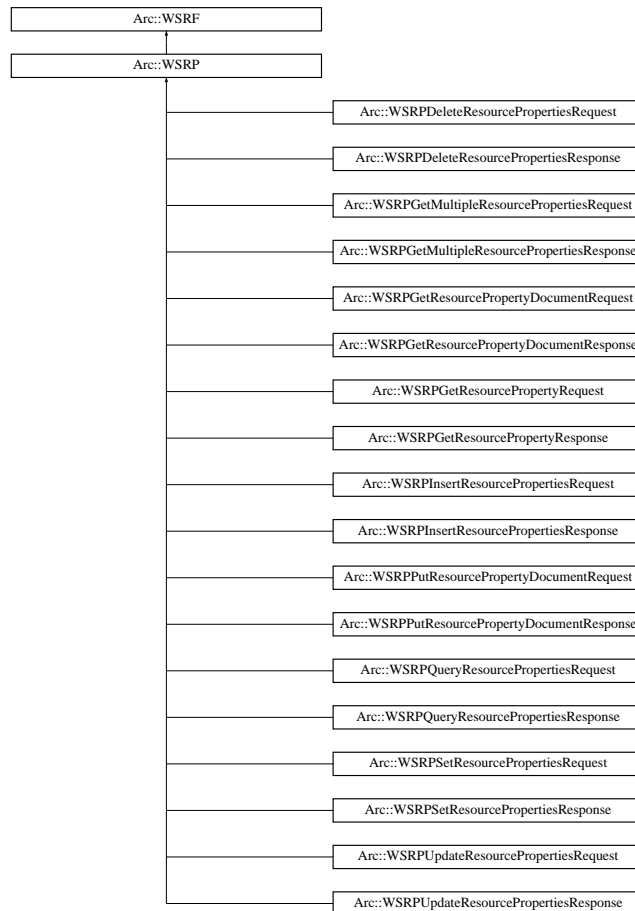
The documentation for this class was generated from the following file:

- `WSRFBaseFault.h`

10.342 Arc::WSRP Class Reference

Base class for WS-ResourceProperties structures.

#include <WSResourceProperties.h> Inheritance diagram for Arc::WSRP:



Public Member Functions

- [WSRP](#) (bool fault=false, const std::string &action="")
- [WSRP](#) (SOAPEnvelope &soap, const std::string &action="")

Protected Member Functions

- void [set_namespaces](#) (void)

10.342.1 Detailed Description

Base class for WS-ResourceProperties structures. Inheriting classes implement specific WS-ResourceProperties messages and their properties/elements. Refer to WS-ResourceProperties specifications for things specific to every message.

10.342.2 Constructor & Destructor Documentation

10.342.2.1 `Arc::WSRP::WSRP (bool fault = false, const std::string & action = "")`

Constructor - prepares object for creation of new [WSRP](#) request/response/fault

10.342.2.2 `Arc::WSRP::WSRP (SOAPEnvelope & soap, const std::string & action = "")`

Constructor - creates object out of supplied SOAP tree. It does not check if 'soap' represents valid WS-ResourceProperties structure. Actual check for validity of structure has to be done by derived class.

10.342.3 Member Function Documentation

10.342.3.1 `void Arc::WSRP::set_namespaces (void) [protected]`

set WS-ResourceProperties namespaces and default prefixes in SOAP message

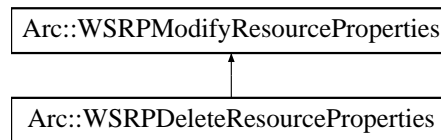
Reimplemented from [Arc::WSRF](#).

The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

10.343 Arc::WSRPDeleteResourceProperties Class Reference

Inheritance diagram for Arc::WSRPDeleteResourceProperties::

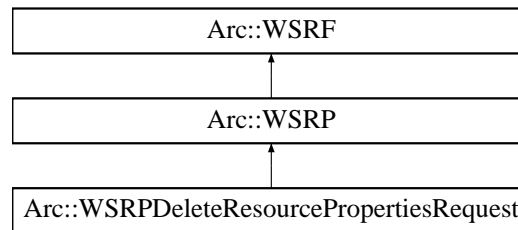


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.344 Arc::WSRPDeleteResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesRequest::

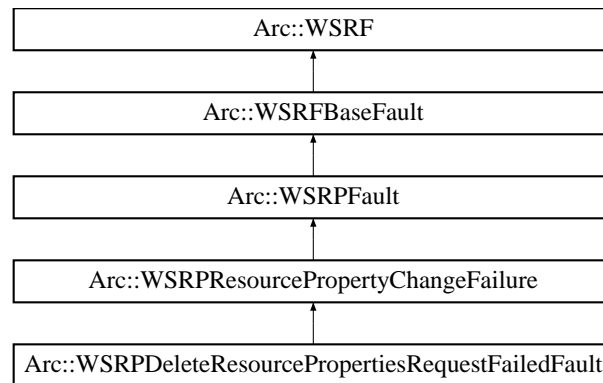


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.345 Arc::WSRPDeleteResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesRequestFailedFault::

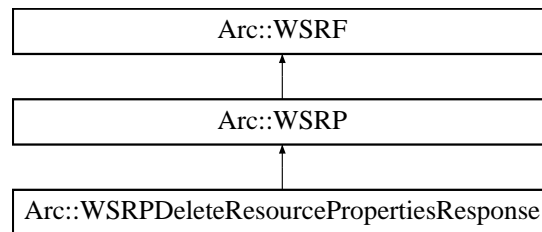


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.346 Arc::WSRPDeleteResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPDeleteResourcePropertiesResponse::



The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.347 Arc::WSRPFault Class Reference

Base class for WS-ResourceProperties faults.

#include <WSResourceProperties.h>Inheritance diagram for Arc::WSRPFault::



Public Member Functions

- [WSRPFault](#) (SOAPEnvelope &soap)
- [WSRPFault](#) (const std::string &type)

10.347.1 Detailed Description

Base class for WS-ResourceProperties faults.

10.347.2 Constructor & Destructor Documentation

10.347.2.1 Arc::WSRPFault::WSRPFault (SOAPEnvelope & soap)

Constructor - creates object out of supplied SOAP tree.

10.347.2.2 Arc::WSRPFault::WSRPFault (const std::string & type)

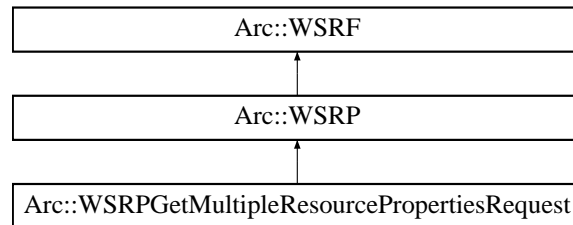
Constructor - creates new [WSRP](#) fault

The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.348 Arc::WSRPGetMultipleResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPGetMultipleResourcePropertiesRequest::

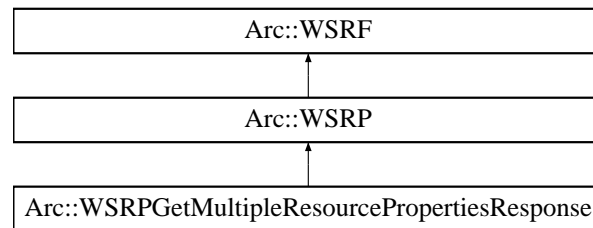


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.349 Arc::WSRPGetMultipleResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPGetMultipleResourcePropertiesResponse::

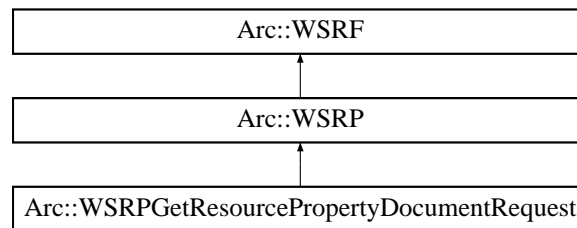


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.350 Arc::WSRPGetResourcePropertyDocumentRequest Class Reference

Inheritance diagram for Arc::WSRPGetResourcePropertyDocumentRequest::

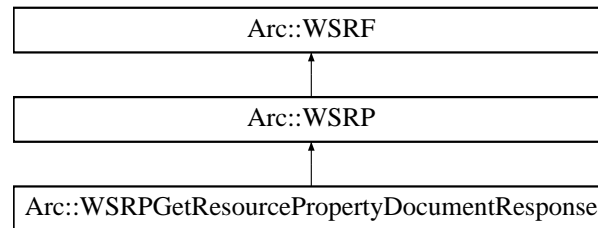


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.351 Arc::WSRPGetResourcePropertyDocumentResponse Class Reference

Inheritance diagram for Arc::WSRPGetResourcePropertyDocumentResponse::

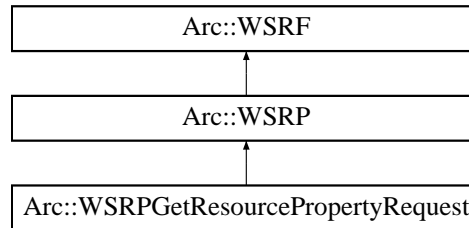


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.352 Arc::WSRPGetResourcePropertyRequest Class Reference

Inheritance diagram for Arc::WSRPGetResourcePropertyRequest::

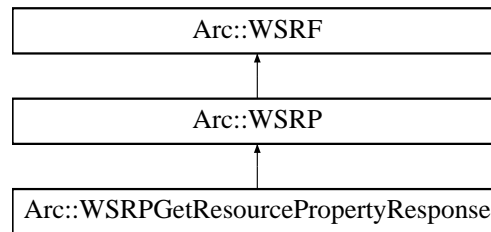


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

10.353 Arc::WSRPGetResourcePropertyResponse Class Reference

Inheritance diagram for Arc::WSRPGetResourcePropertyResponse::

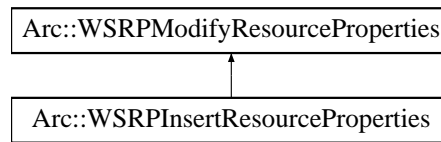


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.354 Arc::WSRPInsertResourceProperties Class Reference

Inheritance diagram for Arc::WSRPInsertResourceProperties::

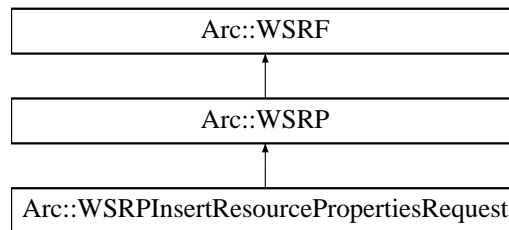


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.355 Arc::WSRPInsertResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPInsertResourcePropertiesRequest::

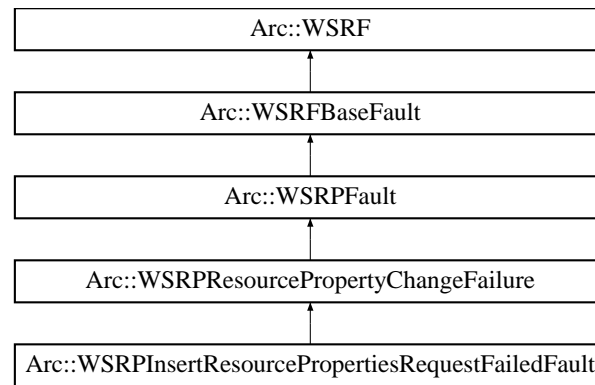


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.356 Arc::WSRPIInsertResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPIInsertResourcePropertiesRequestFailedFault::

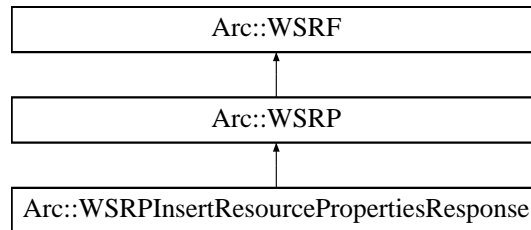


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.357 Arc::WSRPInsertResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPInsertResourcePropertiesResponse::

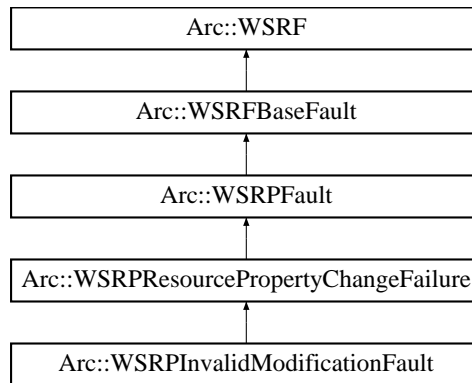


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.358 Arc::WSRPInvalidModificationFault Class Reference

Inheritance diagram for Arc::WSRPInvalidModificationFault::

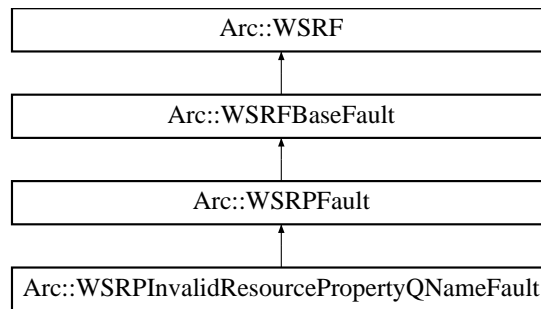


The documentation for this class was generated from the following file:

- `WSResourceProperties.h`

10.359 Arc::WSRPInvalidResourcePropertyQNameFault Class Reference

Inheritance diagram for Arc::WSRPInvalidResourcePropertyQNameFault::

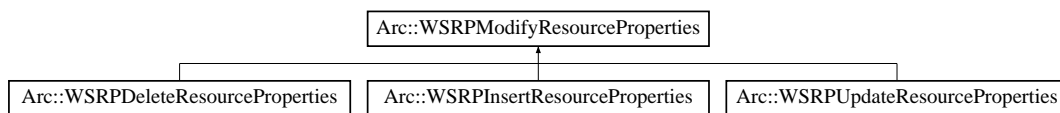


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.360 Arc::WSRPModifyResourceProperties Class Reference

Inheritance diagram for Arc::WSRPModifyResourceProperties::

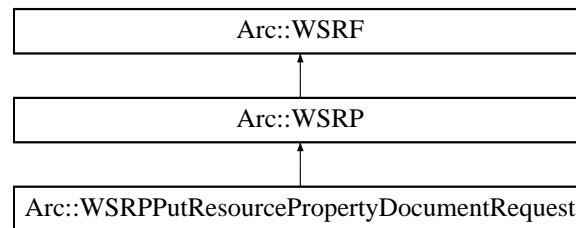


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.361 Arc::WSRPPutResourcePropertyDocumentRequest Class Reference

Inheritance diagram for Arc::WSRPPutResourcePropertyDocumentRequest::

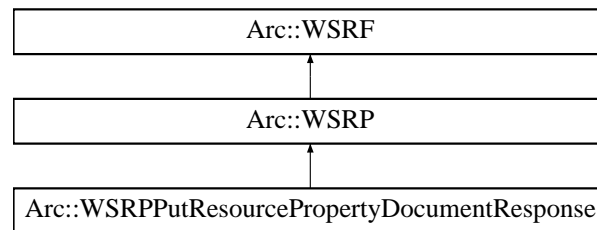


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.362 Arc::WSRPPutResourcePropertyDocumentResponse Class Reference

Inheritance diagram for Arc::WSRPPutResourcePropertyDocumentResponse::

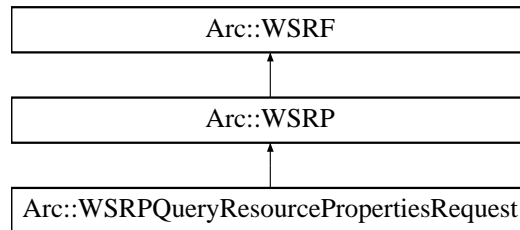


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.363 Arc::WSRPQueryResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPQueryResourcePropertiesRequest::

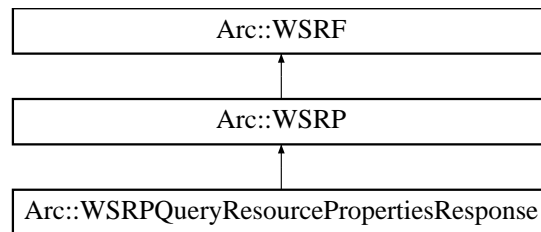


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.364 Arc::WSRPQueryResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPQueryResourcePropertiesResponse::



The documentation for this class was generated from the following file:

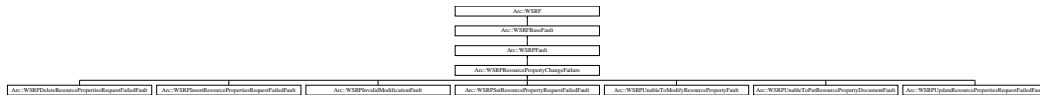
- WSResourceProperties.h

10.365 Arc::WSRPResourcePropertyChangeFailure Class Reference

#include <WSResourceProperties.h>Inheritance
Arc::WSRPResourcePropertyChangeFailure::

diagram

for



Public Member Functions

- [WSRPResourcePropertyChangeFailure](#) (SOAPEnvelope &soap)
- [WSRPResourcePropertyChangeFailure](#) (const std::string &type)

10.365.1 Detailed Description

Base class for WS-ResourceProperties faults which contain ResourcePropertyChangeFailure

10.365.2 Constructor & Destructor Documentation

10.365.2.1 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (SOAPEnvelope & soap) [inline]

Constructor - creates object out of supplied SOAP tree.

10.365.2.2 Arc::WSRPResourcePropertyChangeFailure::WSRPResourcePropertyChangeFailure (const std::string & type) [inline]

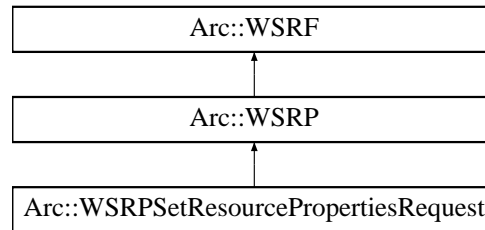
Constructor - creates new [WSRP](#) fault

The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.366 Arc::WSRPSetResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertiesRequest::

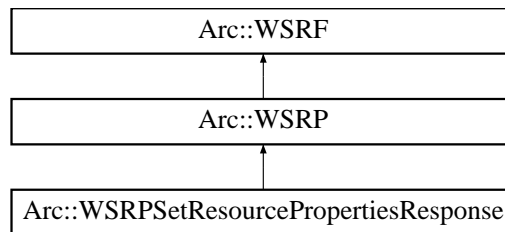


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.367 Arc::WSRPSetResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertiesResponse::

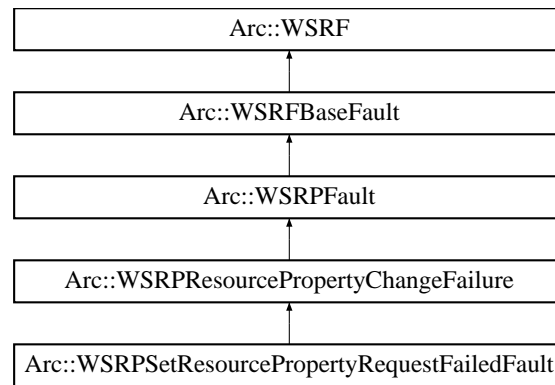


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.368 Arc::WSRPSetResourcePropertyRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPSetResourcePropertyRequestFailedFault:

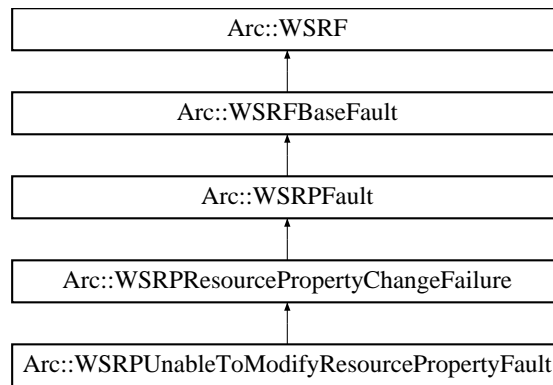


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.369 Arc::WSRPUUnableToModifyResourcePropertyFault Class Reference

Inheritance diagram for Arc::WSRPUUnableToModifyResourcePropertyFault:

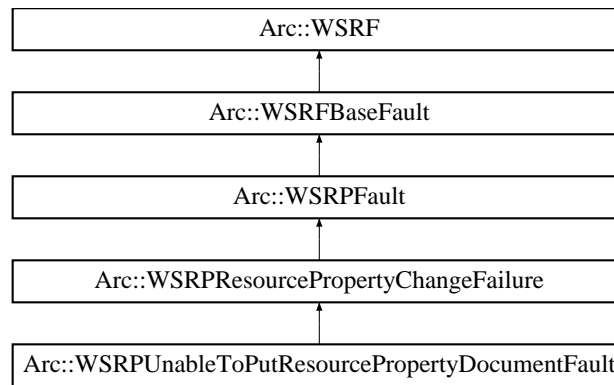


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.370 Arc::WSRPUnableToPutResourcePropertyDocumentFault Class Reference

Inheritance diagram for Arc::WSRPUnableToPutResourcePropertyDocumentFault::

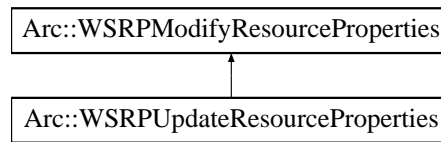


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.371 Arc::WSRPUUpdateResourceProperties Class Reference

Inheritance diagram for Arc::WSRPUUpdateResourceProperties::

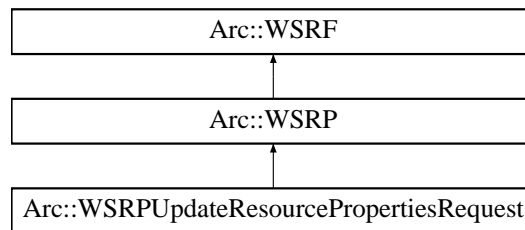


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.372 Arc::WSRPUUpdateResourcePropertiesRequest Class Reference

Inheritance diagram for Arc::WSRPUUpdateResourcePropertiesRequest::

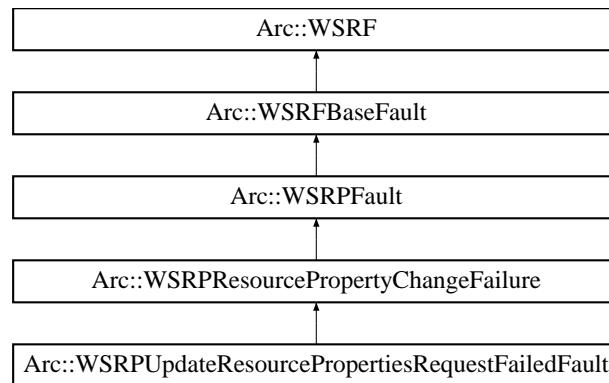


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.373 Arc::WSRPUUpdateResourcePropertiesRequestFailedFault Class Reference

Inheritance diagram for Arc::WSRPUUpdateResourcePropertiesRequestFailedFault::

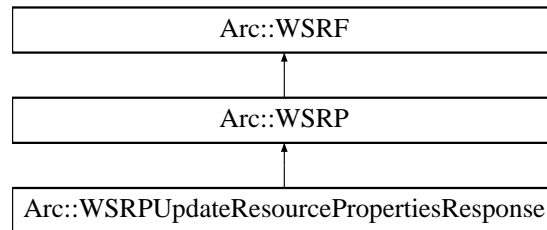


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.374 Arc::WSRPUUpdateResourcePropertiesResponse Class Reference

Inheritance diagram for Arc::WSRPUUpdateResourcePropertiesResponse::

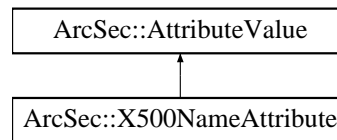


The documentation for this class was generated from the following file:

- WSResourceProperties.h

10.375 ArcSec::X500NameAttribute Class Reference

Inheritance diagram for ArcSec::X500NameAttribute::



Public Member Functions

- virtual std::string [encode](#) ()
- virtual std::string [getType](#) ()
- virtual std::string [getId](#) ()

10.375.1 Member Function Documentation

10.375.1.1 virtual std::string ArcSec::X500NameAttribute::encode () [inline, virtual]

encode the value in a string format

Implements [ArcSec::AttributeValue](#).

10.375.1.2 virtual std::string ArcSec::X500NameAttribute::getId () [inline, virtual]

Get the AttributeId of the <Attribute>

Implements [ArcSec::AttributeValue](#).

10.375.1.3 virtual std::string ArcSec::X500NameAttribute::getType () [inline, virtual]

Get the DataType of the <Attribute>

Implements [ArcSec::AttributeValue](#).

The documentation for this class was generated from the following file:

- X500NameAttribute.h

10.376 Arc::X509Token Class Reference

Class for manipulating X.509 Token [Profile](#).

```
#include <X509Token.h>
```

Public Types

- enum [X509TokenType](#)

Public Member Functions

- [X509Token](#) (SOAPEnvelope &soap, const std::string &keyfile="")
- [X509Token](#) (SOAPEnvelope &soap, const std::string &certfile, const std::string &keyfile, [X509TokenType](#) token_type=Signature)
- [~X509Token](#) (void)
- [operator bool](#) (void)
- bool [Authenticate](#) (const std::string &cafile, const std::string &capath)
- bool [Authenticate](#) (void)

10.376.1 Detailed Description

Class for manipulating X.509 Token [Profile](#). This class is for generating/consuming X.509 Token profile. Currently it is used by x509token handler (src/hed/pdc/x509tokensh/) It is not necessary to directly called this class. If we need to use X.509 Token functionality, we only need to configure the x509token handler into service and client.

10.376.2 Member Enumeration Documentation

10.376.2.1 enum Arc::X509Token::X509TokenType

X509TokeType is for distinguishing two types of operation. It is used as the parameter of constructor.

10.376.3 Constructor & Destructor Documentation

10.376.3.1 Arc::X509Token::X509Token (SOAPEnvelope & soap, const std::string & keyfile = "")

Constructor.Parse X509 Token information from SOAP header. X509 Token related information is extracted from SOAP header and stored in class variables. And then it the [X509Token](#) object will be used for authentication if the tokentype is Signature; otherwise if the tokentype is Encryption, the encrypted soap body will be decrypted and replaced by decrypted message. keyfile is only needed when the [X509Token](#) is encryption token

10.376.3.2 Arc::X509Token::X509Token (SOAPEnvelope & soap, const std::string & certfile, const std::string & keyfile, X509TokenType token_type = Signature)

Constructor. Add X509 Token information into the SOAP header. Generated token contains elements X509 token and signature, and is meant to be used for authentication on the consuming side.

Parameters:

soap The SOAP message to which the X509 Token will be inserted

certfile The certificate file which will be used to encrypt the SOAP body (if parameter tokentype is Encryption), or be used as <wsse:BinarySecurityToken/> (if parameter tokentype is Signature).

keyfile The key file which will be used to create signature. Not needed when create encryption.

tokentype Token type: Signature or Encryption.

10.376.3.3 Arc::X509Token::~~X509Token (void)

Deconstructor. Nothing to be done except finalizing the xmlsec library.

10.376.4 Member Function Documentation**10.376.4.1 bool Arc::X509Token::Authenticate (void)**

Check signature by using the cert information in soap message. Only the signature itself is checked, and it is not guranteed that the certificate which is supposed to check the signature is trusted.

10.376.4.2 bool Arc::X509Token::Authenticate (const std::string & *cafile*, const std::string & *capath*)

Check signature by using the certificare information in [X509Token](#) which is parsed by the constructor, and the trusted certificates specified as one of the two parameters. Not only the signature (in the [X509Token](#)) itself is checked, but also the certificate which is supposed to check the signature needs to be trused (which means the certificate is issued by the ca certificate from CA file or CA directory). At least one the the two parameters should be set.

Parameters:

cafile The CA file

capath The CA directory

Returns:

true if authentication passes; otherwise false

10.376.4.3 Arc::X509Token::operator bool (void)

Returns true of constructor succeeded

The documentation for this class was generated from the following file:

- X509Token.h

10.377 Arc::XmlContainer Class Reference

The documentation for this class was generated from the following file:

- XmlContainer.h

10.378 Arc::XmlDatabase Class Reference

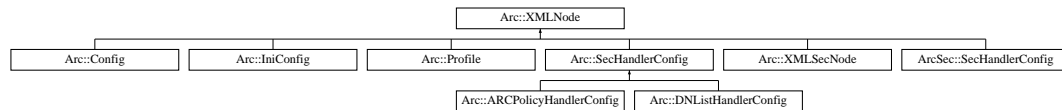
The documentation for this class was generated from the following file:

- XmlDatabase.h

10.379 Arc::XMLNode Class Reference

Wrapper for LibXML library Tree interface.

#include <arc/XMLNode.h>Inheritance diagram for Arc::XMLNode::



Public Member Functions

- [XMLNode](#) (void)
- [XMLNode](#) (const [XMLNode](#) &node)
- [XMLNode](#) (const std::string &xml)
- [XMLNode](#) (const char *xml, int len=-1)
- [XMLNode](#) (long ptr_addr)
- [XMLNode](#) (const [NS](#) &ns, const char *name)
- [~XMLNode](#) (void)
- void [New](#) ([XMLNode](#) &node) const
- void [Exchange](#) ([XMLNode](#) &node)
- void [Move](#) ([XMLNode](#) &node)
- void [Swap](#) ([XMLNode](#) &node)
- [operator bool](#) (void) const
- bool [operator!](#) (void) const
- bool [operator==](#) (const [XMLNode](#) &node)
- bool [operator!=](#) (const [XMLNode](#) &node)
- bool [Same](#) (const [XMLNode](#) &node)
- bool [operator==](#) (bool val)
- bool [operator!=](#) (bool val)
- bool [operator==](#) (const std::string &str)
- bool [operator!=](#) (const std::string &str)
- bool [operator==](#) (const char *str)
- bool [operator!=](#) (const char *str)
- [XMLNode Child](#) (int n=0)
- [XMLNode operator\[\]](#) (const char *name) const
- [XMLNode operator\[\]](#) (const std::string &name) const
- [XMLNode operator\[\]](#) (int n) const
- void [operator++](#) (void)
- void [operator--](#) (void)
- int [Size](#) (void) const
- [XMLNode Get](#) (const std::string &name) const
- std::string [Name](#) (void) const
- std::string [Prefix](#) (void) const
- std::string [FullName](#) (void) const
- std::string [Namespace](#) (void) const
- void [Prefix](#) (const std::string &prefix, int recursion=0)
- void [StripNamespace](#) (int recursion=0)
- void [Name](#) (const char *name)

- void [Name](#) (const std::string &name)
- void [GetXML](#) (std::string &out_xml_str, bool user_friendly=false) const
- void [GetXML](#) (std::string &out_xml_str, const std::string &encoding, bool user_friendly=false) const
- void [GetDoc](#) (std::string &out_xml_str, bool user_friendly=false) const
- [operator std::string](#) (void) const
- [XMLNode](#) & [operator=](#) (const char *content)
- [XMLNode](#) & [operator=](#) (const std::string &content)
- void [Set](#) (const std::string &content)
- [XMLNode](#) & [operator=](#) (const [XMLNode](#) &node)
- [XMLNode](#) [Attribute](#) (int n=0)
- [XMLNode](#) [Attribute](#) (const char *name)
- [XMLNode](#) [Attribute](#) (const std::string &name)
- [XMLNode](#) [NewAttribute](#) (const char *name)
- [XMLNode](#) [NewAttribute](#) (const std::string &name)
- int [AttributesSize](#) (void) const
- void [Namespaces](#) (const [NS](#) &namespaces, bool keep=false, int recursion=-1)
- [NS](#) [Namespaces](#) (void)
- std::string [NamespacePrefix](#) (const char *urn)
- [XMLNode](#) [NewChild](#) (const char *name, int n=-1, bool global_order=false)
- [XMLNode](#) [NewChild](#) (const std::string &name, int n=-1, bool global_order=false)
- [XMLNode](#) [NewChild](#) (const char *name, const [NS](#) &namespaces, int n=-1, bool global_order=false)
- [XMLNode](#) [NewChild](#) (const std::string &name, const [NS](#) &namespaces, int n=-1, bool global_order=false)
- [XMLNode](#) [NewChild](#) (const [XMLNode](#) &node, int n=-1, bool global_order=false)
- void [Replace](#) (const [XMLNode](#) &node)
- void [Destroy](#) (void)
- [XMLNodeList](#) [Path](#) (const std::string &path)
- [XMLNodeList](#) [XPathLookup](#) (const std::string &xpathExpr, const [NS](#) &nsList)
- [XMLNode](#) [GetRoot](#) (void)
- [XMLNode](#) [Parent](#) (void)
- bool [SaveToFile](#) (const std::string &file_name) const
- bool [SaveToStream](#) (std::ostream &out) const
- bool [ReadFromFile](#) (const std::string &file_name)
- bool [ReadFromStream](#) (std::istream &in)
- bool [Validate](#) (const std::string &schema_file, std::string &err_msg)
- bool [Validate](#) ([XMLNode](#) schema_doc, std::string &err_msg)

Protected Member Functions

- [XMLNode](#) (xmlNodePtr node)
- bool [Validate](#) (xmlSchemaPtr schema, std::string &err_msg)

Static Protected Member Functions

- static void [LogError](#) (void *ctx, const char *msg,...)

Protected Attributes

- bool [is_owner_](#)
- bool [is_temporary_](#)

10.379.1 Detailed Description

Wrapper for LibXML library Tree interface. This class wraps XML Node, Document and Property/Attribute structures. Each instance serves as pointer to actual LibXML element and provides convenient (for chosen purpose) methods for manipulating it. This class has no special ties to LibXML library and may be easily rewritten for any XML parser which provides interface similar to LibXML Tree. It implements only small subset of XML capabilities, which is probably enough for performing most of useful actions. This class also filters out (usually) useless textual nodes which are often used to make XML documents human-readable.

10.379.2 Constructor & Destructor Documentation

10.379.2.1 `Arc::XMLNode::XMLNode (xmlNodePtr node) [inline, protected]`

Protected constructor for inherited classes. Creates instance and links to existing LibXML structure. Acquired structure is not owned by class instance. If there is need to completely pass control of LibXML document to then instance's `is_owner_` variable has to be set to true.

10.379.2.2 `Arc::XMLNode::XMLNode (void) [inline]`

Constructor of invalid node. Created instance does not point to XML element. All methods are still allowed for such instance but produce no results.

10.379.2.3 `Arc::XMLNode::XMLNode (const XMLNode & node) [inline]`

Copies existing instance. Underlying XML element is NOT copied. Ownership is NOT inherited. Strictly speaking there should be no const here - but that conflicts with C++.

10.379.2.4 `Arc::XMLNode::XMLNode (const std::string & xml)`

Creates XML document structure from textual representation of XML document. Created structure is pointed and owned by constructed instance.

10.379.2.5 `Arc::XMLNode::XMLNode (const char * xml, int len = -1)`

Creates XML document structure from textual representation of XML document. Created structure is pointed and owned by constructed instance.

10.379.2.6 `Arc::XMLNode::XMLNode (const NS & ns, const char * name)`

Creates empty XML document structure with specified namespaces. Created XML contains only root element named 'name'. Created structure is pointed and owned by constructed instance.

10.379.2.7 Arc::XMLNode::~~XMLNode (void)

Destructor. Also destroys underlying XML document if owned by this instance

10.379.3 Member Function Documentation**10.379.3.1 XMLNode Arc::XMLNode::Child (int *n* = 0)**

Returns [XMLNode](#) instance representing *n*-th child of XML element. If such does not exist invalid [XMLNode](#) instance is returned

10.379.3.2 void Arc::XMLNode::Destroy (void)

Destroys underlying XML element. XML element is unlinked from XML tree and destroyed. After this operation [XMLNode](#) instance becomes invalid

10.379.3.3 void Arc::XMLNode::Exchange (XMLNode & *node*)

Exchanges XML (sub)trees. The following combinations are possible:

- If both this and *node* are referring owned XML tree (top level node) then references are simply exchanged. This operation is fast.
- If both this and *node* are referring to XML (sub)tree of different documents then (sub)trees are exchanged between documents.
- If both this and *node* are referring to XML (sub)tree of same document then (sub)trees are moved inside document.

The main reason for this method is to provide an effective way to insert one XML document inside another. One should take into account that if any of the exchanged nodes is top level it must be also the owner of the document. Otherwise this method will fail. If both nodes are top level owners and/or invalid nodes then this method is identical to [Swap\(\)](#).

10.379.3.4 void Arc::XMLNode::GetXML (std::string & *out_xml_str*, const std::string & *encoding*, bool *user_friendly* = false) const

Get string representation of XML subtree. Fills *out_xml_str* with this instance XML subtree textual representation if the XML subtree corresponds to the encoding format specified in the argument, e.g. utf-8.

10.379.3.5 static void Arc::XMLNode::LogError (void * *ctx*, const char * *msg*, ...) [static, protected]

printf-like callback for libxml

10.379.3.6 void Arc::XMLNode::Move (XMLNode & *node*)

Moves content of this XML (sub)tree to *node*. This operation is similar to [New\(\)](#) except that XML (sub)tree referred by this is destroyed. This method is more effective than combination of [New\(\)](#) and [Destroy\(\)](#)

because internally it is optimized not to copy data if not needed. The main purpose of this is to effectively extract part of XML document.

10.379.3.7 void Arc::XMLNode::Namespaces (const NS & namespaces, bool keep = false, int recursion = -1)

Assigns namespaces of XML document at point specified by this instance. If namespace already exists it gets new prefix. New namespaces are added. It is useful to apply this method to XML being processed in order to refer to it's elements by known prefix. If keep is set to false existing namespace definition residing at this instance and below are removed (default behavior). If recursion is set to positive number then depth of prefix replacement is limited by this number (0 limits it to this node only). For unlimited recursion use -1. If recursion is limited then value of keep is ignored and existing namespaces are always kept.

10.379.3.8 void Arc::XMLNode::New (XMLNode & node) const

Creates a copy of XML (sub)tree. If object does not represent whole document - top level document is created. 'node' becomes a pointer owning new XML document.

10.379.3.9 XMLNode Arc::XMLNode::NewChild (const XMLNode & node, int n = -1, bool global_order = false)

Link a copy of supplied XML node as child. Returns instance referring to new child. XML element is a copy of supplied one but not owned by returned instance

10.379.3.10 XMLNode Arc::XMLNode::NewChild (const char * name, const NS & namespaces, int n = -1, bool global_order = false)

Creates new child XML element at specified position with specified name and namespaces. For more information look at [NewChild\(const char*,int,bool\)](#)

10.379.3.11 XMLNode Arc::XMLNode::NewChild (const char * name, int n = -1, bool global_order = false)

Creates new child XML element at specified position with specified name. Default is to put it at end of list. If global_order is true position applies to whole set of children, otherwise only to children of same name. Returns created node.

Referenced by NewChild().

10.379.3.12 void Arc::XMLNode::operator++ (void)

Convenience operator to switch to next element of same name. If there is no such node this object becomes invalid.

10.379.3.13 void Arc::XMLNode::operator-- (void)

Convenience operator to switch to previous element of same name. If there is no such node this object becomes invalid.

10.379.3.14 XMLNode& Arc::XMLNode::operator= (const XMLNode & node)

Make instance refer to another XML node. Ownership is not inherited. Due to nature of [XMLNode](#) there should be no const here, but that does not fit into C++.

10.379.3.15 XMLNode Arc::XMLNode::operator[] (int n) const

Returns [XMLNode](#) instance representing n-th node in sequence of siblings of same name. Its main purpose is to be used to retrieve an element in an array of children of the same name like node["name"][5].

This method should not be marked const because obtaining unrestricted [XMLNode](#) of child element allows modification of underlying XML tree. But in order to keep const in other places non-const-handling is passed to programmer. Otherwise C++ compiler goes nuts.

10.379.3.16 XMLNode Arc::XMLNode::operator[] (const std::string & name) const [inline]

Returns [XMLNode](#) instance representing first child element with specified name. Similar to operator[](const char *name) const.

References operator[]().

10.379.3.17 XMLNode Arc::XMLNode::operator[] (const char * name) const

Returns [XMLNode](#) instance representing first child element with specified name. Name may be "namespace_prefix:name", "namespace_uri:name" or simply "name". In last case namespace is ignored. If such node does not exist invalid [XMLNode](#) instance is returned. This method should not be marked const because obtaining unrestricted [XMLNode](#) of child element allows modification of underlying XML tree. But in order to keep const in other places non-const-handling is passed to programmer. Otherwise C++ compiler goes nuts.

Referenced by Get(), and operator[]().

10.379.3.18 XMLNodeList Arc::XMLNode::Path (const std::string & path)

Collects nodes corresponding to specified path. This is a convenience function to cover common use of XPath but without performance hit. Path is made of node_name[/node_name[...]] and is relative to current node. node_names are treated in same way as in operator[].

Returns:

all nodes which are represented by path.

10.379.3.19 void Arc::XMLNode::Prefix (const std::string & prefix, int recursion = 0)

Assigns namespace prefix to XML node(s). The 'recursion' allows to assign prefixes recursively. Setting it to -1 allows for unlimited recursion. And 0 limits it to this node.

10.379.3.20 void Arc::XMLNode::Swap (XMLNode & node)

Swaps XML (sub)trees to which this and node refer. For XML subtrees this method is not anyhow different then using the combination

```
XMLNode tmp=*this; *this=node; node=tmp;
```

But in case of either this or node owning XML document ownership is swapped too. And this is the main purpose of this method.

10.379.3.21 `bool Arc::XMLNode::Validate (XMLNode schema_doc, std::string & err_msg)`

XML schema validation against the schema XML document defined as argument

10.379.3.22 `bool Arc::XMLNode::Validate (xmlSchemaPtr schema, std::string & err_msg)` `[protected]`

Convenience method for XML validation

10.379.3.23 `XMLNodeList Arc::XMLNode::XPathLookup (const std::string & xpathExpr, const NS & nsList)`

Uses xPath to look up the whole xml structure,. Returns a list of [XMLNode](#) points. The xpathExpr should be like `"/xx:child1/"` which indicates the namespace and node that you would like to find; The nsList is the namespace the result should belong to (e.g. `xx="uri:test"`). [Query](#) is run on whole XML document but only the elements belonging to this XML subtree are returned.

10.379.4 Field Documentation

10.379.4.1 `bool Arc::XMLNode::is_owner_` `[protected]`

If true node is owned by this instance - hence released in destructor. Normally that may be true only for top level node of XML document.

The documentation for this class was generated from the following file:

- XMLNode.h

10.380 Arc::XMLNodeContainer Class Reference

Container for multiple [XMLNode](#) elements.

```
#include <arc/XMLNode.h>
```

Public Member Functions

- [XMLNodeContainer](#) (void)
- [XMLNodeContainer](#) (const [XMLNodeContainer](#) &)
- [XMLNodeContainer](#) & operator= (const [XMLNodeContainer](#) &)
- void [Add](#) (const [XMLNode](#) &)
- void [Add](#) (const std::list< [XMLNode](#) > &)
- void [AddNew](#) (const [XMLNode](#) &)
- void [AddNew](#) (const std::list< [XMLNode](#) > &)
- int [Size](#) (void) const
- [XMLNode](#) operator[] (int)
- std::list< [XMLNode](#) > [Nodes](#) (void)

10.380.1 Detailed Description

Container for multiple [XMLNode](#) elements.

10.380.2 Constructor & Destructor Documentation

10.380.2.1 Arc::XMLNodeContainer::XMLNodeContainer (const XMLNodeContainer &)

Copy constructor. Add nodes from argument. Nodes owning XML document are copied using [AddNew\(\)](#). Not owning nodes are linked using [Add\(\)](#) method.

10.380.3 Member Function Documentation

10.380.3.1 void Arc::XMLNodeContainer::Add (const XMLNode &)

Link XML subtree referred by node to container. XML tree must be available as long as this object is used.

10.380.3.2 void Arc::XMLNodeContainer::AddNew (const XMLNode &)

Copy XML subtree referenced by node to container. After this operation container refers to independent XML document. This document is deleted when container is destroyed.

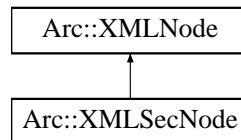
The documentation for this class was generated from the following file:

- XMLNode.h

10.381 Arc::XMLSecNode Class Reference

Extends [XMLNode](#) class to support XML security operation.

`#include <XMLSecNode.h>` Inheritance diagram for Arc::XMLSecNode::



Public Member Functions

- [XMLSecNode](#) ([XMLNode](#) &node)
- void [AddSignatureTemplate](#) (const std::string &id_name, const SignatureMethod sign_method, const std::string &incl_namespaces="")
- bool [SignNode](#) (const std::string &privkey_file, const std::string &cert_file)
- bool [VerifyNode](#) (const std::string &id_name, const std::string &ca_file, const std::string &ca_path, bool verify_trusted=true)
- bool [EncryptNode](#) (const std::string &cert_file, const SymEncryptionType encrpt_type)
- bool [DecryptNode](#) (const std::string &privkey_file, [XMLNode](#) &decrypted_node)

10.381.1 Detailed Description

Extends [XMLNode](#) class to support XML security operation. All [XMLNode](#) methods are exposed by inheriting from [XMLNode](#). [XMLSecNode](#) itself does not own node, instead it uses the node from the base class [XMLNode](#).

10.381.2 Constructor & Destructor Documentation

10.381.2.1 Arc::XMLSecNode::XMLSecNode (XMLNode & node)

Create a object based on an [XMLNode](#) instance.

10.381.3 Member Function Documentation

10.381.3.1 void Arc::XMLSecNode::AddSignatureTemplate (const std::string & id_name, const SignatureMethod sign_method, const std::string & incl_namespaces = "")

Add the signature template for later signing.

Parameters:

id_name The identifier name under this node which will be used for the <Signature> to refer to.

sign_method The sign method for signing. Two options now, RSA_SHA1, DSA_SHA1

10.381.3.2 **bool Arc::XMLSecNode::DecryptNode** (const std::string & *privkey_file*, XMLNode & *decrypted_node*)

Decrypt the <xenc:EncryptedData/> under this node, the decrypted node will be output in the second argument of DecryptNode method. And the <xenc:EncryptedData/> under this node will be removed after decryption.

Parameters:

privkey_file The private key file, which is used for decrypting
decrypted_node Output the decrypted node

10.381.3.3 **bool Arc::XMLSecNode::EncryptNode** (const std::string & *cert_file*, const SymEncryptionType *encript_type*)

Encrypt this node, after encryption, this node will be replaced by the encrypted node

Parameters:

cert_file The certificate file, the public key parsed from this certificate is used to encrypted the symmetric key, and then the symmetric key is used to encrypted the node
encript_type The encryption type when encrypting the node, four option in SymEncryptionType
verify_trusted Verify trusted certificates or not. If set to false, then only the signature will be checked (by using the public key from KeyInfo).

10.381.3.4 **bool Arc::XMLSecNode::SignNode** (const std::string & *privkey_file*, const std::string & *cert_file*)

Sign this node (identified by id_name).

Parameters:

privkey_file The private key file. The private key is used for signing
cert_file The certificate file. The certificate is used as the <KeyInfo> part of the <Signature>; <KeyInfo> will be used for the other end to verify this <Signature>
incl_namespaces InclusiveNamespaces for Tranform in Signature

10.381.3.5 **bool Arc::XMLSecNode::VerifyNode** (const std::string & *id_name*, const std::string & *ca_file*, const std::string & *ca_path*, bool *verify_trusted* = true)

Verify the signature under this node

Parameters:

id_name The id of this node, which is used for identifying the node
ca_file The CA file which used as trused certificate when verify the certificate in the <KeyInfo> part of <Signature>
ca_path The CA directory; either ca_file or ca_path should be set.

The documentation for this class was generated from the following file:

- XMLSecNode.h

Chapter 11

File Documentation

11.1 BrokerPlugin.h File Reference

Plugin, loader and argument classes for broker specialisation. `#include <arc/loader/Loader.h>`
`#include <arc/loader/Plugin.h>`

Data Structures

- class [Arc::BrokerPluginArgument](#)
- class [Arc::BrokerPlugin](#)
- class [Arc::BrokerPluginLoader](#)

Namespaces

- namespace [Arc](#)

11.1.1 Detailed Description

Plugin, loader and argument classes for broker specialisation.

11.2 EntityRetrieverPlugin.h File Reference

Plugin, loader and argument classes for EntityRetriever specialisation. `#include <list>`

```
#include <map>
#include <set>
#include <string>
#include <arc/UserConfig.h>
#include <arc/compute/Endpoint.h>
#include <arc/compute/EndpointQueryingStatus.h>
#include <arc/compute/ExecutionTarget.h>
#include <arc/loader/Loader.h>
#include <arc/loader/FinderLoader.h>
```

Data Structures

- class [Arc::EndpointQueryOptions< T >](#)
Options controlling the query process.
- class [Arc::EndpointQueryOptions< Endpoint >](#)
The [EntityRetriever<Endpoint>](#) (a.k.a. [ServiceEndpointRetriever](#)) needs different options.
- class [Arc::EntityRetrieverPlugin< T >](#)
- class [Arc::EntityRetrieverPluginLoader< T >](#)
- class [Arc::ServiceEndpointRetrieverPlugin](#)
- class [Arc::TargetInformationRetrieverPlugin](#)
- class [Arc::JobListRetrieverPlugin](#)

Namespaces

- namespace [Arc](#)

11.2.1 Detailed Description

Plugin, loader and argument classes for EntityRetriever specialisation.

11.3 ExecutionTarget.h File Reference

Structures holding resource information. #include <list>

```
#include <map>
#include <set>
#include <string>
#include <arc/DateTime.h>
#include <arc/compute/Endpoint.h>
#include <arc/URL.h>
#include <arc/Utils.h>
#include <arc/compute/GLUE2Entity.h>
#include <arc/compute/JobDescription.h>
#include <arc/compute/Software.h>
#include <arc/compute/SubmissionStatus.h>
```

Data Structures

- class [Arc::ApplicationEnvironment](#)
ApplicationEnvironment.
- class [Arc::LocationAttributes](#)
- class [Arc::AdminDomainAttributes](#)
- class [Arc::ExecutionEnvironmentAttributes](#)
- class [Arc::ComputingManagerAttributes](#)
- class [Arc::ComputingShareAttributes](#)
- class [Arc::ComputingEndpointAttributes](#)
- class [Arc::ComputingServiceAttributes](#)
- class [Arc::LocationType](#)
- class [Arc::AdminDomainType](#)
- class [Arc::ExecutionEnvironmentType](#)
- class [Arc::ComputingManagerType](#)
- class [Arc::ComputingShareType](#)
- class [Arc::ComputingEndpointType](#)
- class [Arc::ComputingServiceType](#)
- class [Arc::ExecutionTarget](#)
ExecutionTarget.

Namespaces

- namespace [Arc](#)

11.3.1 Detailed Description

Structures holding resource information.

11.4 GLUE2Entity.h File Reference

template class for GLUE2 entities. `#include <arc/Utils.h>`

Data Structures

- class [Arc::GLUE2Entity< T >](#)

Namespaces

- namespace [Arc](#)

11.4.1 Detailed Description

template class for GLUE2 entities.

11.5 JobControllerPlugin.h File Reference

Plugin, loader and argument classes for job controller specialisation. `#include <list>`

```
#include <vector>
```

```
#include <string>
```

```
#include <arc/URL.h>
```

```
#include <arc/compute/Job.h>
```

```
#include <arc/loader/Loader.h>
```

```
#include <arc/loader/Plugin.h>
```

Data Structures

- class [Arc::JobControllerPlugin](#)
- class [Arc::JobControllerPluginLoader](#)
- class [Arc::JobControllerPluginArgument](#)

Namespaces

- namespace [Arc](#)

11.5.1 Detailed Description

Plugin, loader and argument classes for job controller specialisation.

11.6 JobDescription.h File Reference

Classes related to creating JobDescription objects. #include <list>

```
#include <vector>
#include <string>
#include <arc/DateTime.h>
#include <arc/XMLNode.h>
#include <arc/URL.h>
#include <arc/compute/Software.h>
```

Data Structures

- class [Arc::OptIn< T >](#)
- class [Arc::Range< T >](#)
- class [Arc::ScalableTime< T >](#)
- class [Arc::ScalableTime< int >](#)
- class [Arc::JobIdentificationType](#)
Job identification.
- class [Arc::ExecutableType](#)
Executable.
- class [Arc::RemoteLoggingType](#)
Remote logging.
- class [Arc::NotificationType](#)
- class [Arc::ApplicationType](#)
- class [Arc::SlotRequirementType](#)
- class [Arc::DiskSpaceRequirementType](#)
- class [Arc::ParallelEnvironmentType](#)
- class [Arc::ResourcesType](#)
- class [Arc::SourceType](#)
- class [Arc::TargetType](#)
- class [Arc::InputFileType](#)
- class [Arc::OutputFileType](#)
- class [Arc::DataStagingType](#)
- class [Arc::JobDescriptionResult](#)
- class [Arc::JobDescription](#)

Namespaces

- namespace [Arc](#)

11.6.1 Detailed Description

Classes related to creating JobDescription objects.

11.7 JobDescriptionParserPlugin.h File Reference

Plugin, loader and argument classes for job description parser specialisation. `#include <map>`

`#include <string>`

`#include <arc/loader/Loader.h>`

`#include <arc/loader/Plugin.h>`

Data Structures

- class [Arc::JobDescriptionParserPluginResult](#)
- class [Arc::JobDescriptionParserPlugin](#)
Abstract class for the different parsers.
- class [Arc::JobDescriptionParserPluginLoader](#)
- class [Arc::JobDescriptionParserPluginLoader::iterator](#)

Namespaces

- namespace [Arc](#)

11.7.1 Detailed Description

Plugin, loader and argument classes for job description parser specialisation.

11.8 Software.h File Reference

Software and SoftwareRequirement classes. #include <list>

```
#include <utility>
```

```
#include <string>
```

```
#include <ostream>
```

```
#include <arc/Logger.h>
```

Data Structures

- class [Arc::Software](#)
Used to represent software (names and version) and comparison.
- class [Arc::SoftwareRequirement](#)
Class used to express and resolve version requirements on software.

Namespaces

- namespace [Arc](#)

11.8.1 Detailed Description

Software and SoftwareRequirement classes.

11.9 SubmitterPlugin.h File Reference

Plugin, loader and argument classes for submitter specialisation. `#include <list>`

```
#include <map>
#include <string>
#include <arc/URL.h>
#include <arc/loader/Loader.h>
#include <arc/loader/Plugin.h>
#include <arc/compute/EntityRetriever.h>
#include <arc/compute/Job.h>
#include <arc/compute/JobDescription.h>
#include <arc/compute/SubmissionStatus.h>
#include <arc/data/DataHandle.h>
```

Data Structures

- class [Arc::SubmitterPlugin](#)
Base class for the SubmitterPlugins.
- class [Arc::SubmitterPluginLoader](#)
- class [Arc::SubmitterPluginArgument](#)

Namespaces

- namespace [Arc](#)

11.9.1 Detailed Description

Plugin, loader and argument classes for submitter specialisation.

11.10 TestACCControl.h File Reference

Classes for controlling output of compute test plugins. #include <list>

```
#include <string>
#include <arc/compute/Endpoint.h>
#include <arc/Thread.h>
#include <arc/URL.h>
#include <arc/compute/EndpointQueryingStatus.h>
#include <arc/compute/ExecutionTarget.h>
#include <arc/compute/Job.h>
#include <arc/compute/JobState.h>
#include <arc/compute/SubmissionStatus.h>
```

Data Structures

- class [Arc::BrokerPluginTestACCControl](#)
- class [Arc::JobDescriptionParserPluginTestACCControl](#)
- class [Arc::JobControllerPluginTestACCControl](#)
- class [Arc::SubmitterPluginTestACCControl](#)
- class [Arc::JobStateTEST](#)
- class [Arc::JobListRetrieverPluginTESTControl](#)
- class [Arc::ServiceEndpointRetrieverPluginTESTControl](#)
- class [Arc::TargetInformationRetrieverPluginTESTControl](#)

Namespaces

- namespace [Arc](#)

11.10.1 Detailed Description

Classes for controlling output of compute test plugins.

Index

- ~Counter
 - Arc::Counter, [176](#)
- ~IntraProcessCounter
 - Arc::IntraProcessCounter, [314](#)
- ~JobControllerPluginLoader
 - Arc::JobControllerPluginLoader, [327](#)
- ~JobDescriptionParserPluginLoader
 - Arc::JobDescriptionParserPluginLoader, [334](#)
- ~Loader
 - Arc::Loader, [360](#)
- ~MCCLoader
 - Arc::MCCLoader, [383](#)
- ~Message
 - Arc::Message, [389](#)
- ~PayloadRaw
 - Arc::PayloadRaw, [422](#)
- ~PayloadStream
 - Arc::PayloadStream, [429](#)
- ~Plexer
 - Arc::Plexer, [443](#)
- ~SAMLToken
 - Arc::SAMLToken, [491](#)
- ~SOAPMessage
 - Arc::SOAPMessage, [518](#)
- ~SubmitterPluginLoader
 - Arc::SubmitterPluginLoader, [546](#)
- ~WSAEndpointReference
 - Arc::WSAEndpointReference, [612](#)
- ~X509Token
 - Arc::X509Token, [659](#)
- ~XMLNode
 - Arc::XMLNode, [664](#)
- Abandon
 - Arc::Run, [484](#)
- Acquire
 - Arc::DelegationConsumer, [208](#)
 - Arc::FileAccessContainer, [281](#)
 - Arc::InformationContainer, [303](#)
- acquire
 - Arc::FileLock, [283](#)
- acquireDelegation
 - Arc::ClientX509Delegation, [150](#)
- Action
 - Arc::WSAHeader, [615](#)
- ActivityOldID
 - Arc::JobIdentificationType, [339](#)
- Add
 - Arc::MessageContext, [396](#)
 - Arc::XMLNodeContainer, [669](#)
- add
 - Arc::Adler32Sum, [94](#)
 - Arc::Checksum, [132](#)
 - Arc::ChecksumAny, [136](#)
 - Arc::CRC32Sum, [183](#)
 - Arc::MD5Sum, [386](#)
 - Arc::MessageAttributes, [392](#)
 - Arc::SoftwareRequirement, [529](#)
- add_problematic_delivery_service
 - DataStaging::DTR, [225](#)
- AddBartender
 - Arc::UserConfig, [581](#)
- AddCertExtObj
 - Arc::Credential, [188](#)
- AddChain
 - Arc::VOMSTrustList, [609](#)
- AddConsumer
 - Arc::DelegationContainerSOAP, [212](#)
- addConsumer
 - Arc::ComputingServiceRetriever, [160](#)
 - Arc::EntityRetriever, [256](#)
- addDestination
 - Arc::Logger, [367](#)
- addDestinations
 - Arc::Logger, [367](#)
- addEndpoint
 - Arc::ComputingServiceRetriever, [160](#)
 - Arc::EntityRetriever, [256](#)
- addEntity
 - Arc::ComputingServiceRetriever, [160](#)
 - Arc::ComputingServiceUniq, [163](#)
 - Arc::EntityConsumer, [252](#)
 - Arc::EntityContainer, [253](#)
 - Arc::EntityRetriever, [256](#)
 - Arc::ExecutionTargetSorter, [277](#)
 - Arc::JobSupervisor, [354](#)
- AddExtension
 - Arc::Credential, [188](#)
- AddHTTPOption
 - Arc::URL, [570](#)

- AddJob
 - Arc::JobSupervisor, 354
- AddNew
 - Arc::XMLNodeContainer, 669
- AddOption
 - Arc::OptionParser, 415, 416
 - Arc::URL, 570
- addPolicy
 - ArcSec::Evaluator, 266
 - ArcSec::Policy, 455
- AddRegex
 - Arc::VOMSTrustList, 609
- addRegistrar
 - Arc::InfoRegisterContainer, 300
- AddRejectDiscoveryURLs
 - Arc::UserConfig, 581
- addRequestItem
 - ArcSec::Request, 474
- Address
 - Arc::WSAEndpointReference, 613
- AddSecHandler
 - Arc::ClientSOAP, 146
 - Arc::MCC, 377
 - Arc::Service, 507
- addService
 - Arc::InfoRegisterContainer, 300
 - Arc::InfoRegistrar, 302
- AddSignatureTemplate
 - Arc::XMLSecNode, 670
- addVOMSAC
 - Arc, 61
- adler32
 - Arc::ChecksumAny, 136
- AfterFork
 - Arc::Run, 484
- allocated
 - Arc::PayloadRawBuf, 425
- allocated_
 - Arc::WSRF, 619
- Annotation
 - Arc::JobIdentificationType, 339
- ANY
 - Arc::ConfigEndpoint, 168
- ApplicationEnvironments
 - Arc::ComputingManagerType, 157
- ApplyToConfig
 - Arc::UserConfig, 581
- approveCSR
 - Arc::OAuthConsumer, 411
 - Arc::SAML2SSOHTTIClient, 488
- Arc, 43
 - addVOMSAC, 61
 - ASCTime, 61
 - AttrConstIter, 58
 - AttrIter, 58
 - AttrMap, 59
 - BUSY_ERROR, 60
 - CanonicalDir, 61
 - COMPUTING, 60
 - ContentFromPayload, 61
 - CreateThreadFunction, 61
 - createVOMSAC, 62
 - CredentialLogger, 71
 - DEBUG, 59
 - DebugFormat, 59
 - DirCreate, 62
 - DirDelete, 62
 - EmptyFormat, 59
 - EnvLockAcquire, 62
 - EnvLockRelease, 62
 - EnvLockUnwrap, 62
 - EnvLockUnwrapComplete, 63
 - EnvLockWrap, 63
 - EpochTime, 61
 - ERROR, 60
 - escape_char, 59
 - escape_hex, 59
 - escape_octal, 59
 - escape_chars, 63
 - escape_type, 59
 - FATAL, 60
 - FileCopy, 63
 - FileCreate, 63
 - FileDelete, 63
 - FileLink, 63
 - FileRead, 64
 - FileReadLink, 64
 - FileStat, 64
 - final_xmlsec, 64
 - GENERIC_ERROR, 60
 - get_cert_str, 64
 - get_key_from_certfile, 64
 - get_key_from_certstr, 64
 - get_key_from_keyfile, 64
 - get_key_from_keystr, 65
 - get_node, 65
 - get_plugin_instance, 59
 - getCredentialProperty, 65
 - GUID, 65
 - INDEX, 60
 - INFO, 60
 - init_xmlsec, 65
 - inttostr, 65, 66
 - ISOTime, 61
 - istring_to_level, 66
 - load_key_from_certfile, 66
 - load_key_from_certstr, 67
 - load_key_from_keyfile, 67

- load_trusted_cert_file, 67
- load_trusted_cert_str, 67
- load_trusted_certs, 67
- LogFormat, 59
- LogLevel, 59
- LongFormat, 59
- MDSTime, 61
- OpenSSLInit, 67
- operator<<, 67
- parseVOMSAC, 67, 68
- PARSING_ERROR, 60
- passphrase_callback, 69
- PeriodBase, 60
- PeriodDays, 60
- PeriodHours, 60
- PeriodMicroseconds, 60
- PeriodMilliseconds, 60
- PeriodMinutes, 60
- PeriodNanoseconds, 60
- PeriodSeconds, 60
- PeriodWeeks, 60
- plugins_table_name, 71
- PROTOCOL_RECOGNIZED_ERROR, 60
- RFC1123Time, 61
- ServiceType, 60
- SESSION_CLOSE, 60
- ShortFormat, 59
- STATUS_OK, 60
- StatusKind, 60
- string, 69
- strtobool, 69
- strtoint, 69, 70
- TimeFormat, 60
- TmpDirCreate, 70
- TmpFileCreate, 70
- UNKNOWN_SERVICE_ERROR, 60
- uri_encode, 70
- UserTime, 61
- UTCTime, 61
- VERBOSE, 59
- Version, 71
- VOMSACSeqEncode, 70, 71
- VOMSDecode, 71
- VOMSEncode, 71
- WARNING, 60
- WSAFault, 61
- WSAFaultAssign, 71
- WSAFaultExtract, 71
- WSAFaultInvalidAddressingHeader, 61
- WSAFaultUnknown, 61
- ARC Compute Library (libarccompute), 1, 35
- ARC data staging (libarcdatastaging), 39
- Arc::Adler32Sum, 94
 - add, 94
 - end, 94
 - print, 94
 - scan, 95
 - start, 95
- Arc::AdminDomainAttributes, 96
- Arc::AdminDomainType, 97
- Arc::ApplicationEnvironment, 100
- Arc::ApplicationType, 101
 - Error, 101
 - Executable, 101
 - Input, 101
 - LogDir, 101
 - Output, 101
 - PostExecutable, 101
 - PreExecutable, 101
 - RemoteLogging, 102
- Arc::ArcLocation, 103
 - GetPlugins, 103
 - Init, 103
- Arc::ARCPolicyHandlerConfig, 105
- Arc::ArcVersion, 106
- Arc::AttributeIterator, 109
 - AttributeIterator, 109
 - current_, 111
 - end_, 111
 - hasMore, 110
 - key, 110
 - MessageAttributes, 111
 - operator*, 110
 - operator++, 110
 - operator->, 110
- Arc::AutoPointer, 118
- Arc::Base64, 119
- Arc::BaseConfig, 120
 - MakeConfig, 120
- Arc::Broker, 123
- Arc::BrokerPlugin, 124
- Arc::BrokerPluginArgument, 125
- Arc::BrokerPluginLoader, 126
- Arc::BrokerPluginTestACCCControl, 127
- Arc::CertEnvLocker, 129
- Arc::ChainContext, 131
 - operator PluginsFactory *, 131
- Arc::Checksum, 132
 - add, 132
 - end, 133
 - print, 133
 - scan, 133
 - start, 133
- Arc::ChecksumAny, 135
 - add, 136
 - adler32, 136
 - cksum, 136
 - end, 136

- FileChecksum, 136
- md5, 136
- none, 136
- print, 136
- scan, 137
- start, 137
- type, 135
- undefined, 136
- unknown, 136
- Arc::CIStrngValue, 138
 - CIStrngValue, 138
 - equal, 138
 - operator bool, 138
- Arc::ClassLoader, 140
- Arc::ClassLoaderPluginArgument, 141
- Arc::ClientHTTP, 142
 - GetEntry, 142
 - Load, 142
- Arc::ClientHTTPAttributes, 143
- Arc::ClientHTTPwithSAML2SSO, 144
 - ClientHTTPwithSAML2SSO, 144
 - process, 144
- Arc::ClientInterface, 145
 - Load, 145
- Arc::ClientSOAP, 146
 - AddSecHandler, 146
 - ClientSOAP, 146
 - GetEntry, 146
 - Load, 147
 - process, 147
- Arc::ClientSOAPwithSAML2SSO, 148
 - ClientSOAPwithSAML2SSO, 148
 - process, 148
- Arc::ClientTCP, 149
 - GetEntry, 149
 - Load, 149
- Arc::ClientX509Delegation, 150
 - acquireDelegation, 150
 - ClientX509Delegation, 150
 - createDelegation, 150
- Arc::ComputingEndpointAttributes, 154
- Arc::ComputingEndpointType, 155
- Arc::ComputingManagerAttributes, 156
- Arc::ComputingManagerType, 157
 - ApplicationEnvironments, 157
- Arc::ComputingServiceAttributes, 158
- Arc::ComputingServiceRetriever, 159
 - addConsumer, 160
 - addEndpoint, 160
 - addEntity, 160
 - ComputingServiceRetriever, 159
 - getAllStatuses, 160
 - GetExecutionTargets, 161
 - removeConsumer, 161
 - wait, 161
- Arc::ComputingServiceType, 162
- Arc::ComputingServiceUniq, 163
 - addEntity, 163
- Arc::ComputingShareAttributes, 164
 - FreeSlotsWithDuration, 164
 - MaxDiskSpace, 164
 - MaxMainMemory, 164
 - MaxVirtualMemory, 164
 - Name, 164
- Arc::ComputingShareType, 165
- Arc::Config, 166
 - Config, 166
 - print, 167
- Arc::ConfigEndpoint, 168
 - ANY, 168
 - COMPUTINGINFO, 168
 - ConfigEndpoint, 169
 - REGISTRY, 168
 - RequestedSubmissionInterfaceName, 169
 - Type, 168
- Arc::ConfusaCertHandler, 170
 - ConfusaCertHandler, 170
 - createCertRequest, 170
 - getCertRequestB64, 170
- Arc::ConfusaParserUtils, 171
 - destroy_doc, 171
 - evaluate_path, 171
 - extract_body_information, 171
 - get_doc, 171
 - handle_redirect_step, 171
 - urlencode, 172
 - urlencode_params, 172
- Arc::CountedPointer, 173
- Arc::Counter, 174
 - ~Counter, 176
 - cancel, 176
 - changeExcess, 176
 - changeLimit, 176
 - Counter, 176
 - extend, 177
 - getCounterTicket, 177
 - getcurrentTime, 177
 - getExcess, 178
 - getExpirationReminder, 178
 - getExpiryTime, 178
 - getLimit, 178
 - getValue, 179
 - IDType, 176
 - reserve, 179
 - setExcess, 179
 - setLimit, 179
- Arc::CounterTicket, 181
 - cancel, 181

- CounterTicket, 181
- extend, 181
- isValid, 182
- Arc::CRC32Sum, 183
 - add, 183
 - end, 183
 - print, 183
 - scan, 184
 - start, 184
- Arc::Credential, 185
 - AddCertExtObj, 188
 - AddExtension, 188
 - Credential, 186, 187
 - GenerateEECRequest, 188
 - GenerateRequest, 189
 - GetCAName, 189
 - GetCert, 189
 - GetCertNumofChain, 189
 - GetCertReq, 189
 - GetDN, 189
 - GetEndTime, 189
 - GetExtension, 189
 - getFormat_BIO, 190
 - GetIdentityName, 190
 - GetIssuerName, 190
 - GetLifeTime, 190
 - GetPrivKey, 190
 - GetProxyPolicy, 190
 - GetPubKey, 190
 - GetStartTime, 190
 - GetType, 190
 - GetVerification, 190
 - InitProxyCertInfo, 190
 - InquireRequest, 191
 - IsCredentialsValid, 191
 - IsValid, 191
 - LogError, 191
 - OutputCertificate, 191
 - OutputCertificateChain, 191
 - OutputPrivatekey, 192
 - OutputPublickey, 192
 - SelfSignEECRequest, 192
 - SetLifeTime, 192
 - SetProxyPolicy, 192
 - SetStartTime, 192
 - SignEECRequest, 192, 193
 - SignRequest, 193
 - STACK_OF, 193
- Arc::CredentialError, 194
 - CredentialError, 194
- Arc::CredentialStore, 195
- Arc::Database, 196
 - connect, 196
 - enable_ssl, 196
- Arc::DataStagingType, 205
- Arc::DelegationConsumer, 208
 - Acquire, 208
 - Backup, 209
 - DelegationConsumer, 208
 - Generate, 209
 - ID, 209
 - LogError, 209
 - Request, 209
 - Restore, 209
- Arc::DelegationConsumerSOAP, 210
 - DelegateCredentialsInit, 210
 - DelegatedToken, 210
 - DelegationConsumerSOAP, 210
 - UpdateCredentials, 211
- Arc::DelegationContainerSOAP, 212
 - AddConsumer, 212
 - CheckConsumers, 212
 - context_lock_, 214
 - DelegateCredentialsInit, 213
 - DelegatedToken, 213
 - FindConsumer, 213
 - GetFailure, 213
 - MatchNamespace, 213
 - max_duration_, 214
 - max_size_, 214
 - max_usage_, 214
 - QueryConsumer, 213
 - ReleaseConsumer, 213
 - RemoveConsumer, 213
 - TouchConsumer, 213
 - UpdateCredentials, 214
- Arc::DelegationProvider, 215
 - Delegate, 215
 - DelegationProvider, 215
- Arc::DelegationProviderSOAP, 216
 - DelegateCredentialsInit, 217
 - DelegatedToken, 217
 - DelegationProviderSOAP, 216
 - ID, 217
 - UpdateCredentials, 217
- Arc::DiskSpaceRequirementType, 220
 - CacheDiskSpace, 220
 - DiskSpace, 220
 - SessionDiskSpace, 220
- Arc::DNListHandlerConfig, 222
- Arc::Endpoint, 237
 - Capability, 240
 - CapabilityEnum, 238
 - Endpoint, 238, 239
 - getServiceName, 239
 - GetStringForCapability, 239
 - HasCapability, 239, 240
 - HealthState, 240

- HealthStateInfo, 240
- InterfaceName, 240
- operator<, 240
- operator=, 240
- QualityLevel, 240
- RequestedSubmissionInterfaceName, 241
- ServiceID, 241
- str, 240
- URLString, 241
- Arc::EndpointQueryingStatus, 242
 - EndpointQueryingStatus, 243
 - EndpointQueryingStatusType, 242
 - FAILED, 243
 - getDescription, 243
 - getStatus, 243
 - NOINFORETURNED, 243
 - NOPLUGIN, 243
 - operator bool, 243
 - operator=, 244
 - operator==, 244
 - STARTED, 243
 - str, 244, 245
 - SUCCESSFUL, 243
 - SUSPENDED_NOTREQUIRED, 242
 - UNKNOWN, 242
- Arc::EndpointQueryOptions, 246
 - EndpointQueryOptions, 246
- Arc::EndpointQueryOptions< Endpoint >, 247
 - EndpointQueryOptions, 247
- Arc::EndpointStatusMap, 248
- Arc::EndpointSubmissionStatus, 249
 - EndpointSubmissionStatus, 249
 - EndpointSubmissionStatusType, 249
 - getDescription, 249
 - getStatus, 249
 - operator bool, 250
 - operator=, 250
 - operator==, 250, 251
 - str, 251
- Arc::EntityConsumer, 252
 - addEntity, 252
- Arc::EntityContainer, 253
 - addEntity, 253
- Arc::EntityRetriever, 254
 - addConsumer, 256
 - addEndpoint, 256
 - addEntity, 256
 - clearEndpointStatuses, 257
 - EntityRetriever, 256
 - getAllStatuses, 257
 - getServicesWithStatus, 257
 - getStatusOfEndpoint, 257
 - isDone, 258
 - removeConsumer, 258
 - removeEndpoint, 258
 - setStatusOfEndpoint, 258
 - wait, 259
- Arc::EntityRetriever::Common, 153
- Arc::EntityRetriever::Result, 482
- Arc::EntityRetriever::ThreadArg, 553
- Arc::EntityRetrieverPlugin, 260
- Arc::EntityRetrieverPluginLoader, 261
- Arc::EnvLockWrapper, 262
 - EnvLockWrapper, 262
- Arc::ExecutableType, 272
 - Argument, 272
 - Path, 272
 - SuccessExitCode, 272
- Arc::ExecutionEnvironmentAttributes, 273
 - OperatingSystem, 273
- Arc::ExecutionEnvironmentType, 274
- Arc::ExecutionTarget, 275
 - ExecutionTarget, 275
 - operator<=, 276
 - RegisterJobSubmission, 276
- Arc::ExecutionTargetSorter, 277
 - addEntity, 277
- Arc::ExpirationReminder, 278
 - getExpiryTime, 278
 - getReservationID, 278
 - operator<, 278
- Arc::FileAccess, 279
 - fa_copy, 280
 - fa_mkdirp, 280
 - fa_mkstemp, 280
 - fa_setuid, 280
- Arc::FileAccess::header_t, 294
- Arc::FileAccessContainer, 281
 - Acquire, 281
 - Release, 281
- Arc::FileLock, 282
 - acquire, 283
 - check, 283
 - FileLock, 282
 - release, 283
- Arc::FinderLoader, 285
- Arc::GlobusResult, 289
- Arc::GLUE2, 290
 - ParseExecutionTargets, 290
- Arc::GLUE2Entity, 291
- Arc::GSSCredencial, 292
- Arc::HakaClient, 293
 - processConsent, 293
 - processIdP2Confusa, 293
 - processIdPLogin, 293
- Arc::HTTPClientInfo, 295
- Arc::InfoCache, 296
 - InfoCache, 296

- Arc::InfoCacheInterface, 297
 - Get, 297
- Arc::InfoFilter, 298
 - Filter, 298
 - InfoFilter, 298
- Arc::InfoRegister, 299
- Arc::InfoRegisterContainer, 300
 - addRegistrar, 300
 - addService, 300
 - removeService, 300
- Arc::InfoRegisters, 301
 - InfoRegisters, 301
- Arc::InfoRegistrar, 302
 - addService, 302
 - registration, 302
- Arc::InformationContainer, 303
 - Acquire, 303
 - Assign, 303
 - doc_, 304
 - Get, 303
 - InformationContainer, 303
- Arc::InformationInterface, 305
 - Get, 305
 - InformationInterface, 305
 - lock_, 306
- Arc::InformationRequest, 307
 - InformationRequest, 307
 - SOAP, 307
- Arc::InformationResponse, 308
 - InformationResponse, 308
 - Result, 308
- Arc::IniConfig, 309
- Arc::initializeCredentialsType, 310
 - initializeType, 310
 - NotTryCredentials, 310
 - RequireCredentials, 310
 - SkipCANotTryCredentials, 310
 - SkipCARequireCredentials, 310
 - SkipCATryCredentials, 310
 - SkipCredentials, 310
 - TryCredentials, 310
- Arc::InputFileType, 311
 - Checksum, 311
- Arc::InterruptGuard, 313
- Arc::IntraProcessCounter, 314
 - ~IntraProcessCounter, 314
 - cancel, 315
 - changeExcess, 315
 - changeLimit, 315
 - extend, 315
 - getExcess, 316
 - getLimit, 316
 - getValue, 316
 - IntraProcessCounter, 314
 - reserve, 316
 - setExcess, 317
 - setLimit, 317
- Arc::ISIS_description, 318
- Arc::IString, 319
- Arc::Job, 321
 - Job, 321
 - JobDescription, 324
 - JobDescriptionDocument, 324
 - operator=, 321
 - ReadJobIDsFromFile, 322
 - SaveToStream, 322
 - SetFromXML, 322
 - ToXML, 322
 - WriteJobIDsToFile, 323
 - WriteJobIDToFile, 323
- Arc::JobControllerPlugin, 325
- Arc::JobControllerPluginArgument, 326
- Arc::JobControllerPluginLoader, 327
 - ~JobControllerPluginLoader, 327
 - JobControllerPluginLoader, 327
 - load, 327
- Arc::JobControllerPluginTestACCCControl, 329
- Arc::JobDescription, 330
 - GetSourceLanguage, 330
 - OtherAttributes, 332
 - Parse, 330
 - Prepare, 331
 - SaveToStream, 331
 - UnParse, 332
- Arc::JobDescriptionParserPlugin, 333
- Arc::JobDescriptionParserPluginLoader, 334
 - ~JobDescriptionParserPluginLoader, 334
 - GetJobDescriptionParserPlugins, 334
 - JobDescriptionParserPluginLoader, 334
 - load, 335
- Arc::JobDescriptionParserPluginLoader::iterator, 320
- Arc::JobDescriptionParserPluginResult, 336
- Arc::JobDescriptionParserPluginTestACCCControl, 337
- Arc::JobDescriptionResult, 338
- Arc::JobIdentificationType, 339
 - ActivityOldID, 339
 - Annotation, 339
 - Description, 339
 - JobName, 339
 - Type, 339
- Arc::JobInformationStorage, 341
 - Clean, 342
 - GetName, 342
 - JobInformationStorage, 341
 - Read, 342
 - ReadAll, 343

- Remove, 343
- Write, 343, 344
- Arc::JobInformationStorageXML, 345
 - Clean, 345
 - Read, 345
 - ReadAll, 346
 - Remove, 346
 - Write, 347
- Arc::JobListRetrieverPlugin, 348
- Arc::JobListRetrieverPluginTESTControl, 349
- Arc::JobState, 350
 - GetGeneralState, 350
 - GetSpecificState, 350
 - IsFinished, 351
 - operator(), 351
- Arc::JobStateTEST, 352
- Arc::JobSupervisor, 353
 - addEntity, 354
 - AddJob, 354
 - Cancel, 354
 - Clean, 355
 - JobSupervisor, 353
 - Migrate, 355
 - Renew, 356
 - Resubmit, 356
 - Resume, 357
 - Retrieve, 358
 - Update, 358
- Arc::Loader, 360
 - ~Loader, 360
 - factory_, 360
 - Loader, 360
- Arc::LocationAttributes, 361
- Arc::LocationType, 362
- Arc::LogDestination, 363
- Arc::LogFile, 364
 - log, 364
 - LogFile, 364
 - setBackups, 365
 - setMaxSize, 365
 - setReopen, 365
- Arc::Logger, 366
 - addDestination, 367
 - addDestinations, 367
 - getDestinations, 367
 - getRootLogger, 367
 - Logger, 367
 - msg, 368
 - setDestinations, 368
 - setThreadContext, 368
 - setThreshold, 368
 - setThresholdForDomain, 368, 369
- Arc::LoggerFormat, 370
- Arc::LogMessage, 371
 - getLevel, 372
 - Logger, 372
 - LogMessage, 371
 - operator<<, 372
 - setIdentifier, 372
- Arc::LogStream, 373
 - log, 373
 - LogStream, 373
- Arc::MCC, 376
 - AddSecHandler, 377
 - logger, 378
 - MCC, 377
 - Next, 377
 - next_, 378
 - next_lock_, 378
 - process, 377
 - ProcessSecHandlers, 377
 - sechandlers_, 378
 - Unlink, 377
- Arc::MCC_Status, 379
 - getExplanation, 379
 - getKind, 379
 - getOrigin, 380
 - isOk, 380
 - MCC_Status, 379
 - operator bool, 380
 - operator std::string, 380
- Arc::MCCConfig, 381
 - MakeConfig, 381
- Arc::MCCInterface, 382
 - process, 382
- Arc::MCCLoader, 383
 - ~MCCLoader, 383
 - MCCLoader, 383
- Arc::MCCPluginArgument, 385
- Arc::MD5Sum, 386
 - add, 386
 - end, 386
 - print, 386
 - scan, 387
 - start, 387
- Arc::Message, 388
 - ~Message, 389
 - Attributes, 389
 - Auth, 389
 - AuthContext, 389
 - Context, 389
 - Message, 389
 - operator=, 390
 - Payload, 390
- Arc::MessageAttributes, 391
 - add, 392
 - attributes_, 393
 - count, 392

- get, 392
- getAll, 392
- MessageAttributes, 391
- remove, 392
- removeAll, 393
- set, 393
- Arc::MessageAuth, 394
 - Export, 394
 - Filter, 394
- Arc::MessageAuthContext, 395
- Arc::MessageContext, 396
 - Add, 396
- Arc::MessageContextElement, 397
- Arc::MessagePayload, 398
- Arc::ModuleDesc, 400
- Arc::ModuleManager, 401
 - find, 402
 - findLocation, 402
 - load, 402
 - makePersistent, 402
 - ModuleManager, 402
 - reload, 402
 - setCfg, 402
 - unload, 402, 403
 - unuse, 403
 - use, 403
- Arc::MultiSecAttr, 404
 - Export, 404
 - operator bool, 404
- Arc::MySQLDatabase, 405
 - connect, 405
 - enable_ssl, 405
- Arc::MySQLQuery, 407
 - execute, 407
 - get_array, 407
 - get_row, 407, 408
 - get_row_field, 408
- Arc::NotificationType, 409
- Arc::NS, 410
 - NS, 410
- Arc::OAuthConsumer, 411
 - approveCSR, 411
 - OAuthConsumer, 411
 - parseDN, 411
 - processLogin, 412
 - pushCSR, 412
 - storeCert, 412
- Arc::OpenIdpClient, 413
 - processConsent, 413
 - processIdP2Confusa, 413
 - processIdPLogin, 413
- Arc::OptIn, 414
- Arc::OptionParser, 415
 - AddOption, 415, 416
 - OptionParser, 415
 - Parse, 416
- Arc::OutputFileType, 419
- Arc::ParallelEnvironmentType, 420
- Arc::PathIterator, 421
 - PathIterator, 421
- Arc::PayloadRaw, 422
 - ~PayloadRaw, 422
 - Buffer, 422
 - BufferPos, 423
 - BufferSize, 423
 - Content, 423
 - Insert, 423
 - PayloadRaw, 422
 - Size, 423
 - Truncate, 423
- Arc::PayloadRawBuf, 425
 - allocated, 425
 - length, 425
 - size, 425
- Arc::PayloadRawInterface, 426
 - Buffer, 426
 - BufferPos, 426
 - BufferSize, 426
 - Content, 427
 - Insert, 427
 - Size, 427
 - Truncate, 427
- Arc::PayloadSOAP, 428
 - PayloadSOAP, 428
- Arc::PayloadStream, 429
 - ~PayloadStream, 429
 - Get, 430
 - handle_, 431
 - Limit, 430
 - operator bool, 430
 - PayloadStream, 429
 - Pos, 430
 - Put, 430
 - seekable_, 431
 - Size, 430
 - Timeout, 430, 431
- Arc::PayloadStreamInterface, 432
 - Get, 432, 433
 - Limit, 433
 - operator bool, 433
 - Pos, 433
 - Put, 433, 434
 - Size, 434
 - Timeout, 434
- Arc::PayloadWSRF, 435
 - PayloadWSRF, 435
- Arc::Period, 439
- Arc::Plexer, 443

- ~Plexer, 443
- logger, 444
- Next, 444
- Plexer, 443
- process, 444
- Arc::PlexerEntry, 445
- Arc::Plugin, 446
- Arc::PluginArgument, 447
 - get_factory, 447
 - get_module, 447
- Arc::PluginDesc, 449
- Arc::PluginDescriptor, 450
- Arc::PluginsFactory, 451
 - FilterByKind, 452
 - get_instance, 452
 - load, 452
 - PluginsFactory, 452
 - report, 452
 - scan, 452
 - TryLoad, 452
- Arc::PluginsFactory::modules_t::diterator, 221
- Arc::PluginsFactory::modules_t::miterator, 399
- Arc::Profile, 463
- Arc::Query, 466
 - execute, 466
 - get_array, 466
 - get_row, 467
 - get_row_field, 467
 - Query, 466
- Arc::Range, 468
- Arc::Register_Info_Type, 469
- Arc::RegisteredService, 470
 - RegisteredService, 470
- Arc::RegularExpression, 471
 - match, 471
- Arc::RemoteLoggingType, 472
 - Location, 472
 - optional, 472
 - ServiceType, 472
- Arc::ResourcesType, 478
- Arc::Run, 483
 - Abandon, 484
 - AfterFork, 484
 - AssignStderr, 484
 - AssignStdin, 484
 - AssignStdout, 484
 - Kill, 484
 - ReadStderr, 484
 - ReadStdout, 484
 - Result, 485
 - Start, 485
 - Wait, 485
 - WriteStdin, 485
- Arc::SAML2LoginClient, 487
 - findSimpleSAMLInstallation, 487
 - processLogin, 487
 - SAML2LoginClient, 487
- Arc::SAML2SSOHTTPClient, 488
 - approveCSR, 488
 - parseDN, 488
 - processConsent, 488
 - processIdP2Confusa, 488
 - processIdPLogin, 489
 - processLogin, 489
 - pushCSR, 489
 - storeCert, 489
- Arc::SAMLToken, 490
 - ~SAMLToken, 491
 - Authenticate, 492
 - operator bool, 492
 - SAMLToken, 491
 - SAMLVersion, 491
- Arc::ScalableTime, 493
- Arc::ScalableTime< int >, 494
- Arc::SecAttr, 497
 - Export, 497
 - get, 498
 - getAll, 498
 - Import, 498
 - operator bool, 498
 - operator==, 498
- Arc::SecAttrFormat, 499
- Arc::SecAttrValue, 500
 - operator bool, 500
 - operator==, 500
- Arc::SecHandlerConfig, 503
- Arc::Service, 506
 - AddSecHandler, 507
 - getID, 507
 - logger, 508
 - operator bool, 507
 - ProcessSecHandlers, 507
 - RegistrationCollector, 507
 - sechandlers_, 508
 - Service, 507
 - valid, 508
- Arc::ServiceEndpointRetrieverPlugin, 509
- Arc::ServiceEndpointRetrieverPluginTESTControl, 510
- Arc::ServicePluginArgument, 511
- Arc::SharedMutex, 512
 - forceReset, 512
- Arc::SimpleCondition, 513
 - broadcast, 513
 - forceReset, 513
 - signal, 513
 - signal_nonblock, 513
 - wait, 513

- wait_nonblock, 513
- Arc::SimpleCounter, 515
 - dec, 515
 - forceReset, 515
 - get, 515
 - inc, 515
 - set, 515
 - wait, 516
- Arc::SlotRequirementType, 517
- Arc::SOAPMessage, 518
 - ~SOAPMessage, 518
 - Attributes, 518
 - Payload, 518, 519
 - SOAPMessage, 518
- Arc::Software, 520
 - ComparisonOperator, 521
 - ComparisonOperatorEnum, 521
 - convert, 522
 - empty, 523
 - EQUAL, 521
 - getFamily, 523
 - getName, 523
 - getVersion, 523
 - GREATER THAN, 521
 - GREATER THAN OR EQUAL, 521
 - LESS THAN, 521
 - LESS THAN OR EQUAL, 521
 - NOTEQUAL, 521
 - operator std::string, 523
 - operator<, 524
 - operator<=, 524
 - operator<=, 524
 - operator>, 525
 - operator>=, 525
 - operator(), 524
 - operator==, 525
 - Software, 522
 - toString, 526
 - VERSIONTOKENS, 526
- Arc::SoftwareRequirement, 528
 - add, 529
 - clear, 530
 - empty, 530
 - getComparisonOperatorList, 530
 - getSoftwareList, 530
 - isResolved, 531
 - isSatisfied, 531, 532
 - operator=, 532
 - selectSoftware, 532, 533
 - SoftwareRequirement, 528, 529
- Arc::SourceType, 537
- Arc::SubmissionStatus, 541
- Arc::Submitter, 542
- Arc::SubmitterPlugin, 543
 - Migrate, 543
 - Submit, 543
- Arc::SubmitterPluginArgument, 545
- Arc::SubmitterPluginLoader, 546
 - ~SubmitterPluginLoader, 546
 - load, 546
 - SubmitterPluginLoader, 546
- Arc::SubmitterPluginTestACCCControl, 548
- Arc::TargetInformationRetrieverPlugin, 549
- Arc::TargetInformationRetrieverPluginTESTControl, 550
- Arc::TargetType, 551
- Arc::TCPsec, 552
- Arc::ThreadDataItem, 554
 - Attach, 554
 - Dup, 554
 - Get, 555
 - ThreadDataItem, 554
- Arc::ThreadedPointer, 556
 - Release, 556
 - WaitInRange, 556
 - WaitOutOfRange, 556
- Arc::ThreadInitializer, 558
 - forceReset, 558
 - waitExit, 558
- Arc::ThreadRegistry, 559
 - forceReset, 559
 - WaitForExit, 559
 - WaitOrCancel, 559
- Arc::Time, 560
- Arc::TimedMutex, 563
 - forceReset, 563
 - lock, 563
- Arc::URL, 567
 - AddHTTPOption, 570
 - AddOption, 570
 - CommonLocOption, 570
 - FullPathURIEncoded, 570
 - HTTPOption, 571
 - MetaDataOption, 571
 - Option, 571
 - OptionString, 571
 - RemoveHTTPOption, 571
 - RemoveMetaDataOption, 571
 - RemoveOption, 572
 - URIDecode, 572
 - URIEncode, 572
- Arc::URLLocation, 573
- Arc::User, 574
 - check_file_access, 574
 - SwitchUser, 574
 - User, 574
- Arc::UserConfig, 576
 - AddBartender, 581

- AddRejectDiscoveryURLs, 581
- ApplyToConfig, 581
- ARCUSERDIRECTORY, 602
- Bartender, 581, 582
- Broker, 582, 583
- CACertificatePath, 583
- CACertificatesDirectory, 584
- CertificateLifeTime, 584, 585
- CertificatePath, 585
- ClearRejectDiscoveryURLs, 586
- CredentialsFound, 586
- DEFAULT_BROKER, 602
- DEFAULT_TIMEOUT, 602
- DEFAULTCONFIG, 602
- EXAMPLECONFIG, 602
- GetDefaultServices, 586
- GetService, 586
- GetServices, 587
- GetServicesInGroup, 587
- GetUser, 587
- IdPName, 588
- InfoInterface, 588
- InitializeCredentials, 589
- JobDownloadDirectory, 590
- JobListFile, 591
- KeyPassword, 591, 592
- KeyPath, 592
- KeySize, 593
- LoadConfigurationFile, 593
- operator bool, 594
- OverlayFile, 594
- Password, 595
- ProxyPath, 595, 596
- RejectDiscoveryURLs, 596
- RejectManagementURLs, 596
- SaveToFile, 596
- SetUser, 597
- SLCS, 597
- StoreDirectory, 597, 598
- SubmissionInterface, 598
- SYSCONFIG, 602
- SYSCONFIGARCLOC, 602
- Timeout, 599
- UserConfig, 579, 580
- UserName, 599
- UtilsDirPath, 600
- Verbosity, 600, 601
- VOMSESPath, 601
- Arc::UsernameToken, 604
 - Authenticate, 605
 - operator bool, 605
 - PasswordType, 604
 - Username, 605
 - UsernameToken, 604, 605
- Arc::UserSwitch, 606
- Arc::VOMSACInfo, 607
- Arc::VOMSTrustList, 608
 - AddChain, 609
 - AddRegex, 609
 - VOMSTrustList, 608
- Arc::WatchdogChannel, 610
 - WatchdogChannel, 610
- Arc::WatchdogListener, 611
 - Listen, 611
- Arc::WSAEndpointReference, 612
 - ~WSAEndpointReference, 612
 - Address, 613
 - hasAddress, 613
 - MetaData, 613
 - operator XMLNode, 613
 - operator=, 613
 - ReferenceParameters, 613
 - WSAEndpointReference, 612
- Arc::WSAHeader, 614
 - Action, 615
 - Check, 615
 - FaultTo, 615
 - From, 615
 - hasAction, 615
 - hasMessageID, 615
 - hasRelatesTo, 615
 - hasRelationshipType, 615
 - hasTo, 616
 - header_allocated_, 617
 - MessageID, 616
 - NewReferenceParameter, 616
 - operator XMLNode, 616
 - ReferenceParameter, 616
 - RelatesTo, 616
 - RelationshipType, 616
 - ReplyTo, 617
 - To, 617
 - WSAHeader, 615
- Arc::WSRF, 618
 - allocated_, 619
 - operator bool, 619
 - set_namespaces, 619
 - SOAP, 619
 - valid_, 619
 - WSRF, 619
- Arc::WSRFBBaseFault, 620
 - set_namespaces, 620
 - WSRFBBaseFault, 620
- Arc::WSRFResourceUnavailableFault, 621
- Arc::WSRFResourceUnknownFault, 622
- Arc::WSRP, 623
 - set_namespaces, 624
 - WSRP, 624

- Arc::WSRPDeleteResourceProperties, 625
- Arc::WSRPDeleteResourcePropertiesRequest, 626
- Arc::WSRPDeleteResourcePropertiesRequestFailedFault, 627
- Arc::WSRPDeleteResourcePropertiesResponse, 628
- Arc::WSRPFault, 629
 - WSRPFault, 629
- Arc::WSRPGetMultipleResourcePropertiesRequest, 630
- Arc::WSRPGetMultipleResourcePropertiesResponse, 631
- Arc::WSRPGetResourcePropertyDocumentRequest, 632
- Arc::WSRPGetResourcePropertyDocumentResponse, 633
- Arc::WSRPGetResourcePropertyRequest, 634
- Arc::WSRPGetResourcePropertyResponse, 635
- Arc::WSRPInsertResourceProperties, 636
- Arc::WSRPInsertResourcePropertiesRequest, 637
- Arc::WSRPInsertResourcePropertiesRequestFailedFault, 638
- Arc::WSRPInsertResourcePropertiesResponse, 639
- Arc::WSRPInvalidModificationFault, 640
- Arc::WSRPInvalidResourcePropertyQNameFault, 641
- Arc::WSRPModifyResourceProperties, 642
- Arc::WSRPPutResourcePropertyDocumentRequest, 643
- Arc::WSRPPutResourcePropertyDocumentResponse, 644
- Arc::WSRPQueryResourcePropertiesRequest, 645
- Arc::WSRPQueryResourcePropertiesResponse, 646
- Arc::WSRPResourcePropertyChangeFailure, 647
 - WSRPResourcePropertyChangeFailure, 647
- Arc::WSRPSetResourcePropertiesRequest, 648
- Arc::WSRPSetResourcePropertiesResponse, 649
- Arc::WSRPSetResourcePropertyRequestFailedFault, 650
- Arc::WSRPUnableToModifyResourcePropertyFault, 651
- Arc::WSRPUnableToPutResourcePropertyDocumentFault, 652
- Arc::WSRPUpdateResourceProperties, 653
- Arc::WSRPUpdateResourcePropertiesRequest, 654
- Arc::WSRPUpdateResourcePropertiesRequestFailedFault, 655
- Arc::WSRPUpdateResourcePropertiesResponse, 656
- Arc::X509Token, 658
 - ~X509Token, 659
 - Authenticate, 659
 - operator bool, 659
 - X509Token, 658
 - X509TokenType, 658
- Arc::XmlContainer, 660
- Arc::XmlDatabase, 661
- Arc::XMLNode, 662
 - ~XMLNode, 664
 - Child, 665
 - Destroy, 665
 - Exchange, 665
 - GetXML, 665
 - is_owner_, 668
 - LogError, 665
 - Move, 665
 - Namespaces, 666
 - New, 666
 - NewChild, 666
 - operator++, 666
 - operator--, 666
 - operator=, 666
 - Path, 667
 - Prefix, 667
 - Swap, 667
 - Validate, 668
 - XMLNode, 664
 - XPathLookup, 668
- Arc::XMLNodeContainer, 669
 - Add, 669
 - AddNew, 669
 - XMLNodeContainer, 669
- Arc::XMLSecNode, 670
 - AddSignatureTemplate, 670
 - DecryptNode, 670
 - EncryptNode, 671
 - SignNode, 671
 - VerifyNode, 671
 - XMLSecNode, 670
- ArcCredential, 73
 - CERT_TYPE_CA, 73
 - CERT_TYPE_EEC, 73
 - CERT_TYPE_GSI_2_LIMITED_PROXY, 74
 - CERT_TYPE_GSI_2_PROXY, 74
 - CERT_TYPE_GSI_3_IMPERSONATION_PROXY, 74
 - CERT_TYPE_GSI_3_INDEPENDENT_PROXY, 74
 - CERT_TYPE_GSI_3_LIMITED_PROXY, 74
 - CERT_TYPE_GSI_3_RESTRICTED_PROXY, 74
 - CERT_TYPE_RFC_ANYLANGUAGE_PROXY, 74
 - CERT_TYPE_RFC_IMPERSONATION_PROXY, 74
 - CERT_TYPE_RFC_INDEPENDENT_PROXY, 74

- CERT_TYPE_RFC_LIMITED_PROXY, [74](#)
- CERT_TYPE_RFC_RESTRICTED_PROXY, [74](#)
- certType, [73](#)
- ArcCredential::ACACI, [77](#)
- ArcCredential::ACATTHOLDER, [78](#)
- ArcCredential::ACATTR, [79](#)
- ArcCredential::ACATTRIBUTE, [80](#)
- ArcCredential::ACC, [81](#)
- ArcCredential::ACCERTS, [82](#)
- ArcCredential::ACDIGEST, [83](#)
- ArcCredential::ACFORM, [84](#)
- ArcCredential::ACFULLATTRIBUTES, [85](#)
- ArcCredential::ACHOLDER, [86](#)
- ArcCredential::ACIETFATTR, [87](#)
- ArcCredential::ACINFO, [88](#)
- ArcCredential::ACIS, [89](#)
- ArcCredential::ACSEQ, [90](#)
- ArcCredential::ACTARGET, [91](#)
- ArcCredential::ACTARGETS, [92](#)
- ArcCredential::ACVAL, [93](#)
- ArcCredential::cert_verify_context, [128](#)
- ArcCredential::PROXYCERTINFO_st, [464](#)
- ArcCredential::PROXYPOLICY_st, [465](#)
- ArcSec::AlgFactory, [98](#)
 - createAlg, [98](#)
- ArcSec::AnyURIAttribute, [99](#)
 - encode, [99](#)
 - getId, [99](#)
 - getType, [99](#)
- ArcSec::ArcPeriod, [104](#)
- ArcSec::Attr, [107](#)
- ArcSec::AttributeFactory, [108](#)
- ArcSec::AttributeProxy, [112](#)
 - getAttribute, [112](#)
- ArcSec::AttributeValue, [113](#)
 - encode, [114](#)
 - equal, [114](#)
 - getId, [114](#)
 - getType, [114](#)
- ArcSec::Attrs, [115](#)
- ArcSec::AuthzRequest, [116](#)
- ArcSec::AuthzRequestSection, [117](#)
- ArcSec::BooleanAttribute, [122](#)
 - encode, [122](#)
 - getId, [122](#)
 - getType, [122](#)
- ArcSec::CombiningAlg, [152](#)
 - combine, [152](#)
 - getalgId, [152](#)
- ArcSec::DateAttribute, [206](#)
 - encode, [206](#)
 - getId, [206](#)
 - getType, [206](#)
- ArcSec::DateTimeAttribute, [207](#)
 - encode, [207](#)
 - getId, [207](#)
 - getType, [207](#)
- ArcSec::DenyOverridesCombiningAlg, [218](#)
 - combine, [218](#)
 - getalgId, [218](#)
- ArcSec::DurationAttribute, [236](#)
 - encode, [236](#)
 - getId, [236](#)
 - getType, [236](#)
- ArcSec::EqualFunction, [263](#)
 - evaluate, [263](#)
 - getFunctionName, [263](#)
- ArcSec::EvalResult, [264](#)
- ArcSec::EvaluationCtx, [265](#)
 - EvaluationCtx, [265](#)
- ArcSec::Evaluator, [266](#)
 - addPolicy, [266](#)
 - evaluate, [266](#), [267](#)
 - getAlgFactory, [267](#)
 - getAttrFactory, [267](#)
 - getFnFactory, [268](#)
 - getName, [268](#)
 - setCombiningAlg, [268](#)
- ArcSec::EvaluatorContext, [269](#)
 - operator AlgFactory *, [269](#)
 - operator AttributeFactory *, [269](#)
 - operator FnFactory *, [269](#)
- ArcSec::EvaluatorLoader, [270](#)
 - getEvaluator, [270](#)
 - getPolicy, [270](#)
 - getRequest, [270](#)
- ArcSec::FnFactory, [286](#)
 - createFn, [286](#)
- ArcSec::Function, [287](#)
 - evaluate, [287](#)
- ArcSec::GenericAttribute, [288](#)
 - encode, [288](#)
 - getId, [288](#)
 - getType, [288](#)
- ArcSec::InRangeFunction, [312](#)
 - evaluate, [312](#)
- ArcSec::MatchFunction, [375](#)
 - evaluate, [375](#)
 - getFunctionName, [375](#)
- ArcSec::OrderedCombiningAlg, [418](#)
- ArcSec::PDP, [436](#)
- ArcSec::PDPConfigContext, [437](#)
- ArcSec::PDPPluginArgument, [438](#)
- ArcSec::PeriodAttribute, [440](#)
 - encode, [440](#)
 - getId, [440](#)
 - getType, [440](#)

- ArcSec::PermitOverridesCombiningAlg, 441
 - combine, 441
 - getalgId, 441
- ArcSec::Policy, 454
 - addPolicy, 455
 - eval, 455
 - getEffect, 455
 - getEvalName, 455
 - getEvalResult, 455
 - getName, 455
 - make_policy, 455
 - Policy, 454
 - setEvalResult, 455
 - setEvaluatorContext, 455
- ArcSec::PolicyParser, 458
 - parsePolicy, 458
- ArcSec::PolicyStore, 459
 - PolicyStore, 459
- ArcSec::PolicyStore::PolicyElement, 457
- ArcSec::Request, 473
 - addRequestItem, 474
 - getEvalName, 474
 - getName, 474
 - getRequestItems, 474
 - make_request, 474
 - Request, 473
 - setAttributeFactory, 474
 - setRequestItems, 474
- ArcSec::RequestAttribute, 475
 - duplicate, 475
 - RequestAttribute, 475
- ArcSec::RequestItem, 476
 - RequestItem, 476
- ArcSec::RequestTuple, 477
- ArcSec::Response, 479
- ArcSec::ResponseItem, 480
- ArcSec::ResponseList, 481
- ArcSec::SecHandler, 501
- ArcSec::SecHandlerConfig, 502
- ArcSec::SecHandlerPluginArgument, 504
- ArcSec::Security, 505
- ArcSec::Source, 535
 - Source, 535
- ArcSec::SourceFile, 536
- ArcSec::SourceURL, 538
- ArcSec::StringAttribute, 540
 - encode, 540
 - getId, 540
 - getType, 540
- ArcSec::TimeAttribute, 562
 - encode, 562
 - getId, 562
 - getType, 562
- ArcSec::X509NameAttribute, 657
 - encode, 657
 - getId, 657
 - getType, 657
- ARCUSERDIRECTORY
 - Arc::UserConfig, 602
- Argument
 - Arc::ExecutableType, 272
- ASCTime
 - Arc, 61
- Assign
 - Arc::InformationContainer, 303
- AssignStderr
 - Arc::Run, 484
- AssignStdin
 - Arc::Run, 484
- AssignStdout
 - Arc::Run, 484
- Attach
 - Arc::ThreadDataItem, 554
- AttrConstIter
 - Arc, 58
- AttributeIterator
 - Arc::AttributeIterator, 109
- Attributes
 - Arc::Message, 389
 - Arc::SOAPMessage, 518
- attributes_
 - Arc::MessageAttributes, 393
- AttrIter
 - Arc, 58
- AttrMap
 - Arc, 59
- Auth
 - Arc::Message, 389
- AuthContext
 - Arc::Message, 389
- Authenticate
 - Arc::SAMLToken, 492
 - Arc::UsernameToken, 605
 - Arc::X509Token, 659
- AuthN::certInfo, 130
- AuthN::PrivateKeyInfoCodec, 460
- Backup
 - Arc::DelegationConsumer, 209
- Bartender
 - Arc::UserConfig, 581, 582
- broadcast
 - Arc::SimpleCondition, 513
- Broker
 - Arc::UserConfig, 582, 583
- BrokerPlugin.h, 673
- Buffer
 - Arc::PayloadRaw, 422

- Arc::PayloadRawInterface, 426
- BufferPos
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 426
- BufferSize
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 426
- BUSY_ERROR
 - Arc, 60
- CACertificatePath
 - Arc::UserConfig, 583
- CACertificatesDirectory
 - Arc::UserConfig, 584
- CACHE_ALREADY_PRESENT
 - datastaging, 41
- CACHE_CHECKED
 - DataStaging::DTRStatus, 235
- CACHE_DOWNLOADED
 - datastaging, 41
- CACHE_ERROR
 - DataStaging::DTRErrorStatus, 230
- CACHE_LOCKED
 - datastaging, 42
- CACHE_NOT_USED
 - datastaging, 42
- CACHE_PROCESSED
 - DataStaging::DTRStatus, 235
- CACHE_SKIP
 - datastaging, 42
- CACHE_WAIT
 - DataStaging::DTRStatus, 235
- CACHEABLE
 - datastaging, 41
- CacheDiskSpace
 - Arc::DiskSpaceRequirementType, 220
- CacheState
 - datastaging, 41
- calculate_shares
 - DataStaging::TransferShares, 565
- Cancel
 - Arc::JobSupervisor, 354
- cancel
 - Arc::Counter, 176
 - Arc::CounterTicket, 181
 - Arc::IntraProcessCounter, 315
- CANCELLED
 - DataStaging::DTRStatus, 235
- CANCELLED_FINISHED
 - DataStaging::DTRStatus, 235
- CanonicalDir
 - Arc, 61
- Capability
 - Arc::Endpoint, 240
- CapabilityEnum
 - Arc::Endpoint, 238
- CERT_TYPE_CA
 - ArcCredential, 73
- CERT_TYPE_EEC
 - ArcCredential, 73
- CERT_TYPE_GSI_2_LIMITED_PROXY
 - ArcCredential, 74
- CERT_TYPE_GSI_2_PROXY
 - ArcCredential, 74
- CERT_TYPE_GSI_3_IMPERSONATION_PROXY
 - ArcCredential, 74
- CERT_TYPE_GSI_3_INDEPENDENT_PROXY
 - ArcCredential, 74
- CERT_TYPE_GSI_3_LIMITED_PROXY
 - ArcCredential, 74
- CERT_TYPE_GSI_3_RESTRICTED_PROXY
 - ArcCredential, 74
- CERT_TYPE_RFC_ANYLANGUAGE_PROXY
 - ArcCredential, 74
- CERT_TYPE_RFC_IMPERSONATION_PROXY
 - ArcCredential, 74
- CERT_TYPE_RFC_INDEPENDENT_PROXY
 - ArcCredential, 74
- CERT_TYPE_RFC_LIMITED_PROXY
 - ArcCredential, 74
- CERT_TYPE_RFC_RESTRICTED_PROXY
 - ArcCredential, 74
- CertificateLifeTime
 - Arc::UserConfig, 584, 585
- CertificatePath
 - Arc::UserConfig, 585
- certType
 - ArcCredential, 73
- changeExcess
 - Arc::Counter, 176
 - Arc::IntraProcessCounter, 315
- changeLimit
 - Arc::Counter, 176
 - Arc::IntraProcessCounter, 315
- Check
 - Arc::WSAHeader, 615
- check
 - Arc::FileLock, 283
- CHECK_CACHE
 - DataStaging::DTRStatus, 235
- check_file_access
 - Arc::User, 574
- CheckComm
 - DataStaging::DataDeliveryComm, 201
- CheckConsumers
 - Arc::DelegationContainerSOAP, 212
- CHECKING_CACHE

- DataStaging::DTRStatus, 235
- Checksum
 - Arc::InputFileType, 311
- Child
 - Arc::XMLNode, 665
- CIStrngValue
 - Arc::CIStrngValue, 138
- cksum
 - Arc::ChecksumAny, 136
- Classes for controlling output of compute test plug-ins, 38
- Clean
 - Arc::JobInformationStorage, 342
 - Arc::JobInformationStorageXML, 345
 - Arc::JobSupervisor, 355
- clear
 - Arc::SoftwareRequirement, 530
- clearEndpointStatuses
 - Arc::EntityRetriever, 257
- ClearRejectDiscoveryURLs
 - Arc::UserConfig, 586
- ClientHTTPwithSAML2SSO
 - Arc::ClientHTTPwithSAML2SSO, 144
- ClientSOAP
 - Arc::ClientSOAP, 146
- ClientSOAPwithSAML2SSO
 - Arc::ClientSOAPwithSAML2SSO, 148
- ClientX509Delegation
 - Arc::ClientX509Delegation, 150
- combine
 - ArcSec::CombiningAlg, 152
 - ArcSec::DenyOverridesCombiningAlg, 218
 - ArcSec::PermitOverridesCombiningAlg, 441
- CommClosed
 - DataStaging::DataDeliveryComm, 200
- CommExited
 - DataStaging::DataDeliveryComm, 200
- CommFailed
 - DataStaging::DataDeliveryComm, 200
- CommInit
 - DataStaging::DataDeliveryComm, 200
- CommNoError
 - DataStaging::DataDeliveryComm, 200
- CommonLocOption
 - Arc::URL, 570
- CommStatusType
 - DataStaging::DataDeliveryComm, 200
- CommTimeout
 - DataStaging::DataDeliveryComm, 200
- ComparisonOperator
 - Arc::Software, 521
- ComparisonOperatorEnum
 - Arc::Software, 521
- compute
 - JobListRetriever, 2, 36
 - ServiceEndpointRetriever, 2, 36
 - TargetInformationRetriever, 2, 36
- COMPUTING
 - Arc, 60
- COMPUTINGINFO
 - Arc::ConfigEndpoint, 168
- ComputingServiceRetriever
 - Arc::ComputingServiceRetriever, 159
- Config
 - Arc::Config, 166
- ConfigEndpoint
 - Arc::ConfigEndpoint, 169
- ConfusaCertHandler
 - Arc::ConfusaCertHandler, 170
- connect
 - Arc::Database, 196
 - Arc::MySQLDatabase, 405
- Content
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 427
- ContentFromPayload
 - Arc, 61
- Context
 - Arc::Message, 389
- context_lock_
 - Arc::DelegationContainerSOAP, 214
- convert
 - Arc::Software, 522
- count
 - Arc::MessageAttributes, 392
- Counter
 - Arc::Counter, 176
- CounterTicket
 - Arc::CounterTicket, 181
- createAlg
 - ArcSec::AlgFactory, 98
- createCertRequest
 - Arc::ConfusaCertHandler, 170
- createDelegation
 - Arc::ClientX509Delegation, 150
- createFn
 - ArcSec::FnFactory, 286
- CreateThreadFunction
 - Arc, 61
- createVOMSAC
 - Arc, 62
- Credential
 - Arc::Credential, 186, 187
- CredentialError
 - Arc::CredentialError, 194
- CredentialLogger
 - Arc, 71
- CredentialsFound

- Arc::UserConfig, 586
- current_
 - Arc::AttributeIterator, 111
- DataDeliveryComm
 - DataStaging::DataDeliveryComm, 200
- DataStaging, 75
- datastaging
 - CACHE_ALREADY_PRESENT, 41
 - CACHE_DOWNLOADED, 41
 - CACHE_LOCKED, 42
 - CACHE_NOT_USED, 42
 - CACHE_SKIP, 42
 - CACHEABLE, 41
 - CacheState, 41
 - DELIVERY, 42
 - DTR_ptr, 41
 - DTRLogger, 41
 - GENERATOR, 42
 - INITIATED, 42
 - NON_CACHEABLE, 41
 - POST_PROCESSOR, 42
 - PRE_PROCESSOR, 42
 - ProcessState, 42
 - RUNNING, 42
 - SCHEDULER, 42
 - StagingProcesses, 42
 - STOPPED, 42
 - TO_STOP, 42
- DataStaging::DataDelivery, 198
 - receiveDTR, 198
- DataStaging::DataDeliveryComm, 199
 - CheckComm, 201
 - CommClosed, 200
 - CommExited, 200
 - CommFailed, 200
 - CommInit, 200
 - CommNoError, 200
 - CommStatusType, 200
 - CommTimeout, 200
 - DataDeliveryComm, 200
 - PullStatus, 201
- DataStaging::DataDeliveryComm::Status, 539
- DataStaging::DataDeliveryCommHandler, 202
- DataStaging::DataDeliveryLocalComm, 203
- DataStaging::DataDeliveryRemoteComm, 204
- DataStaging::DTR, 223
 - add_problematic_delivery_service, 225
 - DTR, 225
 - registerCallback, 225
 - reset, 225
 - set_error_status, 225
- DataStaging::DTRCacheParameters, 227
- DataStaging::DTRCallback, 228
 - receiveDTR, 228
- DataStaging::DTRErrorStatus, 229
 - CACHE_ERROR, 230
 - DTRErrorLocation, 229
 - DTRErrorStatus, 230
 - DTRErrorStatusType, 230
 - ERROR_DESTINATION, 229
 - ERROR_SOURCE, 229
 - ERROR_TRANSFER, 229
 - ERROR_UNKNOWN, 230
 - INTERNAL_LOGIC_ERROR, 230
 - INTERNAL_PROCESS_ERROR, 230
 - LOCAL_FILE_ERROR, 230
 - NO_ERROR_LOCATION, 229
 - NONE_ERROR, 230
 - PERMANENT_REMOTE_ERROR, 230
 - SELF_REPLICATION_ERROR, 230
 - STAGING_TIMEOUT_ERROR, 230
 - TEMPORARY_REMOTE_ERROR, 230
 - TRANSFER_SPEED_ERROR, 230
- DataStaging::DTRLList, 231
 - dumpState, 231
 - filter_dtrs_by_job, 231
 - filter_dtrs_by_next_receiver, 232
 - filter_dtrs_by_owner, 232
 - filter_dtrs_by_status, 232
 - filter_dtrs_by_statuses, 232
 - filter_pending_dtrs, 233
- DataStaging::DTRStatus, 234
 - CACHE_CHECKED, 235
 - CACHE_PROCESSED, 235
 - CACHE_WAIT, 235
 - CANCELLED, 235
 - CANCELLED_FINISHED, 235
 - CHECK_CACHE, 235
 - CHECKING_CACHE, 235
 - DONE, 235
 - DTRStatusType, 235
 - ERROR, 235
 - NEW, 235
 - NULL_STATE, 235
 - PRE_CLEAN, 235
 - PRE_CLEARED, 235
 - PRE_CLEANNING, 235
 - PROCESS_CACHE, 235
 - PROCESSING_CACHE, 235
 - QUERY_REPLICA, 235
 - QUERYING_REPLICA, 235
 - REGISTER_REPLICA, 235
 - REGISTERING_REPLICA, 235
 - RELEASE_REQUEST, 235
 - RELEASING_REQUEST, 235
 - REPLICA_QUERIED, 235
 - REPLICA_REGISTERED, 235

- REQUEST_RELEASED, 235
- RESOLVE, 235
- RESOLVED, 235
- RESOLVING, 235
- STAGE_PREPARE, 235
- STAGED_PREPARED, 235
- STAGING_PREPARING, 235
- STAGING_PREPARING_WAIT, 235
- TRANSFER, 235
- TRANSFERRED, 235
- TRANSFERRING, 235
- TRANSFERRING_CANCEL, 235
- DataStaging::Processor, 461
 - receiveDTR, 461
 - start, 461
 - stop, 462
- DataStaging::Scheduler, 495
 - getInstance, 495
 - receiveDTR, 496
 - SetPreferredPattern, 496
 - start, 496
 - stop, 496
- DataStaging::TransferParameters, 564
 - max_inactivity_time, 564
 - min_average_bandwidth, 564
 - min_current_bandwidth, 564
- DataStaging::TransferShares, 565
 - calculate_shares, 565
 - decrease_number_of_slots, 565
- DataStaging::TransferSharesConf, 566
 - GROUP, 566
 - NONE, 566
 - ROLE, 566
 - ShareType, 566
 - USER, 566
 - VO, 566
- DEBUG
 - Arc, 59
- DebugFormat
 - Arc, 59
- dec
 - Arc::SimpleCounter, 515
- decrease_number_of_slots
 - DataStaging::TransferShares, 565
- DecryptNode
 - Arc::XMLSecNode, 670
- DEFAULT_BROKER
 - Arc::UserConfig, 602
- DEFAULT_TIMEOUT
 - Arc::UserConfig, 602
- DEFAULTCONFIG
 - Arc::UserConfig, 602
- Delegate
 - Arc::DelegationProvider, 215
- DelegateCredentialsInit
 - Arc::DelegationConsumerSOAP, 210
 - Arc::DelegationContainerSOAP, 213
 - Arc::DelegationProviderSOAP, 217
- DelegatedToken
 - Arc::DelegationConsumerSOAP, 210
 - Arc::DelegationContainerSOAP, 213
 - Arc::DelegationProviderSOAP, 217
- DelegationConsumer
 - Arc::DelegationConsumer, 208
- DelegationConsumerSOAP
 - Arc::DelegationConsumerSOAP, 210
- DelegationProvider
 - Arc::DelegationProvider, 215
- DelegationProviderSOAP
 - Arc::DelegationProviderSOAP, 216
- DELIVERY
 - datastaging, 42
- Description
 - Arc::JobIdentificationType, 339
- Destroy
 - Arc::XMLNode, 665
- destroy_doc
 - Arc::ConfusaParserUtils, 171
- DirCreate
 - Arc, 62
- DirDelete
 - Arc, 62
- DiskSpace
 - Arc::DiskSpaceRequirementType, 220
- doc_
 - Arc::InformationContainer, 304
- DONE
 - DataStaging::DTRStatus, 235
- DTR
 - DataStaging::DTR, 225
- DTR_ptr
 - datastaging, 41
- DTRErrorLocation
 - DataStaging::DTRErrorStatus, 229
- DTRErrorStatus
 - DataStaging::DTRErrorStatus, 230
- DTRErrorStatusType
 - DataStaging::DTRErrorStatus, 230
- DTRLogger
 - datastaging, 41
- DTRStatusType
 - DataStaging::DTRStatus, 235
- dumpState
 - DataStaging::DTRLList, 231
- Dup
 - Arc::ThreadDataItem, 554
- duplicate
 - ArcSec::RequestAttribute, 475

- empty
 - Arc::Software, 523
 - Arc::SoftwareRequirement, 530
- EmptyFormat
 - Arc, 59
- enable_ssl
 - Arc::Database, 196
 - Arc::MySQLDatabase, 405
- encode
 - ArcSec::AnyURIAttribute, 99
 - ArcSec::AttributeValue, 114
 - ArcSec::BooleanAttribute, 122
 - ArcSec::DateAttribute, 206
 - ArcSec::DateTimeAttribute, 207
 - ArcSec::DurationAttribute, 236
 - ArcSec::GenericAttribute, 288
 - ArcSec::PeriodAttribute, 440
 - ArcSec::StringAttribute, 540
 - ArcSec::TimeAttribute, 562
 - ArcSec::X500NameAttribute, 657
- EncryptNode
 - Arc::XMLSecNode, 671
- end
 - Arc::Adler32Sum, 94
 - Arc::Checksum, 133
 - Arc::ChecksumAny, 136
 - Arc::CRC32Sum, 183
 - Arc::MD5Sum, 386
- end_
 - Arc::AttributeIterator, 111
- Endpoint
 - Arc::Endpoint, 238, 239
- EndpointQueryingStatus
 - Arc::EndpointQueryingStatus, 243
- EndpointQueryingStatusType
 - Arc::EndpointQueryingStatus, 242
- EndpointQueryOptions
 - Arc::EndpointQueryOptions, 246
 - Arc::EndpointQueryOptions< Endpoint >, 247
- EndpointSubmissionStatus
 - Arc::EndpointSubmissionStatus, 249
- EndpointSubmissionStatusType
 - Arc::EndpointSubmissionStatus, 249
- EntityRetriever
 - Arc::EntityRetriever, 256
- EntityRetrieverPlugin.h, 674
- EnvLockAcquire
 - Arc, 62
- EnvLockRelease
 - Arc, 62
- EnvLockUnwrap
 - Arc, 62
- EnvLockUnwrapComplete
 - Arc, 63
- EnvLockWrap
 - Arc, 63
- EnvLockWrapper
 - Arc::EnvLockWrapper, 262
- EpochTime
 - Arc, 61
- EQUAL
 - Arc::Software, 521
- equal
 - Arc::CStringValue, 138
 - ArcSec::AttributeValue, 114
- ERROR
 - Arc, 60
 - DataStaging::DTRStatus, 235
- Error
 - Arc::ApplicationType, 101
- ERROR_DESTINATION
 - DataStaging::DTRErrorStatus, 229
- ERROR_SOURCE
 - DataStaging::DTRErrorStatus, 229
- ERROR_TRANSFER
 - DataStaging::DTRErrorStatus, 229
- ERROR_UNKNOWN
 - DataStaging::DTRErrorStatus, 230
- escape_char
 - Arc, 59
- escape_hex
 - Arc, 59
- escape_octal
 - Arc, 59
- escape_chars
 - Arc, 63
- escape_type
 - Arc, 59
- eval
 - ArcSec::Policy, 455
- evaluate
 - ArcSec::EqualFunction, 263
 - ArcSec::Evaluator, 266, 267
 - ArcSec::Function, 287
 - ArcSec::InRangeFunction, 312
 - ArcSec::MatchFunction, 375
- evaluate_path
 - Arc::ConfusaParserUtils, 171
- EvaluationCtx
 - ArcSec::EvaluationCtx, 265
- EXAMPLECONFIG
 - Arc::UserConfig, 602
- Exchange
 - Arc::XMLNode, 665
- Executable
 - Arc::ApplicationType, 101
- execute

- Arc::MySQLQuery, 407
- Arc::Query, 466
- ExecutionTarget
 - Arc::ExecutionTarget, 275
- ExecutionTarget.h, 675
- Export
 - Arc::MessageAuth, 394
 - Arc::MultiSecAttr, 404
 - Arc::SecAttr, 497
- extend
 - Arc::Counter, 177
 - Arc::CounterTicket, 181
 - Arc::IntraProcessCounter, 315
- extract_body_information
 - Arc::ConfusaParserUtils, 171
- fa_copy
 - Arc::FileAccess, 280
- fa_mkdirp
 - Arc::FileAccess, 280
- fa_mkstemp
 - Arc::FileAccess, 280
- fa_setuid
 - Arc::FileAccess, 280
- factory_
 - Arc::Loader, 360
- FAILED
 - Arc::EndpointQueryingStatus, 243
- FATAL
 - Arc, 60
- FaultTo
 - Arc::WSAHeader, 615
- FileChecksum
 - Arc::ChecksumAny, 136
- FileCopy
 - Arc, 63
- FileCreate
 - Arc, 63
- FileDelete
 - Arc, 63
- FileLink
 - Arc, 63
- FileLock
 - Arc::FileLock, 282
- FileRead
 - Arc, 64
- FileReadLink
 - Arc, 64
- FileStat
 - Arc, 64
- Filter
 - Arc::InfoFilter, 298
 - Arc::MessageAuth, 394
- filter_dtrs_by_job
 - DataStaging::DTRLList, 231
- filter_dtrs_by_next_receiver
 - DataStaging::DTRLList, 232
- filter_dtrs_by_owner
 - DataStaging::DTRLList, 232
- filter_dtrs_by_status
 - DataStaging::DTRLList, 232
- filter_dtrs_by_statuses
 - DataStaging::DTRLList, 232
- filter_pending_dtrs
 - DataStaging::DTRLList, 233
- FilterByKind
 - Arc::PluginsFactory, 452
- final_xmlsec
 - Arc, 64
- find
 - Arc::ModuleManager, 402
- FindConsumer
 - Arc::DelegationContainerSOAP, 213
- findLocation
 - Arc::ModuleManager, 402
- findSimpleSAMLInstallation
 - Arc::SAML2LoginClient, 487
- forceReset
 - Arc::SharedMutex, 512
 - Arc::SimpleCondition, 513
 - Arc::SimpleCounter, 515
 - Arc::ThreadInitializer, 558
 - Arc::ThreadRegistry, 559
 - Arc::TimedMutex, 563
- FreeSlotsWithDuration
 - Arc::ComputingShareAttributes, 164
- From
 - Arc::WSAHeader, 615
- FullPathURIEncoded
 - Arc::URL, 570
- Generate
 - Arc::DelegationConsumer, 209
- GenerateEECRequest
 - Arc::Credential, 188
- GenerateRequest
 - Arc::Credential, 189
- GENERATOR
 - datastaging, 42
- GENERIC_ERROR
 - Arc, 60
- Get
 - Arc::InfoCacheInterface, 297
 - Arc::InformationContainer, 303
 - Arc::InformationInterface, 305
 - Arc::PayloadStream, 430
 - Arc::PayloadStreamInterface, 432, 433
 - Arc::ThreadDataItem, 555

- get
 - Arc::MessageAttributes, 392
 - Arc::SecAttr, 498
 - Arc::SimpleCounter, 515
- get_array
 - Arc::MySQLQuery, 407
 - Arc::Query, 466
- get_cert_str
 - Arc, 64
- get_doc
 - Arc::ConfusaParserUtils, 171
- get_factory
 - Arc::PluginArgument, 447
- get_instance
 - Arc::PluginsFactory, 452
- get_key_from_certfile
 - Arc, 64
- get_key_from_certstr
 - Arc, 64
- get_key_from_keyfile
 - Arc, 64
- get_key_from_keyst
 - Arc, 65
- get_module
 - Arc::PluginArgument, 447
- get_node
 - Arc, 65
- get_plugin_instance
 - Arc, 59
- get_row
 - Arc::MySQLQuery, 407, 408
 - Arc::Query, 467
- get_row_field
 - Arc::MySQLQuery, 408
 - Arc::Query, 467
- getAlgFactory
 - ArcSec::Evaluator, 267
- getalgId
 - ArcSec::CombiningAlg, 152
 - ArcSec::DenyOverridesCombiningAlg, 218
 - ArcSec::PermitOverridesCombiningAlg, 441
- getAll
 - Arc::MessageAttributes, 392
 - Arc::SecAttr, 498
- getAllStatuses
 - Arc::ComputingServiceRetriever, 160
 - Arc::EntityRetriever, 257
- getAttrFactory
 - ArcSec::Evaluator, 267
- getAttribute
 - ArcSec::AttributeProxy, 112
- GetCAName
 - Arc::Credential, 189
- GetCert
 - Arc::Credential, 189
- GetCertNumofChain
 - Arc::Credential, 189
- GetCertReq
 - Arc::Credential, 189
- getCertRequestB64
 - Arc::ConfusaCertHandler, 170
- getComparisonOperatorList
 - Arc::SoftwareRequirement, 530
- getCounterTicket
 - Arc::Counter, 177
- getCredentialProperty
 - Arc, 65
- getCurrentTime
 - Arc::Counter, 177
- GetDefaultServices
 - Arc::UserConfig, 586
- getDescription
 - Arc::EndpointQueryingStatus, 243
 - Arc::EndpointSubmissionStatus, 249
- getDestinations
 - Arc::Logger, 367
- GetDN
 - Arc::Credential, 189
- getEffect
 - ArcSec::Policy, 455
- GetEndTime
 - Arc::Credential, 189
- GetEntry
 - Arc::ClientHTTP, 142
 - Arc::ClientSOAP, 146
 - Arc::ClientTCP, 149
- getEvalName
 - ArcSec::Policy, 455
 - ArcSec::Request, 474
- getEvalResult
 - ArcSec::Policy, 455
- getEvaluator
 - ArcSec::EvaluatorLoader, 270
- getExcess
 - Arc::Counter, 178
 - Arc::IntraProcessCounter, 316
- GetExecutionTargets
 - Arc::ComputingServiceRetriever, 161
- getExpirationReminder
 - Arc::Counter, 178
- getExpiryTime
 - Arc::Counter, 178
 - Arc::ExpirationReminder, 278
- getExplanation
 - Arc::MCC_Status, 379
- GetExtension
 - Arc::Credential, 189
- GetFailure

- Arc::DelegationContainerSOAP, 213
- getFamily
 - Arc::Software, 523
- getFnFactory
 - ArcSec::Evaluator, 268
- getFormat_BIO
 - Arc::Credential, 190
- getFunctionName
 - ArcSec::EqualFunction, 263
 - ArcSec::MatchFunction, 375
- GetGeneralState
 - Arc::JobState, 350
- getID
 - Arc::Service, 507
- getId
 - ArcSec::AnyURIAttribute, 99
 - ArcSec::AttributeValue, 114
 - ArcSec::BooleanAttribute, 122
 - ArcSec::DateAttribute, 206
 - ArcSec::DateTimeAttribute, 207
 - ArcSec::DurationAttribute, 236
 - ArcSec::GenericAttribute, 288
 - ArcSec::PeriodAttribute, 440
 - ArcSec::StringAttribute, 540
 - ArcSec::TimeAttribute, 562
 - ArcSec::X500NameAttribute, 657
- GetIdentityName
 - Arc::Credential, 190
- getInstance
 - DataStaging::Scheduler, 495
- GetIssuerName
 - Arc::Credential, 190
- GetJobDescriptionParserPlugins
 - Arc::JobDescriptionParserPluginLoader, 334
- getKind
 - Arc::MCC_Status, 379
- getLevel
 - Arc::LogMessage, 372
- GetLifeTime
 - Arc::Credential, 190
- getLimit
 - Arc::Counter, 178
 - Arc::IntraProcessCounter, 316
- GetName
 - Arc::JobInformationStorage, 342
- getName
 - Arc::Software, 523
 - ArcSec::Evaluator, 268
 - ArcSec::Policy, 455
 - ArcSec::Request, 474
- getOrigin
 - Arc::MCC_Status, 380
- GetPlugins
 - Arc::ArcLocation, 103
- getPolicy
 - ArcSec::EvaluatorLoader, 270
- GetPrivKey
 - Arc::Credential, 190
- GetProxyPolicy
 - Arc::Credential, 190
- GetPubKey
 - Arc::Credential, 190
- getRequest
 - ArcSec::EvaluatorLoader, 270
- getRequestItems
 - ArcSec::Request, 474
- getReservationID
 - Arc::ExpirationReminder, 278
- getRootLogger
 - Arc::Logger, 367
- GetService
 - Arc::UserConfig, 586
- getServiceName
 - Arc::Endpoint, 239
- GetServices
 - Arc::UserConfig, 587
- GetServicesInGroup
 - Arc::UserConfig, 587
- getServicesWithStatus
 - Arc::EntityRetriever, 257
- getSoftwareList
 - Arc::SoftwareRequirement, 530
- GetSourceLanguage
 - Arc::JobDescription, 330
- GetSpecificState
 - Arc::JobState, 350
- GetStartTime
 - Arc::Credential, 190
- getStatus
 - Arc::EndpointQueryingStatus, 243
 - Arc::EndpointSubmissionStatus, 249
- getStatusOfEndpoint
 - Arc::EntityRetriever, 257
- GetStringForCapability
 - Arc::Endpoint, 239
- GetType
 - Arc::Credential, 190
- getType
 - ArcSec::AnyURIAttribute, 99
 - ArcSec::AttributeValue, 114
 - ArcSec::BooleanAttribute, 122
 - ArcSec::DateAttribute, 206
 - ArcSec::DateTimeAttribute, 207
 - ArcSec::DurationAttribute, 236
 - ArcSec::GenericAttribute, 288
 - ArcSec::PeriodAttribute, 440
 - ArcSec::StringAttribute, 540
 - ArcSec::TimeAttribute, 562

- ArcSec::X500NameAttribute, 657
- GetUser
 - Arc::UserConfig, 587
- getValue
 - Arc::Counter, 179
 - Arc::IntraProcessCounter, 316
- GetVerification
 - Arc::Credential, 190
- getVersion
 - Arc::Software, 523
- GetXML
 - Arc::XMLNode, 665
- GLUE2Entity.h, 676
- GREATERTHAN
 - Arc::Software, 521
- GREATERTHANOREQUAL
 - Arc::Software, 521
- GROUP
 - DataStaging::TransferSharesConf, 566
- GUID
 - Arc, 65
- handle_
 - Arc::PayloadStream, 431
- handle_redirect_step
 - Arc::ConfusaParserUtils, 171
- hasAction
 - Arc::WSAHeader, 615
- hasAddress
 - Arc::WSAEndpointReference, 613
- HasCapability
 - Arc::Endpoint, 239, 240
- hasMessageID
 - Arc::WSAHeader, 615
- hasMore
 - Arc::AttributeIterator, 110
- hasRelatesTo
 - Arc::WSAHeader, 615
- hasRelationshipType
 - Arc::WSAHeader, 615
- hasTo
 - Arc::WSAHeader, 616
- header_allocated_
 - Arc::WSAHeader, 617
- HealthState
 - Arc::Endpoint, 240
- HealthStateInfo
 - Arc::Endpoint, 240
- HTTPOption
 - Arc::URL, 571
- ID
 - Arc::DelegationConsumer, 209
 - Arc::DelegationProviderSOAP, 217
- IdPName
 - Arc::UserConfig, 588
- IDType
 - Arc::Counter, 176
- Import
 - Arc::SecAttr, 498
- inc
 - Arc::SimpleCounter, 515
- INDEX
 - Arc, 60
- INFO
 - Arc, 60
- InfoCache
 - Arc::InfoCache, 296
- InfoFilter
 - Arc::InfoFilter, 298
- InfoInterface
 - Arc::UserConfig, 588
- InfoRegisters
 - Arc::InfoRegisters, 301
- InformationContainer
 - Arc::InformationContainer, 303
- InformationInterface
 - Arc::InformationInterface, 305
- InformationRequest
 - Arc::InformationRequest, 307
- InformationResponse
 - Arc::InformationResponse, 308
- Init
 - Arc::ArcLocation, 103
- init_xmlsec
 - Arc, 65
- InitializeCredentials
 - Arc::UserConfig, 589
- initializeType
 - Arc::initializeCredentialsType, 310
- INITIATED
 - datastaging, 42
- InitProxyCertInfo
 - Arc::Credential, 190
- Input
 - Arc::ApplicationType, 101
- InquireRequest
 - Arc::Credential, 191
- Insert
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 427
- InterfaceName
 - Arc::Endpoint, 240
- INTERNAL_LOGIC_ERROR
 - DataStaging::DTRErrorStatus, 230
- INTERNAL_PROCESS_ERROR
 - DataStaging::DTRErrorStatus, 230
- IntraProcessCounter

- Arc::IntraProcessCounter, 314
- inttostr
 - Arc, 65, 66
- is_owner_
 - Arc::XMLNode, 668
- IsCredentialsValid
 - Arc::Credential, 191
- isDone
 - Arc::EntityRetriever, 258
- IsFinished
 - Arc::JobState, 351
- isOk
 - Arc::MCC_Status, 380
- ISOTime
 - Arc, 61
- isResolved
 - Arc::SoftwareRequirement, 531
- isSatisfied
 - Arc::SoftwareRequirement, 531, 532
- istring_to_level
 - Arc, 66
- IsValid
 - Arc::Credential, 191
- isValid
 - Arc::CounterTicket, 182
- Job
 - Arc::Job, 321
- JobControllerPlugin.h, 677
- JobControllerPluginLoader
 - Arc::JobControllerPluginLoader, 327
- JobDescription
 - Arc::Job, 324
- JobDescription related classes, 34
- JobDescription.h, 678
- JobDescriptionDocument
 - Arc::Job, 324
- JobDescriptionParserPlugin.h, 679
- JobDescriptionParserPluginLoader
 - Arc::JobDescriptionParserPluginLoader, 334
- JobDownloadDirectory
 - Arc::UserConfig, 590
- JobInformationStorage
 - Arc::JobInformationStorage, 341
- JobListFile
 - Arc::UserConfig, 591
- JobListRetriever
 - compute, 2, 36
- JobName
 - Arc::JobIdentificationType, 339
- JobSupervisor
 - Arc::JobSupervisor, 353
- key
 - Arc::AttributeIterator, 110
- KeyPassword
 - Arc::UserConfig, 591, 592
- KeyPath
 - Arc::UserConfig, 592
- KeySize
 - Arc::UserConfig, 593
- Kill
 - Arc::Run, 484
- length
 - Arc::PayloadRawBuf, 425
- LESSTHAN
 - Arc::Software, 521
- LESSTHANOREQUAL
 - Arc::Software, 521
- Limit
 - Arc::PayloadStream, 430
 - Arc::PayloadStreamInterface, 433
- Listen
 - Arc::WatchdogListener, 611
- Load
 - Arc::ClientHTTP, 142
 - Arc::ClientInterface, 145
 - Arc::ClientSOAP, 147
 - Arc::ClientTCP, 149
- load
 - Arc::JobControllerPluginLoader, 327
 - Arc::JobDescriptionParserPluginLoader, 335
 - Arc::ModuleManager, 402
 - Arc::PluginsFactory, 452
 - Arc::SubmitterPluginLoader, 546
- load_key_from_certfile
 - Arc, 66
- load_key_from_certstr
 - Arc, 67
- load_key_from_keyfile
 - Arc, 67
- load_trusted_cert_file
 - Arc, 67
- load_trusted_cert_str
 - Arc, 67
- load_trusted_certs
 - Arc, 67
- LoadConfigurationFile
 - Arc::UserConfig, 593
- Loader
 - Arc::Loader, 360
- LOCAL_FILE_ERROR
 - DataStaging::DTRErrorStatus, 230
- Location
 - Arc::RemoteLoggingType, 472
- lock
 - Arc::TimedMutex, 563

- lock_
 - Arc::InformationInterface, 306
- log
 - Arc::LogFile, 364
 - Arc::LogStream, 373
- LogDir
 - Arc::ApplicationType, 101
- LogError
 - Arc::Credential, 191
 - Arc::DelegationConsumer, 209
 - Arc::XMLNode, 665
- LogFile
 - Arc::LogFile, 364
- LogFormat
 - Arc, 59
- Logger
 - Arc::Logger, 367
 - Arc::LogMessage, 372
- logger
 - Arc::MCC, 378
 - Arc::Plexer, 444
 - Arc::Service, 508
- LogLevel
 - Arc, 59
- LogMessage
 - Arc::LogMessage, 371
- LogStream
 - Arc::LogStream, 373
- LongFormat
 - Arc, 59
- make_policy
 - ArcSec::Policy, 455
- make_request
 - ArcSec::Request, 474
- MakeConfig
 - Arc::BaseConfig, 120
 - Arc::MCCConfig, 381
- makePersistent
 - Arc::ModuleManager, 402
- match
 - Arc::RegularExpression, 471
- MatchNamespace
 - Arc::DelegationContainerSOAP, 213
- max_duration_
 - Arc::DelegationContainerSOAP, 214
- max_inactivity_time
 - DataStaging::TransferParameters, 564
- max_size_
 - Arc::DelegationContainerSOAP, 214
- max_usage_
 - Arc::DelegationContainerSOAP, 214
- MaxDiskSpace
 - Arc::ComputingShareAttributes, 164
- MaxMainMemory
 - Arc::ComputingShareAttributes, 164
- MaxVirtualMemory
 - Arc::ComputingShareAttributes, 164
- MCC
 - Arc::MCC, 377
- MCC_Status
 - Arc::MCC_Status, 379
- MCCLoader
 - Arc::MCCLoader, 383
- md5
 - Arc::ChecksumAny, 136
- MDSTime
 - Arc, 61
- Message
 - Arc::Message, 389
- MessageAttributes
 - Arc::AttributeIterator, 111
 - Arc::MessageAttributes, 391
- MessageID
 - Arc::WSAHeader, 616
- MetaData
 - Arc::WSAEndpointReference, 613
- MetaDataOption
 - Arc::URL, 571
- Migrate
 - Arc::JobSupervisor, 355
 - Arc::SubmitterPlugin, 543
- min_average_bandwidth
 - DataStaging::TransferParameters, 564
- min_current_bandwidth
 - DataStaging::TransferParameters, 564
- ModuleManager
 - Arc::ModuleManager, 402
- Move
 - Arc::XMLNode, 665
- msg
 - Arc::Logger, 368
- Name
 - Arc::ComputingShareAttributes, 164
- Namespaces
 - Arc::XMLNode, 666
- NEW
 - DataStaging::DTRStatus, 235
- New
 - Arc::XMLNode, 666
- NewChild
 - Arc::XMLNode, 666
- NewReferenceParameter
 - Arc::WSAHeader, 616
- Next
 - Arc::MCC, 377
 - Arc::Plexer, 444

- next_
 - Arc::MCC, [378](#)
- next_lock_
 - Arc::MCC, [378](#)
- NO_ERROR_LOCATION
 - DataStaging::DTRErrorStatus, [229](#)
- NOINFORETURNED
 - Arc::EndpointQueryingStatus, [243](#)
- NON_CACHEABLE
 - datastaging, [41](#)
- NONE
 - DataStaging::TransferSharesConf, [566](#)
- none
 - Arc::ChecksumAny, [136](#)
- NONE_ERROR
 - DataStaging::DTRErrorStatus, [230](#)
- NOPLUGIN
 - Arc::EndpointQueryingStatus, [243](#)
- NOTEQUAL
 - Arc::Software, [521](#)
- NotTryCredentials
 - Arc::initializeCredentialsType, [310](#)
- NS
 - Arc::NS, [410](#)
- NULL_STATE
 - DataStaging::DTRStatus, [235](#)
- OAuthConsumer
 - Arc::OAuthConsumer, [411](#)
- OpenSSLInit
 - Arc, [67](#)
- OperatingSystem
 - Arc::ExecutionEnvironmentAttributes, [273](#)
- operator AlgFactory *
 - ArcSec::EvaluatorContext, [269](#)
- operator AttributeFactory *
 - ArcSec::EvaluatorContext, [269](#)
- operator bool
 - Arc::CStringValue, [138](#)
 - Arc::EndpointQueryingStatus, [243](#)
 - Arc::EndpointSubmissionStatus, [250](#)
 - Arc::MCC_Status, [380](#)
 - Arc::MultiSecAttr, [404](#)
 - Arc::PayloadStream, [430](#)
 - Arc::PayloadStreamInterface, [433](#)
 - Arc::SAMLToken, [492](#)
 - Arc::SecAttr, [498](#)
 - Arc::SecAttrValue, [500](#)
 - Arc::Service, [507](#)
 - Arc::UserConfig, [594](#)
 - Arc::UsernameToken, [605](#)
 - Arc::WSRF, [619](#)
 - Arc::X509Token, [659](#)
- operator FnFactory *
 - ArcSec::EvaluatorContext, [269](#)
- operator PluginsFactory *
 - Arc::ChainContext, [131](#)
- operator std::string
 - Arc::MCC_Status, [380](#)
 - Arc::Software, [523](#)
- operator XMLNode
 - Arc::WSAEndpointReference, [613](#)
 - Arc::WSAHeader, [616](#)
- operator <
 - Arc::Endpoint, [240](#)
 - Arc::ExpirationReminder, [278](#)
 - Arc::Software, [524](#)
- operator <<
 - Arc, [67](#)
 - Arc::ExecutionTarget, [276](#)
 - Arc::LogMessage, [372](#)
 - Arc::Software, [526](#)
- operator <=
 - Arc::Software, [524](#)
- operator >
 - Arc::Software, [525](#)
- operator >=
 - Arc::Software, [525](#)
- operator *
 - Arc::AttributeIterator, [110](#)
- operator ()
 - Arc::JobState, [351](#)
 - Arc::Software, [524](#)
- operator ++
 - Arc::AttributeIterator, [110](#)
 - Arc::XMLNode, [666](#)
- operator >
 - Arc::AttributeIterator, [110](#)
- operator --
 - Arc::XMLNode, [666](#)
- operator =
 - Arc::Endpoint, [240](#)
 - Arc::EndpointQueryingStatus, [244](#)
 - Arc::EndpointSubmissionStatus, [250](#)
 - Arc::Job, [321](#)
 - Arc::Message, [390](#)
 - Arc::SoftwareRequirement, [532](#)
 - Arc::WSAEndpointReference, [613](#)
 - Arc::XMLNode, [666](#)
- operator ==
 - Arc::EndpointQueryingStatus, [244](#)
 - Arc::EndpointSubmissionStatus, [250](#), [251](#)
 - Arc::SecAttr, [498](#)
 - Arc::SecAttrValue, [500](#)
 - Arc::Software, [525](#)
- Option
 - Arc::URL, [571](#)
- optional

- Arc::RemoteLoggingType, 472
- OptionParser
 - Arc::OptionParser, 415
- OptionString
 - Arc::URL, 571
- OtherAttributes
 - Arc::JobDescription, 332
- Output
 - Arc::ApplicationType, 101
- OutputCertificate
 - Arc::Credential, 191
- OutputCertificateChain
 - Arc::Credential, 191
- OutputPrivateKey
 - Arc::Credential, 192
- OutputPublicKey
 - Arc::Credential, 192
- OverlayFile
 - Arc::UserConfig, 594
- Parse
 - Arc::JobDescription, 330
 - Arc::OptionParser, 416
- parseDN
 - Arc::OAuthConsumer, 411
 - Arc::SAML2SSOHTTIClient, 488
- ParseExecutionTargets
 - Arc::GLUE2, 290
- parsePolicy
 - ArcSec::PolicyParser, 458
- parseVOMSAC
 - Arc, 67, 68
- PARSING_ERROR
 - Arc, 60
- passphrase_callback
 - Arc, 69
- Password
 - Arc::UserConfig, 595
- PasswordType
 - Arc::UsernameToken, 604
- Path
 - Arc::ExecutableType, 272
 - Arc::XMLNode, 667
- PathIterator
 - Arc::PathIterator, 421
- Payload
 - Arc::Message, 390
 - Arc::SOAPMessage, 518, 519
- PayloadRaw
 - Arc::PayloadRaw, 422
- PayloadSOAP
 - Arc::PayloadSOAP, 428
- PayloadStream
 - Arc::PayloadStream, 429
- PayloadWSRF
 - Arc::PayloadWSRF, 435
- PeriodBase
 - Arc, 60
- PeriodDays
 - Arc, 60
- PeriodHours
 - Arc, 60
- PeriodMicroseconds
 - Arc, 60
- PeriodMilliseconds
 - Arc, 60
- PeriodMinutes
 - Arc, 60
- PeriodNanoseconds
 - Arc, 60
- PeriodSeconds
 - Arc, 60
- PeriodWeeks
 - Arc, 60
- PERMANENT_REMOTE_ERROR
 - DataStaging::DTRErrorStatus, 230
- Plexer
 - Arc::Plexer, 443
- Plugin related classes for compute specialisations, 37
- plugins_table_name
 - Arc, 71
- PluginsFactory
 - Arc::PluginsFactory, 452
- Policy
 - ArcSec::Policy, 454
- PolicyStore
 - ArcSec::PolicyStore, 459
- Pos
 - Arc::PayloadStream, 430
 - Arc::PayloadStreamInterface, 433
- POST_PROCESSOR
 - datastaging, 42
- PostExecutable
 - Arc::ApplicationType, 101
- PRE_CLEAN
 - DataStaging::DTRStatus, 235
- PRE_CLEARED
 - DataStaging::DTRStatus, 235
- PRE_CLEANNING
 - DataStaging::DTRStatus, 235
- PRE_PROCESSOR
 - datastaging, 42
- PreExecutable
 - Arc::ApplicationType, 101
- Prefix
 - Arc::XMLNode, 667
- Prepare

- Arc::JobDescription, 331
- print
 - Arc::Adler32Sum, 94
 - Arc::Checksum, 133
 - Arc::ChecksumAny, 136
 - Arc::Config, 167
 - Arc::CRC32Sum, 183
 - Arc::MD5Sum, 386
- process
 - Arc::ClientHTTPwithSAML2SSO, 144
 - Arc::ClientSOAP, 147
 - Arc::ClientSOAPwithSAML2SSO, 148
 - Arc::MCC, 377
 - Arc::MCCInterface, 382
 - Arc::Plexer, 444
- PROCESS_CACHE
 - DataStaging::DTRStatus, 235
- processConsent
 - Arc::HakaClient, 293
 - Arc::OpenIdpClient, 413
 - Arc::SAML2SSOHTTPClient, 488
- processIdP2Confusa
 - Arc::HakaClient, 293
 - Arc::OpenIdpClient, 413
 - Arc::SAML2SSOHTTPClient, 488
- processIdPLogin
 - Arc::HakaClient, 293
 - Arc::OpenIdpClient, 413
 - Arc::SAML2SSOHTTPClient, 489
- PROCESSING_CACHE
 - DataStaging::DTRStatus, 235
- processLogin
 - Arc::OAuthConsumer, 412
 - Arc::SAML2LoginClient, 487
 - Arc::SAML2SSOHTTPClient, 489
- ProcessSecHandlers
 - Arc::MCC, 377
 - Arc::Service, 507
- ProcessState
 - datastaging, 42
- PROTOCOL_RECOGNIZED_ERROR
 - Arc, 60
- ProxyPath
 - Arc::UserConfig, 595, 596
- PullStatus
 - DataStaging::DataDeliveryComm, 201
- pushCSR
 - Arc::OAuthConsumer, 412
 - Arc::SAML2SSOHTTPClient, 489
- Put
 - Arc::PayloadStream, 430
 - Arc::PayloadStreamInterface, 433, 434
- QualityLevel
 - Arc::Endpoint, 240
- Query
 - Arc::Query, 466
- QUERY_REPLICA
 - DataStaging::DTRStatus, 235
- QueryConsumer
 - Arc::DelegationContainerSOAP, 213
- QUERYING_REPLICA
 - DataStaging::DTRStatus, 235
- Read
 - Arc::JobInformationStorage, 342
 - Arc::JobInformationStorageXML, 345
- ReadAll
 - Arc::JobInformationStorage, 343
 - Arc::JobInformationStorageXML, 346
- ReadJobIDsFromFile
 - Arc::Job, 322
- ReadStderr
 - Arc::Run, 484
- ReadStdout
 - Arc::Run, 484
- receiveDTR
 - DataStaging::DataDelivery, 198
 - DataStaging::DTRCallback, 228
 - DataStaging::Processor, 461
 - DataStaging::Scheduler, 496
- ReferenceParameter
 - Arc::WSAHeader, 616
- ReferenceParameters
 - Arc::WSAEndpointReference, 613
- REGISTER_REPLICA
 - DataStaging::DTRStatus, 235
- registerCallback
 - DataStaging::DTR, 225
- RegisteredService
 - Arc::RegisteredService, 470
- REGISTERING_REPLICA
 - DataStaging::DTRStatus, 235
- RegisterJobSubmission
 - Arc::ExecutionTarget, 276
- registration
 - Arc::InfoRegistrar, 302
- RegistrationCollector
 - Arc::Service, 507
- REGISTRY
 - Arc::ConfigEndpoint, 168
- RejectDiscoveryURLs
 - Arc::UserConfig, 596
- RejectManagementURLs
 - Arc::UserConfig, 596
- RelatesTo
 - Arc::WSAHeader, 616
- RelationshipType

- Arc::WSAHeader, 616
- Release
 - Arc::FileAccessContainer, 281
 - Arc::ThreadedPointer, 556
- release
 - Arc::FileLock, 283
- RELEASE_REQUEST
 - DataStaging::DTRStatus, 235
- ReleaseConsumer
 - Arc::DelegationContainerSOAP, 213
- RELEASING_REQUEST
 - DataStaging::DTRStatus, 235
- reload
 - Arc::ModuleManager, 402
- RemoteLogging
 - Arc::ApplicationType, 102
- Remove
 - Arc::JobInformationStorage, 343
 - Arc::JobInformationStorageXML, 346
- remove
 - Arc::MessageAttributes, 392
- removeAll
 - Arc::MessageAttributes, 393
- RemoveConsumer
 - Arc::DelegationContainerSOAP, 213
- removeConsumer
 - Arc::ComputingServiceRetriever, 161
 - Arc::EntityRetriever, 258
- removeEndpoint
 - Arc::EntityRetriever, 258
- RemoveHTTPOption
 - Arc::URL, 571
- RemoveMetaDataOption
 - Arc::URL, 571
- RemoveOption
 - Arc::URL, 572
- removeService
 - Arc::InfoRegisterContainer, 300
- Renew
 - Arc::JobSupervisor, 356
- REPLICA_QUERIED
 - DataStaging::DTRStatus, 235
- REPLICA_REGISTERED
 - DataStaging::DTRStatus, 235
- ReplyTo
 - Arc::WSAHeader, 617
- report
 - Arc::PluginsFactory, 452
- Request
 - Arc::DelegationConsumer, 209
 - ArcSec::Request, 473
- REQUEST_RELEASED
 - DataStaging::DTRStatus, 235
- RequestAttribute
 - ArcSec::RequestAttribute, 475
- RequestedSubmissionInterfaceName
 - Arc::ConfigEndpoint, 169
 - Arc::Endpoint, 241
- RequestItem
 - ArcSec::RequestItem, 476
- RequireCredentials
 - Arc::initializeCredentialsType, 310
- reserve
 - Arc::Counter, 179
 - Arc::IntraProcessCounter, 316
- reset
 - DataStaging::DTR, 225
- RESOLVE
 - DataStaging::DTRStatus, 235
- RESOLVED
 - DataStaging::DTRStatus, 235
- RESOLVING
 - DataStaging::DTRStatus, 235
- Restore
 - Arc::DelegationConsumer, 209
- Resubmit
 - Arc::JobSupervisor, 356
- Result
 - Arc::InformationResponse, 308
 - Arc::Run, 485
- Resume
 - Arc::JobSupervisor, 357
- Retrieve
 - Arc::JobSupervisor, 358
- RFC1123Time
 - Arc, 61
- ROLE
 - DataStaging::TransferSharesConf, 566
- RUNNING
 - datastaging, 42
- SAML2LoginClient
 - Arc::SAML2LoginClient, 487
- SAMLTOKEN
 - Arc::SAMLToken, 491
- SAMLVersion
 - Arc::SAMLToken, 491
- SaveToFile
 - Arc::UserConfig, 596
- SaveToStream
 - Arc::Job, 322
 - Arc::JobDescription, 331
- scan
 - Arc::Adler32Sum, 95
 - Arc::Checksum, 133
 - Arc::ChecksumAny, 137
 - Arc::CRC32Sum, 184
 - Arc::MD5Sum, 387

- Arc::PluginsFactory, 452
- SCHEDULER
 - datastaging, 42
- sechandlers_
 - Arc::MCC, 378
 - Arc::Service, 508
- seekable_
 - Arc::PayloadStream, 431
- selectSoftware
 - Arc::SoftwareRequirement, 532, 533
- SELF_REPLICATION_ERROR
 - DataStaging::DTRErrorStatus, 230
- SelfSignEECRequest
 - Arc::Credential, 192
- Service
 - Arc::Service, 507
- ServiceEndpointRetriever
 - compute, 2, 36
- ServiceID
 - Arc::Endpoint, 241
- ServiceType
 - Arc, 60
 - Arc::RemoteLoggingType, 472
- SESSION_CLOSE
 - Arc, 60
- SessionDiskSpace
 - Arc::DiskSpaceRequirementType, 220
- set
 - Arc::MessageAttributes, 393
 - Arc::SimpleCounter, 515
- set_error_status
 - DataStaging::DTR, 225
- set_namespaces
 - Arc::WSRF, 619
 - Arc::WSRFBaseFault, 620
 - Arc::WSRP, 624
- setAttributeFactory
 - ArcSec::Request, 474
- setBackups
 - Arc::LogFile, 365
- setCfg
 - Arc::ModuleManager, 402
- setCombiningAlg
 - ArcSec::Evaluator, 268
- setDestinations
 - Arc::Logger, 368
- setEvalResult
 - ArcSec::Policy, 455
- setEvaluatorContext
 - ArcSec::Policy, 455
- setExcess
 - Arc::Counter, 179
 - Arc::IntraProcessCounter, 317
- SetFromXML
 - Arc::Job, 322
- setIdIdentifier
 - Arc::LogMessage, 372
- SetLifeTime
 - Arc::Credential, 192
- setLimit
 - Arc::Counter, 179
 - Arc::IntraProcessCounter, 317
- setMaxSize
 - Arc::LogFile, 365
- SetPreferredPattern
 - DataStaging::Scheduler, 496
- SetProxyPolicy
 - Arc::Credential, 192
- setReopen
 - Arc::LogFile, 365
- setRequestItems
 - ArcSec::Request, 474
- SetStartTime
 - Arc::Credential, 192
- setStatusOfEndpoint
 - Arc::EntityRetriever, 258
- setThreadContext
 - Arc::Logger, 368
- setThreshold
 - Arc::Logger, 368
- setThresholdForDomain
 - Arc::Logger, 368, 369
- SetUser
 - Arc::UserConfig, 597
- ShareType
 - DataStaging::TransferSharesConf, 566
- ShortFormat
 - Arc, 59
- signal
 - Arc::SimpleCondition, 513
- signal_nonblock
 - Arc::SimpleCondition, 513
- SignEECRequest
 - Arc::Credential, 192, 193
- SignNode
 - Arc::XMLSecNode, 671
- SignRequest
 - Arc::Credential, 193
- Size
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 427
 - Arc::PayloadStream, 430
 - Arc::PayloadStreamInterface, 434
- size
 - Arc::PayloadRawBuf, 425
- SkipCANotTryCredentials
 - Arc::initializeCredentialsType, 310
- SkipCARequireCredentials

- Arc::initializeCredentialsType, 310
- SkipCATryCredentials
 - Arc::initializeCredentialsType, 310
- SkipCredentials
 - Arc::initializeCredentialsType, 310
- SLCS
 - Arc::UserConfig, 597
- SOAP
 - Arc::InformationRequest, 307
 - Arc::WSRF, 619
- SOAPMessage
 - Arc::SOAPMessage, 518
- Software
 - Arc::Software, 522
- Software.h, 680
- SoftwareRequirement
 - Arc::SoftwareRequirement, 528, 529
- Source
 - ArcSec::Source, 535
- STACK_OF
 - Arc::Credential, 193
- STAGE_PREPARE
 - DataStaging::DTRStatus, 235
- STAGED_PREPARED
 - DataStaging::DTRStatus, 235
- STAGING_PREPARING
 - DataStaging::DTRStatus, 235
- STAGING_PREPARING_WAIT
 - DataStaging::DTRStatus, 235
- STAGING_TIMEOUT_ERROR
 - DataStaging::DTRErrorStatus, 230
- StagingProcesses
 - datastaging, 42
- Start
 - Arc::Run, 485
- start
 - Arc::Adler32Sum, 95
 - Arc::Checksum, 133
 - Arc::ChecksumAny, 137
 - Arc::CRC32Sum, 184
 - Arc::MD5Sum, 387
 - DataStaging::Processor, 461
 - DataStaging::Scheduler, 496
- STARTED
 - Arc::EndpointQueryingStatus, 243
- STATUS_OK
 - Arc, 60
- StatusKind
 - Arc, 60
- std::list, 359
- stop
 - DataStaging::Processor, 462
 - DataStaging::Scheduler, 496
- STOPPED
 - datastaging, 42
- storeCert
 - Arc::OAuthConsumer, 412
 - Arc::SAML2SSOHTTPClient, 489
- StoreDirectory
 - Arc::UserConfig, 597, 598
- str
 - Arc::Endpoint, 240
 - Arc::EndpointQueryingStatus, 244, 245
 - Arc::EndpointSubmissionStatus, 251
- string
 - Arc, 69
- strtobool
 - Arc, 69
- strtoint
 - Arc, 69, 70
- Structures holding resource information, 33
- SubmissionInterface
 - Arc::UserConfig, 598
- Submit
 - Arc::SubmitterPlugin, 543
- SubmitterPlugin.h, 681
- SubmitterPluginLoader
 - Arc::SubmitterPluginLoader, 546
- SuccessExitCode
 - Arc::ExecutableType, 272
- SUCCESSFUL
 - Arc::EndpointQueryingStatus, 243
- SUSPENDED_NOTREQUIRED
 - Arc::EndpointQueryingStatus, 242
- Swap
 - Arc::XMLNode, 667
- SwitchUser
 - Arc::User, 574
- SYSCONFIG
 - Arc::UserConfig, 602
- SYSCONFIGARCLOC
 - Arc::UserConfig, 602
- TargetInformationRetriever
 - compute, 2, 36
- TEMPORARY_REMOTE_ERROR
 - DataStaging::DTRErrorStatus, 230
- TestACCCControl.h, 682
- ThreadDataItem
 - Arc::ThreadDataItem, 554
- TimeFormat
 - Arc, 60
- Timeout
 - Arc::PayloadStream, 430, 431
 - Arc::PayloadStreamInterface, 434
 - Arc::UserConfig, 599
- TmpDirCreate
 - Arc, 70

- TmpFileCreate
 - Arc, 70
- To
 - Arc::WSAHeader, 617
- TO_STOP
 - datastaging, 42
- toString
 - Arc::Software, 526
- TouchConsumer
 - Arc::DelegationContainerSOAP, 213
- ToXML
 - Arc::Job, 322
- TRANSFER
 - DataStaging::DTRStatus, 235
- TRANSFER_SPEED_ERROR
 - DataStaging::DTRErrorStatus, 230
- TRANSFERRED
 - DataStaging::DTRStatus, 235
- TRANSFERRING
 - DataStaging::DTRStatus, 235
- TRANSFERRING_CANCEL
 - DataStaging::DTRStatus, 235
- Truncate
 - Arc::PayloadRaw, 423
 - Arc::PayloadRawInterface, 427
- TryCredentials
 - Arc::initializeCredentialsType, 310
- TryLoad
 - Arc::PluginsFactory, 452
- Type
 - Arc::ConfigEndpoint, 168
 - Arc::JobIdentificationType, 339
- type
 - Arc::ChecksumAny, 135
- undefined
 - Arc::ChecksumAny, 136
- UNKNOWN
 - Arc::EndpointQueryingStatus, 242
- unknown
 - Arc::ChecksumAny, 136
- UNKNOWN_SERVICE_ERROR
 - Arc, 60
- Unlink
 - Arc::MCC, 377
- unload
 - Arc::ModuleManager, 402, 403
- UnParse
 - Arc::JobDescription, 332
- unuse
 - Arc::ModuleManager, 403
- Update
 - Arc::JobSupervisor, 358
- UpdateCredentials
 - Arc::DelegationConsumerSOAP, 211
 - Arc::DelegationContainerSOAP, 214
 - Arc::DelegationProviderSOAP, 217
- uri_encode
 - Arc, 70
- URIDecode
 - Arc::URL, 572
- URIEncode
 - Arc::URL, 572
- urlencode
 - Arc::ConfusaParserUtils, 172
- urlencode_params
 - Arc::ConfusaParserUtils, 172
- URLString
 - Arc::Endpoint, 241
- use
 - Arc::ModuleManager, 403
- USER
 - DataStaging::TransferSharesConf, 566
- User
 - Arc::User, 574
- UserConfig
 - Arc::UserConfig, 579, 580
- UserName
 - Arc::UserConfig, 599
- Username
 - Arc::UsernameToken, 605
- UsernameToken
 - Arc::UsernameToken, 604, 605
- UserTime
 - Arc, 61
- UTCTime
 - Arc, 61
- UtilsDirPath
 - Arc::UserConfig, 600
- valid
 - Arc::Service, 508
- valid_
 - Arc::WSRF, 619
- Validate
 - Arc::XMLNode, 668
- VERBOSE
 - Arc, 59
- Verbosity
 - Arc::UserConfig, 600, 601
- VerifyNode
 - Arc::XMLSecNode, 671
- Version
 - Arc, 71
- VERSIONTOKENS
 - Arc::Software, 526
- VO
 - DataStaging::TransferSharesConf, 566

- VOMSACSeqEncode
 - Arc, [70](#), [71](#)
- VOMSDecode
 - Arc, [71](#)
- VOMSEncode
 - Arc, [71](#)
- VOMSESPath
 - Arc::UserConfig, [601](#)
- VOMSTrustList
 - Arc::VOMSTrustList, [608](#)
- Wait
 - Arc::Run, [485](#)
- wait
 - Arc::ComputingServiceRetriever, [161](#)
 - Arc::EntityRetriever, [259](#)
 - Arc::SimpleCondition, [513](#)
 - Arc::SimpleCounter, [516](#)
- wait_nonblock
 - Arc::SimpleCondition, [513](#)
- waitExit
 - Arc::ThreadInitializer, [558](#)
- WaitForExit
 - Arc::ThreadRegistry, [559](#)
- WaitInRange
 - Arc::ThreadedPointer, [556](#)
- WaitOrCancel
 - Arc::ThreadRegistry, [559](#)
- WaitOutOfRange
 - Arc::ThreadedPointer, [556](#)
- WARNING
 - Arc, [60](#)
- WatchdogChannel
 - Arc::WatchdogChannel, [610](#)
- Write
 - Arc::JobInformationStorage, [343](#), [344](#)
 - Arc::JobInformationStorageXML, [347](#)
- WriteJobIDsToFile
 - Arc::Job, [323](#)
- WriteJobIDToFile
 - Arc::Job, [323](#)
- WriteStdin
 - Arc::Run, [485](#)
- WSAEndpointReference
 - Arc::WSAEndpointReference, [612](#)
- WSAFault
 - Arc, [61](#)
- WSAFaultAssign
 - Arc, [71](#)
- WSAFaultExtract
 - Arc, [71](#)
- WSAFaultInvalidAddressingHeader
 - Arc, [61](#)
- WSAFaultUnknown
 - Arc, [61](#)
- WSAHeader
 - Arc::WSAHeader, [615](#)
- WSRF
 - Arc::WSRF, [619](#)
- WSRFBBaseFault
 - Arc::WSRFBBaseFault, [620](#)
- WSRP
 - Arc::WSRP, [624](#)
- WSRPFault
 - Arc::WSRPFault, [629](#)
- WSRPResourcePropertyChangeFailure
 - Arc::WSRPResourcePropertyChangeFailure, [647](#)
- X509Token
 - Arc::X509Token, [658](#)
- X509TokenType
 - Arc::X509Token, [658](#)
- XMLNode
 - Arc::XMLNode, [664](#)
- XMLNodeContainer
 - Arc::XMLNodeContainer, [669](#)
- XMLSecNode
 - Arc::XMLSecNode, [670](#)
- XPathLookup
 - Arc::XMLNode, [668](#)