

# globus ftp client Reference Manual

## 7.6

Generated by Doxygen 1.4.7

Thu Nov 21 14:58:38 2013

# Contents

<b>1</b>	<b>Globus FTP Client API</b>	<b>1</b>
<b>2</b>	<b>globus ftp client Module Index</b>	<b>1</b>
<b>3</b>	<b>globus ftp client Data Structure Index</b>	<b>2</b>
<b>4</b>	<b>globus ftp client Page Index</b>	<b>2</b>
<b>5</b>	<b>globus ftp client Module Documentation</b>	<b>2</b>
<b>6</b>	<b>globus ftp client Data Structure Documentation</b>	<b>109</b>
<b>7</b>	<b>globus ftp client Page Documentation</b>	<b>110</b>

## 1 Globus FTP Client API

The Globus FTP Client library provides a convenient way of accessing files on remote FTP servers. In addition to supporting the basic FTP protocol, the FTP Client library supports several security and performance extensions to make FTP more suitable for Grid applications. These extensions are described in the Grid FTP Protocol document.

In addition to protocol support for grid applications, the FTP Client library provides a **plugin architecture** (p. 68) for installing application or grid-specific fault recovery and performance tuning algorithms within the library. Application writers may then target their code toward the FTP Client library, and by simply enabling the appropriate plugins, easily tune their application to run it on a different grid.

All applications which use the Globus FTP Client API must include the header file "globus\_ftp\_client.h" and activate the **GLOBUS\_FTP\_CLIENT\_MODULE** (p. 2).

To use the Globus FTP Client API, one must create an **FTP Client handle** (p. 7). This structure contains context information about FTP operations which are being executed, a cache of FTP control and data connections, and information about plugins which are being used. The specifics of the connection caching and plugins are found in the "**Handle Attributes** (p. 11)" section of this manual.

Once the handle is created, one may begin transferring files or doing other FTP operations by calling the functions in the "**FTP Operations** (p. 18)" section of this manual. In addition to whole-file transfers, the API supports partial file transfers, restarting transfers from a known point, and various FTP directory management commands. All FTP operations may have a set of attributes, defined in the "@ref globus\_ftp\_client\_operationattr" section, associated with them to tune various FTP parameters. The data structures and functions needed to restart a file transfer are described in the "**Restart Markers** (p. 3)" section of this manual. For operations which require the user to send to or receive data from an FTP server the must call the functions in the "@ref globus\_ftp\_client\_data" section of the manual.

## 2 globus ftp client Module Index

### 2.1 globus ftp client Modules

Here is a list of all modules:

<b>Activation</b>	<b>2</b>
-------------------	----------

<b>Restart Markers</b>	<b>3</b>
<b>Handle Management</b>	<b>7</b>
<b>Handle Attributes</b>	<b>11</b>
<b>FTP Operations</b>	<b>18</b>
<b>FTP Operation Attributes</b>	<b>41</b>
<b>Reading and Writing Data</b>	<b>59</b>
<b>Plugins</b>	<b>68</b>
<b>Debugging Plugin</b>	<b>61</b>
<b>Performance Marker Plugin</b>	<b>64</b>
<b>Restart Marker Plugin</b>	<b>96</b>
<b>Restart Plugin</b>	<b>99</b>
<b>Netlogger Throughput Plugin</b>	<b>102</b>
<b>Throughput Performance Plugin</b>	<b>105</b>

### **3 globus ftp client Data Structure Index**

#### **3.1 globus ftp client Data Structures**

Here are the data structures with brief descriptions:

<b>globus_ftp_client_restart_extended_block_t (Extended block mode restart marker )</b>	<b>109</b>
<b>globus_ftp_client_restart_marker_t (Restart marker )</b>	<b>109</b>
<b>globus_ftp_client_restart_stream_t (Stream mode restart marker )</b>	<b>110</b>

### **4 globus ftp client Page Index**

#### **4.1 globus ftp client Related Pages**

Here is a list of all related documentation pages:

<b>Bug List</b>	<b>110</b>
-----------------	------------

### **5 globus ftp client Module Documentation**

#### **5.1 Activation**

The Globus FTP Client library uses the standard module activation and deactivation API to initialize its state.

## Defines

- `#define GLOBUS_FTP_CLIENT_MODULE`

### 5.1.1 Detailed Description

The Globus FTP Client library uses the standard module activation and deactivation API to initialize its state.

Before any FTP functions are called, the module must be activated

```
globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
```

This function returns `GLOBUS_SUCCESS` if the FTP library was successfully initialized. This may be called multiple times.

To deactivate the FTP library, the following must be called

```
globus_module_deactivate(GLOBUS_FTP_CLIENT_MODULE);
```

### 5.1.2 Define Documentation

#### 5.1.2.1 `#define GLOBUS_FTP_CLIENT_MODULE`

Module descriptor.

## 5.2 Restart Markers

FTP Restart Markers.

### Data Structures

- `union globus_ftp_client_restart_marker_t`  
*Restart marker.*

### Functions

- `globus_result_t globus_ftp_client_restart_marker_init(globus_ftp_client_restart_marker_t *marker)`
- `globus_result_t globus_ftp_client_restart_marker_copy(globus_ftp_client_restart_marker_t *new_marker, globus_ftp_client_restart_marker_t *marker)`
- `globus_result_t globus_ftp_client_restart_marker_destroy(globus_ftp_client_restart_marker_t *marker)`
- `globus_result_t globus_ftp_client_restart_marker_insert_range(globus_ftp_client_restart_marker_t *marker, globus_off_t offset, globus_off_t end_offset)`
- `globus_result_t globus_ftp_client_restart_marker_set_ascii_offset(globus_ftp_client_restart_marker_t *marker, globus_off_t offset, globus_off_t ascii_offset)`
- `globus_result_t globus_ftp_client_restart_marker_set_offset(globus_ftp_client_restart_marker_t *marker, globus_off_t offset)`
- `globus_result_t globus_ftp_client_restart_marker_get_total(globus_ftp_client_restart_marker_t *marker, globus_off_t *total_bytes)`
- `globus_result_t globus_ftp_client_restart_marker_to_string(globus_ftp_client_restart_marker_t *marker, char **marker_string)`
- `globus_result_t globus_ftp_client_restart_marker_from_string(globus_ftp_client_restart_marker_t *marker, const char *marker_string)`

## 5.2.1 Detailed Description

FTP Restart Markers.

The Globus FTP Client library provides the ability to start a file transfer from a known location into the file. This is accomplished by passing a restart marker to the **globus\_ftp\_client\_get()** (p. 34), **globus\_ftp\_client\_put()** (p. 36), or **globus\_ftp\_client\_third\_party\_transfer()** (p. 37) functions.

## 5.2.2 Function Documentation

### 5.2.2.1 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_init (globus\_ftp\_client\_restart\_marker\_t \* marker)**

Initialize a restart marker.

#### Parameters:

*marker* New restart marker.

#### See also:

**globus\_ftp\_client\_restart\_marker\_t** (p. 109), **globus\_ftp\_client\_restart\_marker\_destroy()** (p. 4)

### 5.2.2.2 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_copy (globus\_ftp\_client\_restart\_marker\_t \* new\_marker, globus\_ftp\_client\_restart\_marker\_t \* marker)**

Create a copy of a restart marker.

This function copies the contents of marker to new\_marker.

#### Parameters:

*new\_marker* A pointer to a new restart marker.

*marker* The marker to copy.

#### See also:

**globus\_ftp\_client\_restart\_marker\_init()** (p. 4), **globus\_ftp\_client\_restart\_marker\_destroy()** (p. 4)

### 5.2.2.3 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_destroy (globus\_ftp\_client\_restart\_marker\_t \* marker)**

Destroy a restart marker.

#### Parameters:

*marker* Restart marker. This marker must be initialized by either calling **globus\_ftp\_client\_restart\_marker\_init()** (p. 4) or **globus\_ftp\_client\_restart\_marker\_copy()** (p. 4)

#### See also:

**globus\_ftp\_client\_restart\_marker\_t** (p. 109), **globus\_ftp\_client\_restart\_marker\_init()** (p. 4), **globus\_ftp\_client\_restart\_marker\_copy()** (p. 4)

#### 5.2.2.4 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_insert\_range (globus\_ftp\_client\_restart\_marker\_t \* marker, globus\_off\_t offset, globus\_off\_t end\_offset)**

Insert a range into a restart marker

This function updates a restart marker with a new byte range, suitable for using to restart an extended block mode transfer.

Adjacent ranges within the marker will be combined into a single entry in the marker.

The marker must first be initialized by calling **globus\_ftp\_client\_restart\_marker\_init()** (p. 4) or **globus\_ftp\_client\_restart\_marker\_copy()** (p. 4).

A marker can only hold a range list or a stream offset. Calling this function after calling **globus\_ftp\_client\_restart\_marker\_set\_offset()** (p. 6) will result in a marker suitable only for use restarting an extended block mode transfer.

##### **Parameters:**

*marker* A restart marker

*offset* The starting offset of the range.

*end\_offset* The ending offset of the range.

##### **See also:**

**globus\_ftp\_client\_restart\_marker\_set\_offset()** (p. 6) **globus\_ftp\_client\_operationattr\_set\_mode()** (p. 49)

#### 5.2.2.5 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_set\_ascii\_offset (globus\_ftp\_client\_restart\_marker\_t \* marker, globus\_off\_t offset, globus\_off\_t ascii\_offset)**

Set the offset for a restart marker.

This function modifies a restart marker to contain a stream offset, suitable for using to restart a stream mode transfer.

The marker must first be initialized by calling **globus\_ftp\_client\_restart\_marker\_init()** (p. 4) or **globus\_ftp\_client\_restart\_marker\_copy()** (p. 4).

A marker can only hold a range list or a stream offset. Calling this function after calling **globus\_ftp\_client\_restart\_marker\_insert\_range()** (p. 5) will delete the ranges associated with the marker, and replace it with a marker suitable only for use restarting a stream mode transfer.

When restarting an ASCII type transfer, use **globus\_ftp\_client\_restart\_marker\_set\_ascii\_offset()** (p. 5) to set both the offset used in the local representation of an ASCII file, and the network representation of the ASCII file. For UNIX systems, the former includes counts newlines as one character towards the file offset, and the latter counts them as 2 characters (CRLF).

##### **Parameters:**

*marker* A restart marker

*offset* The local stream offset.

*ascii\_offset* The network ascii representation of the offset.

##### **See also:**

**globus\_ftp\_client\_restart\_marker\_insert\_range()** (p. 5), **globus\_ftp\_client\_restart\_marker\_set\_offset()** (p. 6), **globus\_ftp\_client\_operationattr\_set\_mode()** (p. 49), **globus\_ftp\_client\_operationattr\_set\_type()** (p. 49)

#### 5.2.2.6 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_set\_offset** (globus\_ftp\_client\_restart\_marker\_t \* *marker*, globus\_off\_t *offset*)

Set the offset for a restart marker.

This function modifies a restart marker to contain a stream offset, suitable for using to restart a stream mode transfer.

The marker must first be initialized by calling **globus\_ftp\_client\_restart\_marker\_init()** (p. 4) or **globus\_ftp\_client\_restart\_marker\_copy()** (p. 4).

A marker can only hold a range list or a stream offset. Calling this function after calling **globus\_ftp\_client\_restart\_marker\_insert\_range()** (p. 5) will delete the ranges associated with the marker, and replace it with a marker suitable only for use restarting a stream mode transfer.

When restarting an ASCII type transfer, the offset must take into account the additional carriage return characters added to the data stream.

##### **Parameters:**

*marker* A restart marker

*offset* The stream offset

##### **See also:**

**globus\_ftp\_client\_restart\_marker\_insert\_range()** (p. 5), **globus\_ftp\_client\_operationattr\_set\_mode()** (p. 49), **globus\_ftp\_client\_operationattr\_set\_type()** (p. 49)

#### 5.2.2.7 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_get\_total** (globus\_ftp\_client\_restart\_marker\_t \* *marker*, globus\_off\_t \* *total\_bytes*)

Get total bytes accounted for in restart marker

This function will return the sum of all bytes accounted for in a restart marker.

If this restart marker contains a stream offset then this value is the same as the offset (not the ascii offset) that it was set with. If it is a range list, it a sum of all the bytes in the ranges.

##### **Parameters:**

*marker* A previously initialized or copied restart marker

*total\_bytes* pointer to storage for total bytes in marker

##### **Returns:**

- Error on NULL marker or total bytes
- <possible return>="">

#### 5.2.2.8 **globus\_result\_t globus\_ftp\_client\_restart\_marker\_to\_string** (globus\_ftp\_client\_restart\_marker\_t \* *marker*, char \*\* *marker\_string*)

Create a string representation of a restart marker.

This function sets the *marker\_string* parameter to point to a freshly allocated string suitable for sending as an argument to the FTP REST command, or for a later call to **globus\_ftp\_client\_restart\_marker\_from\_string()** (p. 7).

The string pointed to by *marker\_string* must be freed by the caller.

**Parameters:**

*marker* An initialized FTP client restart marker.

*marker\_string* A pointer to a char \* to be set to a freshly allocated marker string.

**See also:**

**Restart Markers** (p. 3)

### 5.2.2.9 globus\_result\_t globus\_ftp\_client\_restart\_marker\_from\_string (globus\_ftp\_client\_restart\_marker\_t \* *marker*, const char \* *marker\_string*)

Initialize a restart marker from a string.

This function initializes a new restart, *marker*, based on the *marker\_string* parameter. The string may be either a single offset for a stream-mode restart marker, or a comma-separated list of start-end ranges.

**Parameters:**

*marker* The restart marker to be uninitialized.

*marker\_string* The string containing a textual representation of a restart marker.

**See also:**

**Restart Markers** (p. 3)

## 5.3 Handle Management

Create/Destroy/Modify an FTP Client Handle.

**Initialize**

- globus\_result\_t globus\_ftp\_client\_handle\_init (globus\_ftp\_client\_handle\_t \*handle, globus\_ftp\_client\_handleattr\_t \*attr)

**Destroy**

- globus\_result\_t globus\_ftp\_client\_handle\_destroy (globus\_ftp\_client\_handle\_t \*handle)

**URL Caching**

- globus\_result\_t globus\_ftp\_client\_handle\_cache\_url\_state (globus\_ftp\_client\_handle\_t \*handle, const char \*url)
- globus\_result\_t globus\_ftp\_client\_handle\_flush\_url\_state (globus\_ftp\_client\_handle\_t \*handle, const char \*url)

**User Pointer**

- globus\_result\_t globus\_ftp\_client\_handle\_set\_user\_pointer (globus\_ftp\_client\_handle\_t \*handle, void \*user\_pointer)
- globus\_result\_t globus\_ftp\_client\_handle\_get\_user\_pointer (const globus\_ftp\_client\_handle\_t \*handle, void \*\*user\_pointer)



## Plugins

- `globus_result_t globus_ftp_client_handle_add_plugin (globus_ftp_client_handle_t *handle, globus_ftp_client_plugin_t *plugin)`
- `globus_result_t globus_ftp_client_handle_remove_plugin (globus_ftp_client_handle_t *handle, globus_ftp_client_plugin_t *plugin)`

## Typedefs

- `typedef globus_i_ftp_client_handle_t * globus_ftp_client_handle_t`

### 5.3.1 Detailed Description

Create/Destroy/Modify an FTP Client Handle.

Within the Globus FTP Client Library, all FTP operations require a handle parameter. Currently, only one FTP operation may be in progress at once per FTP handle. FTP connections may be cached between FTP operations, for improved performance.

This section defines operations to create and destroy FTP Client handles, as well as to modify handles' connection caches.

### 5.3.2 Typedef Documentation

#### 5.3.2.1 `typedef struct globus_i_ftp_client_handle_t* globus_ftp_client_handle_t`

FTP Client Handle.

An FTP client handle is used to associate state with a group of operations. Handles can have **attributes** (p. 12) associated with them. All FTP **operations** (p. 18) take a handle pointer as a parameter.

See also:

`globus_ftp_client_handle_init()` (p. 8), `globus_ftp_client_handle_destroy()` (p. 9), `globus_ftp_client_handleattr_t` (p. 12)

### 5.3.3 Function Documentation

#### 5.3.3.1 `globus_result_t globus_ftp_client_handle_init (globus_ftp_client_handle_t * handle, globus_ftp_client_handleattr_t * attr)`

Initialize a client FTP handle.

Initialize an FTP handle which can be used in subsequent get, put, or transfer requests. A handle may have at most one get, put, or third-party transfer in progress.

Parameters:

*handle* The handle to be initialized.

*attr* Initial attributes to be used to create this handle.

See also:

`globus_ftp_client_handle_destroy()` (p. 9)

### 5.3.3.2 `globus_result_t globus_ftp_client_handle_destroy (globus_ftp_client_handle_t * handle)`

Destroy a client FTP handle.

A FTP client handle may not be destroyed if a get, put, or third-party transfer is in progress.

#### Parameters:

*handle* The handle to be destroyed.

#### See also:

`globus_ftp_client_handle_init()` (p. 8)

### 5.3.3.3 `globus_result_t globus_ftp_client_handle_cache_url_state (globus_ftp_client_handle_t * handle, const char * url)`

Cache connections to an FTP server.

Explicitly cache connections to URL server in an FTP handle. When an URL is cached, the client library will not close the connection to the URL server after a file transfer completes.

#### Parameters:

*handle* Handle which will contain a cached connection to the URL server.

*url* The URL of the FTP or GSIFTP server to cache.

#### See also:

`globus_ftp_client_flush_url_state()`

### 5.3.3.4 `globus_result_t globus_ftp_client_handle_flush_url_state (globus_ftp_client_handle_t * handle, const char * url)`

Remove a cached connection from the FTP client handle.

Explicitly remove a cached connection to an FTP server from the FTP handle. If an idle connection to an FTP server exists, it will be closed.

#### Parameters:

*handle* Handle which will contain a cached connection to the URL server.

*url* The URL of the FTP or GSIFTP server to cache.

### 5.3.3.5 `globus_result_t globus_ftp_client_handle_set_user_pointer (globus_ftp_client_handle_t * handle, void * user_pointer)`

Set/Get the user pointer field from an ftp client handle.

The user pointer is provided to all the user of the FTP client library to associate a pointer to any application-specific data to an FTP client handle. This pointer is never internally used by the client library.

#### Parameters:

*handle* The FTP client handle to set or query.

*user\_pointer* The value of the user pointer field.

#### Note:

Access to the `user_pointer` are not synchronized, the user must take care to make sure that multiple threads are not modifying it's value.

#### **5.3.3.6 globus\_result\_t globus\_ftp\_client\_handle\_add\_plugin (globus\_ftp\_client\_handle\_t \* *handle*, globus\_ftp\_client\_plugin\_t \* *plugin*)**

Add a plugin to an FTP client handle.

This function adds a plugin to an FTP client handle after it has been created. Plugins may be added to an ftp client handle whenever an operation is not in progress. The plugin will be appended to the list of plugins present in the handle, and will be invoked during any subsequent operations processed with this handle.

Only one instance of a particular plugin may be added to a particular handle.

Plugins may be removed from a handle by calling `globus_ftp_client_remove_plugin()`.

##### **Parameters:**

*handle* The FTP client handle to set or query.

*plugin* A pointer to the plugin structure to add to this handle.

##### **See also:**

`globus_ftp_client_remove_plugin()`, `globus_ftp_client_handleattr_add_plugin()`, `globus_ftp_client_handleattr_remove_plugin()`

#### **5.3.3.7 globus\_result\_t globus\_ftp\_client\_handle\_remove\_plugin (globus\_ftp\_client\_handle\_t \* *handle*, globus\_ftp\_client\_plugin\_t \* *plugin*)**

Remove a plugin to an FTP client handle.

This function removes a plugin from an FTP client handle after it has been created. Plugins may be removed from an ftp client handle whenever an operation is not in progress. The plugin will be removed from the list of plugins, and will not be used during any subsequent operations processed with this handle.

This function can remove plugins which were added at **handle initialization time** (p. 8) or by calling `globus_ftp_client_handle_add_plugin()` (p. 10).

##### **Parameters:**

*handle* The FTP client handle to set or query.

*plugin* A pointer to the plugin structure to remove from this handle.

##### **See also:**

`globus_ftp_client_add_plugin()`, `globus_ftp_client_handleattr_add_plugin()`, `globus_ftp_client_handleattr_remove_plugin()`

#### **5.3.3.8 globus\_result\_t globus\_ftp\_client\_handle\_get\_user\_pointer (const globus\_ftp\_client\_handle\_t \* *handle*, void \*\* *user\_pointer*)**

Set/Get the user pointer field from an ftp client handle.

The user pointer is provided to all the user of the FTP client library to associate a pointer to any application-specific data to an FTP client handle. This pointer is never internally used by the client library.

##### **Parameters:**

*handle* The FTP client handle to set or query.

*user\_pointer* The value of the user pointer field.

##### **Note:**

Access to the `user_pointer` are not synchronized, the user must take care to make sure that multiple threads are not modifying it's value.

## 5.4 Handle Attributes

Handle attributes are used to control additional features of the FTP Client handle.

### Initialize

- `globus_result_t globus_ftp_client_handleattr_init (globus_ftp_client_handleattr_t *attr)`

### Destroy

- `globus_result_t globus_ftp_client_handleattr_destroy (globus_ftp_client_handleattr_t *attr)`

### Copy

- `globus_result_t globus_ftp_client_handleattr_copy (globus_ftp_client_handleattr_t *dest, globus_ftp_client_handleattr_t *src)`

### Connection Caching

- `globus_result_t globus_ftp_client_handleattr_set_cache_all (globus_ftp_client_handleattr_t *attr, globus_bool_t cache_all)`
- `globus_result_t globus_ftp_client_handleattr_get_cache_all (const globus_ftp_client_handleattr_t *attr, globus_bool_t *cache_all)`

### Non-root relative URLs

- `globus_result_t globus_ftp_client_handleattr_set_rfc1738_url (globus_ftp_client_handleattr_t *attr, globus_bool_t rfc1738_url)`
- `globus_result_t globus_ftp_client_handleattr_get_rfc1738_url (const globus_ftp_client_handleattr_t *attr, globus_bool_t *rfc1738_url)`

### Client Info

- `globus_result_t globus_ftp_client_handleattr_set_clientinfo (globus_ftp_client_handleattr_t *attr, const char *app_name, const char *app_version, const char *other)`
- `globus_result_t globus_ftp_client_handleattr_get_clientinfo (globus_ftp_client_handleattr_t *attr, char **app_name, char **app_version, char **other)`

### GridFTP2 support

- `globus_result_t globus_ftp_client_handleattr_set_gridftp2 (globus_ftp_client_handleattr_t *attr, globus_bool_t gridftp2)`
- `globus_result_t globus_ftp_client_handleattr_get_gridftp2 (const globus_ftp_client_handleattr_t *attr, globus_bool_t *gridftp2)`

### Command Pipelining

- `globus_result_t globus_ftp_client_handleattr_set_pipeline (globus_ftp_client_handleattr_t *attr, globus_size_t outstanding_commands, globus_ftp_client_pipeline_callback_t pipeline_callback, void *pipeline_arg)`

- **globus\_result\_t globus\_ftp\_client\_handleattr\_get\_pipeline** (const **globus\_ftp\_client\_handleattr\_t** \*attr, **globus\_size\_t** \*outstanding\_commands, **globus\_ftp\_client\_pipeline\_callback\_t** \*pipeline\_callback, void \*\*pipeline\_arg)

## URL Caching

- **globus\_result\_t globus\_ftp\_client\_handleattr\_add\_cached\_url** (**globus\_ftp\_client\_handleattr\_t** \*attr, const char \*url)
- **globus\_result\_t globus\_ftp\_client\_handleattr\_remove\_cached\_url** (**globus\_ftp\_client\_handleattr\_t** \*attr, const char \*url)

## Netlogger management

- **globus\_result\_t globus\_ftp\_client\_handleattr\_set\_netlogger** (**globus\_ftp\_client\_handleattr\_t** \*attr, **globus\_netlogger\_handle\_t** \*nl\_handle)
- **globus\_result\_t globus\_ftp\_client\_handleattr\_set\_netlogger\_ftp\_io** (**globus\_ftp\_client\_handleattr\_t** \*attr, **globus\_netlogger\_handle\_t** \*nl\_handle, **globus\_bool\_t** ftp, **globus\_bool\_t** io)

## Plugin Management

- **globus\_result\_t globus\_ftp\_client\_handleattr\_add\_plugin** (**globus\_ftp\_client\_handleattr\_t** \*attr, **globus\_ftp\_client\_plugin\_t** \*plugin)
- **globus\_result\_t globus\_ftp\_client\_handleattr\_remove\_plugin** (**globus\_ftp\_client\_handleattr\_t** \*attr, **globus\_ftp\_client\_plugin\_t** \*plugin)

## Typedefs

- typedef **globus\_i\_ftp\_client\_handleattr\_t** \* **globus\_ftp\_client\_handleattr\_t**

### 5.4.1 Detailed Description

Handle attributes are used to control additional features of the FTP Client handle.

These features are operation independent.

The attribute which can currently set on a handle concern the connection caching behavior of the handle, and the associations of plugins with a handle.

See also:

**globus\_ftp\_client\_handle\_t** (p. 8)

### 5.4.2 Typedef Documentation

#### 5.4.2.1 typedef struct **globus\_i\_ftp\_client\_handleattr\_t**\* **globus\_ftp\_client\_handleattr\_t**

Handle Attributes.

Handle attributes are used to control the caching behavior of the ftp client handle, and to implement the plugin features for reliability and performance tuning.

See also:

**globus\_ftp\_client\_handle\_t** (p. 8), **Handle Attributes** (p. 11)

### 5.4.3 Function Documentation

#### 5.4.3.1 `globus_result_t globus_ftp_client_handleattr_init (globus_ftp_client_handleattr_t * attr)`

Initialize an FTP client handle attribute set.

This function creates an empty FTP Client handle attribute set. This function must be called on each attribute set before any of the other functions in this section may be called.

##### Parameters:

*attr* The new handle attribute.

##### See also:

`globus_ftp_client_handleattr_destroy()` (p. 13)

#### 5.4.3.2 `globus_result_t globus_ftp_client_handleattr_destroy (globus_ftp_client_handleattr_t * attr)`

Destroy an FTP client handle attribute set.

This function destroys an ftp client handle attribute set. All attributes on this set will be lost. The user must call `globus_ftp_client_handleattr_init()` (p. 13) again on this attribute set before calling any other handle attribute functions on it.

##### Parameters:

*attr* The attribute set to destroy.

#### 5.4.3.3 `globus_result_t globus_ftp_client_handleattr_copy (globus_ftp_client_handleattr_t * dest, globus_ftp_client_handleattr_t * src)`

Create a duplicate of a handle attribute set.

The duplicated attribute set has a deep copy of all data in the attribute set, so the original may be destroyed while the copy is still valid.

##### Parameters:

*dest* The attribute set to be initialized to the same values as *src*.

*src* The original attribute set to duplicate.

#### 5.4.3.4 `globus_result_t globus_ftp_client_handleattr_set_cache_all (globus_ftp_client_handleattr_t * attr, globus_bool_t cache_all)`

Set/Get the cache all connections attribute for an ftp client handle attribute set.

This attribute allows the user to cause all control connections to be cached between ftp operations. When this is enabled, the user skips the authentication handshake and connection establishment overhead for multiple subsequent ftp operations to the same server.

Memory and network connections associated with the caching will be used until the handle is destroyed. If fine grained caching is needed, then the user should disable this attribute and explicitly cache specific URLs.

##### Parameters:

*attr* Attribute to query or modify.

*cache\_all* Value of the `cache_all` attribute.

See also:

`globus_ftp_client_handleattr_add_cached_url()` (p. 15), `globus_ftp_client_handleattr_remove_cached_url()` (p. 15), `globus_ftp_client_handle_cache_url_state()` (p. 9) `globus_ftp_client_handle_flush_url_state()` (p. 9)

**5.4.3.5** `globus_result_t globus_ftp_client_handleattr_set_rfc1738_url (globus_ftp_client_handleattr_t * attr, globus_bool_t rfc1738_url)`

Enable/Disable rfc1738 support for non-root relative URLs.

**Parameters:**

*attr* Attribute to modify

*rfc1738\_url* Set to GLOBUS\_TRUE to enable non-root relative URLs. Default of GLOBUS\_FALSE specifies root-relative URLs.

**5.4.3.6** `globus_result_t globus_ftp_client_handleattr_set_clientinfo (globus_ftp_client_handleattr_t * attr, const char * app_name, const char * app_version, const char * other)`

Set/Get client info reported to server.

**Parameters:**

*attr* Attribute to modify

*app\_name* Name of client application.

*app\_version* Client application specific version string.

*other* Additional client info to be reported to the server. This may be used to pass custom info to a custom server module. The format of the string must be: key1="value1";key2="value2";[keyn="valuen";]

Any parameter may be NULL. By default, generic library info will be reported to the server – set all NULL to disable this.

**5.4.3.7** `globus_result_t globus_ftp_client_handleattr_set_gridftp2 (globus_ftp_client_handleattr_t * attr, globus_bool_t gridftp2)`

Enable/Disable GridFTP2 [GFD.41] support for servers supporting it.

This currently only applies to the GET/PUT command.

**Parameters:**

*attr* Attribute to modify

*gridftp2* Set to GLOBUS\_FALSE to disable GridFTP2 support. GridFTP2 support is enabled by default for servers that support it.

**5.4.3.8** `globus_result_t globus_ftp_client_handleattr_set_pipeline (globus_ftp_client_handleattr_t * attr, globus_size_t outstanding_commands, globus_ftp_client_pipeline_callback_t pipeline_callback, void * pipeline_arg)`

Enable/Disable command queueing for pipelined transfers.

**Parameters:**

*attr* Attribute to modify

***outstanding\_commands*** Set to the number of commands to have sent without receiving a reply. Use 0 for the library default.

***pipeline\_callback*** Set to a function of type `globus_ftp_client_pipeline_callback_t` to enable command pipelining. This function will be called during a transfer operation to request the next urls to be transferred.

***pipeline\_arg*** User data that will be passed in the `pipeline_callback`.

#### **5.4.3.9 `globus_result_t globus_ftp_client_handleattr_add_cached_url (globus_ftp_client_handleattr_t * attr, const char * url)`**

Enable/Disable caching for a specific URL.

This function adds/removes the specified URL into the default cache for a handle attribute. Handles initialized with this attr will keep connections to FTP servers associated with the URLs in its cache open between **operations** (p. 18).

##### **Parameters:**

***attr*** Attribute to modify

***url*** URL string to cache

#### **5.4.3.10 `globus_result_t globus_ftp_client_handleattr_remove_cached_url (globus_ftp_client_handleattr_t * attr, const char * url)`**

Enable/Disable caching for a specific URL.

This function adds/removes the specified URL into the default cache for a handle attribute. Handles initialized with this attr will keep connections to FTP servers associated with the URLs in its cache open between **operations** (p. 18).

##### **Parameters:**

***attr*** Attribute to modify

***url*** URL string to cache

#### **5.4.3.11 `globus_result_t globus_ftp_client_handleattr_set_netlogger (globus_ftp_client_handleattr_t * attr, globus_netlogger_handle_t * nl_handle)`**

Set the netlogger handle used with this transfer.

Each handle can have a netlogger handle associated with it for logging its data.

Only 1 netlogger handle can be associated with a client handle.

##### **Parameters:**

***attr*** The attribute set to modify.

***nl\_handle*** The open netlogger handle to be associated with this attribute set.

#### **5.4.3.12 `globus_result_t globus_ftp_client_handleattr_add_plugin (globus_ftp_client_handleattr_t * attr, globus_ftp_client_plugin_t * plugin)`**

Add/Remove a plugin to a handle attribute set.



Each handle attribute set contains a list of plugins associated with it. When a handle is created with a particular attribute set, it will be associated with a copy of those plugins.

Only one instance of a specific plugin may be added to an attribute set. Each plugin must have a different name.

A copy of the plugin is created via the plugins 'copy' method when it is added to an attribute set. Thus, any changes to a particular plugin must be done before the plugin is added to an attribute set, and before the attribute set is used to create handles.

**Parameters:**

*attr* The attribute set to modify.

*plugin* The plugin to add or remove from the list.

**5.4.3.13 globus\_result\_t globus\_ftp\_client\_handleattr\_get\_cache\_all (const globus\_ftp\_client\_handleattr\_t \* attr, globus\_bool\_t \* cache\_all)**

Set/Get the cache all connections attribute for an ftp client handle attribute set.

This attribute allows the user to cause all control connections to be cached between ftp operations. When this is enabled, the user skips the authentication handshake and connection establishment overhead for multiple subsequent ftp operations to the same server.

Memory and network connections associated with the caching will be used until the handle is destroyed. If fine grained caching is needed, then the user should disable this attribute and explicitly cache specific URLs.

**Parameters:**

*attr* Attribute to query or modify.

*cache\_all* Value of the cache\_all attribute.

**See also:**

**globus\_ftp\_client\_handleattr\_add\_cached\_url()** (p. 15), **globus\_ftp\_client\_handleattr\_remove\_cached\_url()** (p. 15), **globus\_ftp\_client\_handle\_cache\_url\_state()** (p. 9) **globus\_ftp\_client\_handle\_flush\_url\_state()** (p. 9)

**5.4.3.14 globus\_result\_t globus\_ftp\_client\_handleattr\_get\_rfc1738\_url (const globus\_ftp\_client\_handleattr\_t \* attr, globus\_bool\_t \* rfc1738\_url)**

Enable/Disable rfc1738 support for non-root relative URLs.

**Parameters:**

*attr* Attribute to modify

*rfc1738\_url* Set to GLOBUS\_TRUE to enable non-root relative URLs. Default of GLOBUS\_FALSE specifies root-relative URLs.

**5.4.3.15 globus\_result\_t globus\_ftp\_client\_handleattr\_get\_clientinfo (globus\_ftp\_client\_handleattr\_t \* attr, char \*\* app\_name, char \*\* app\_version, char \*\* other)**

Set/Get client info reported to server.

**Parameters:**

*attr* Attribute to modify

**app\_name** Name of client application.

**app\_version** Client application specific version string.

**other** Additional client info to be reported to the server. This may be used to pass custom info to a custom server module. The format of the string must be: key1="value1";key2="value2";[keyn="valuen";]

Any parameter may be NULL. By default, generic library info will be reported to the server – set all NULL to disable this.

#### **5.4.3.16 globus\_result\_t globus\_ftp\_client\_handleattr\_get\_gridftp2 (const globus\_ftp\_client\_handleattr\_t \* attr, globus\_bool\_t \* gridftp2)**

Enable/Disable GridFTP2 [GFD.41] support for servers supporting it.

This currently only applies to the GET/PUT command.

##### **Parameters:**

**attr** Attribute to modify

**gridftp2** Set to GLOBUS\_FALSE to disable GridFTP2 support. GridFTP2 support is enabled by default for servers that support it.

#### **5.4.3.17 globus\_result\_t globus\_ftp\_client\_handleattr\_get\_pipeline (const globus\_ftp\_client\_handleattr\_t \* attr, globus\_size\_t \* outstanding\_commands, globus\_ftp\_client\_pipeline\_callback\_t \* pipeline\_callback, void \*\* pipeline\_arg)**

Enable/Disable command queueing for pipelined transfers.

##### **Parameters:**

**attr** Attribute to modify

**outstanding\_commands** Set to the number of commands to have sent without receiving a reply. Use 0 for the library default.

**pipeline\_callback** Set to a function of type globus\_ftp\_client\_pipeline\_callback\_t to enable command pipelining. This function will be called during a transfer operation to request the next urls to be transferred.

**pipeline\_arg** User data that will be passed in the pipeline\_callback.

#### **5.4.3.18 globus\_result\_t globus\_ftp\_client\_handleattr\_set\_netlogger\_ftp\_io (globus\_ftp\_client\_handleattr\_t \* attr, globus\_netlogger\_handle\_t \* nl\_handle, globus\_bool\_t ftp, globus\_bool\_t io)**

Set the netlogger handle used with this transfer.

Each handle can have a netlogger handle associated with it for logging its data.

Only 1 netlogger handle can be associated with a client handle.

##### **Parameters:**

**attr** The attribute set to modify.

**nl\_handle** The open netlogger handle to be associated with this attribute set.

#### 5.4.3.19 `globus_result_t globus_ftp_client_handleattr_remove_plugin (globus_ftp_client_handleattr_t *attr, globus_ftp_client_plugin_t *plugin)`

Add/Remove a plugin to a handle attribute set.

Each handle attribute set contains a list of plugins associated with it. When a handle is created with a particular attribute set, it will be associated with a copy of those plugins.

Only one instance of a specific plugin may be added to an attribute set. Each plugin must have a different name.

A copy of the plugin is created via the plugins 'copy' method when it is added to an attribute set. Thus, any changes to a particular plugin must be done before the plugin is added to an attribute set, and before the attribute set is used to create handles.

#### Parameters:

*attr* The attribute set to modify.

*plugin* The plugin to add or remove from the list.

## 5.5 FTP Operations

Initiate an FTP operation.

#### File or Directory Existence

- `globus_result_t globus_ftp_client_exists (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

#### Features

- `globus_result_t globus_ftp_client_features_init (globus_ftp_client_features_t *u_features)`
- `globus_result_t globus_ftp_client_features_destroy (globus_ftp_client_features_t *u_features)`
- `globus_result_t globus_ftp_client_feat (globus_ftp_client_handle_t *u_handle, char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_features_t *u_features, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`
- `globus_result_t globus_ftp_client_is_feature_supported (const globus_ftp_client_features_t *u_features, globus_ftp_client_tristate_t *answer, globus_ftp_client_probed_feature_t feature)`

#### Make Directory

- `globus_result_t globus_ftp_client_mkdir (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

#### Remove Directory

- `globus_result_t globus_ftp_client_rmdir (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Change Working Directory

- `globus_result_t globus_ftp_client_cwd (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_byte_t **cwd_buffer, globus_size_t *cwd_buffer_length, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Delete

- `globus_result_t globus_ftp_client_delete (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## List

- `globus_result_t globus_ftp_client_list (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`
- `globus_result_t globus_ftp_client_verbose_list (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## STAT

- `globus_result_t globus_ftp_client_stat (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_byte_t **stat_buffer, globus_size_t *stat_buffer_length, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## MLST

- `globus_result_t globus_ftp_client_mlst (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_byte_t **mlst_buffer, globus_size_t *mlst_buffer_length, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Move

- `globus_result_t globus_ftp_client_move (globus_ftp_client_handle_t *u_handle, const char *source_url, const char *dest_url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## chmod

- `globus_result_t globus_ftp_client_chmod (globus_ftp_client_handle_t *u_handle, const char *url, int mode, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## chgrp

- `globus_result_t globus_ftp_client_chgrp (globus_ftp_client_handle_t *u_handle, const char *url, const char *group, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## utime

- globus\_result\_t globus\_ftp\_client\_utime (globus\_ftp\_client\_handle\_t \*u\_handle, const char \*url, const struct tm \*utime\_time, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

## symlink

- globus\_result\_t globus\_ftp\_client\_symlink (globus\_ftp\_client\_handle\_t \*u\_handle, const char \*source\_url, const char \*link\_url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

## Get

- globus\_result\_t globus\_ftp\_client\_get (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_partial\_get (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_off\_t partial\_offset, globus\_off\_t partial\_end\_offset, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_extended\_get (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, const char \*eret\_alg\_str, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

## Put

- globus\_result\_t globus\_ftp\_client\_put (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_partial\_put (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_off\_t partial\_offset, globus\_off\_t partial\_end\_offset, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_extended\_put (globus\_ftp\_client\_handle\_t \*handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_restart\_marker\_t \*restart, const char \*esto\_alg\_str, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

## 3rd Party Transfer

- globus\_result\_t globus\_ftp\_client\_third\_party\_transfer (globus\_ftp\_client\_handle\_t \*handle, const char \*source\_url, globus\_ftp\_client\_operationattr\_t \*source\_attr, const char \*dest\_url, globus\_ftp\_client\_operationattr\_t \*dest\_attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_partial\_third\_party\_transfer (globus\_ftp\_client\_handle\_t \*handle, const char \*source\_url, globus\_ftp\_client\_operationattr\_t \*source\_attr, const char \*dest\_url, globus\_ftp\_client\_operationattr\_t \*dest\_attr, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_off\_t partial\_offset, globus\_off\_t partial\_end\_offset, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)
- globus\_result\_t globus\_ftp\_client\_extended\_third\_party\_transfer (globus\_ftp\_client\_handle\_t \*handle, const char \*source\_url, globus\_ftp\_client\_operationattr\_t \*source\_attr, const char \*eret\_alg\_str, const char \*dest\_url, globus\_ftp\_client\_operationattr\_t \*dest\_attr, const char \*esto\_alg\_str, globus\_ftp\_client\_restart\_marker\_t \*restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

## Modification Time

- `globus_result_t globus_ftp_client_modification_time (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_abstime_t *modification_time, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Size

- `globus_result_t globus_ftp_client_size (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_off_t *size, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Cksm

- `globus_result_t globus_ftp_client_cksm (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, char *cksm, globus_off_t offset, globus_off_t length, const char *algorithm, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

## Abort

- `globus_result_t globus_ftp_client_abort (globus_ftp_client_handle_t *u_handle)`

## Typedefs

- `typedef void(*) globus_ftp_client_complete_callback_t (void *user_arg, globus_ftp_client_handle_t *handle, globus_object_t *error)`
- `typedef globus_i_ftp_client_features_s * globus_ftp_client_features_t`

## Enumerations

- `enum globus_ftp_client_tristate_t`
- `enum globus_ftp_client_probed_feature_t`

## Functions

- `globus_result_t globus_ftp_client_machine_list (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`
- `globus_result_t globus_ftp_client_recursive_list (globus_ftp_client_handle_t *u_handle, const char *url, globus_ftp_client_operationattr_t *attr, globus_ftp_client_complete_callback_t complete_callback, void *callback_arg)`

### 5.5.1 Detailed Description

Initiate an FTP operation.

This module contains the API functions for a user to request a get, put, third-party transfer, or other FTP file operation.

## 5.5.2 Typedef Documentation

### 5.5.2.1 typedef void(\*) globus\_ftp\_client\_complete\_callback\_t(void \*user\_arg, globus\_ftp\_client\_handle\_t \*handle, globus\_object\_t \*error)

Operation complete callback.

Every FTP Client operation (get, put, transfer, mkdir, etc) is asynchronous. A callback of this type is passed to each of the operation function calls to let the user know when the operation is complete. The completion callback is called only once per operation, after all other callbacks for the operation have returned.

#### Parameters:

*user\_arg* The user\_arg parameter passed to the operation.

*handle* The handle on which the operation was done.

*error* A Globus error object indicating any problem which occurred, or GLOBUS\_SUCCESS, if the operation completed successfully.

### 5.5.2.2 typedef struct globus\_i\_ftp\_client\_features\_s\* globus\_ftp\_client\_features\_t

Feature Handle

Handle used to associate state with feature operations.

#### See also:

**globus\_ftp\_client\_feat** (p. 23), **globus\_ftp\_client\_features\_init** (p. 23), **globus\_ftp\_client\_features\_destroy** (p. 23)

## 5.5.3 Enumeration Type Documentation

### 5.5.3.1 enum globus\_ftp\_client\_tristate\_t

Types for feature existence

FALSE and TRUE are known to be fact that a feature does or does not exist MAYBE means that the feature may exist.

### 5.5.3.2 enum globus\_ftp\_client\_probed\_feature\_t

Types of features.

## 5.5.4 Function Documentation

### 5.5.4.1 globus\_result\_t globus\_ftp\_client\_exists(globus\_ftp\_client\_handle\_t \*u\_handle, const char \*url, globus\_ftp\_client\_operationattr\_t \*attr, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \*callback\_arg)

Check for the existence of a file or directory on an FTP server.

This function attempts to determine whether the specified URL points to a valid file or directory. The *complete\_callback* will be invoked with the result of the existence check passed as a globus error object, or GLOBUS\_SUCCESS.

#### Parameters:

*u\_handle* An FTP Client handle to use for the existence check operation.

**url** The URL of the directory or file to check. The URL may be an ftp or gsiftp URL.  
**attr** Attributes to use for this operation.  
**complete\_callback** Callback to be invoked once the existence operation is completed.  
**callback\_arg** Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- **u\_handle** is **GLOBUS\_NULL**
- **url** is **GLOBUS\_NULL**
- **url** cannot be parsed
- **url** is not a ftp or gsiftp url
- **complete\_callback** is **GLOBUS\_NULL**
- **handle** already has an operation in progress

**5.5.4.2 globus\_result\_t globus\_ftp\_client\_features\_init (globus\_ftp\_client\_features\_t \* u\_features)**

Initialize the feature set, to be later used by **globus\_ftp\_client\_feat()** (p. 23).

Each feature gets initial value **GLOBUS\_FTP\_CLIENT\_MAYBE**.

**Note:**

Structure initialized by this function must be destroyed using **globus\_ftp\_client\_features\_destroy()** (p. 23)

**Returns:**

**GLOBUS\_SUCCESS** on success, otherwise error.

**5.5.4.3 globus\_result\_t globus\_ftp\_client\_features\_destroy (globus\_ftp\_client\_features\_t \* u\_features)**

Destroy the feature set.

**Note:**

Structure passed to this function must have been previously initialized by **globus\_ftp\_client\_features\_init()** (p. 23).

**Returns:**

**GLOBUS\_SUCCESS** on success, otherwise error.

**5.5.4.4 globus\_result\_t globus\_ftp\_client\_feat (globus\_ftp\_client\_handle\_t \* u\_handle, char \* url, globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_client\_features\_t \* u\_features, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \* callback\_arg)**

Check the features supported by the server (FTP FEAT command).

After this procedure completes, the features set (parameter **u\_features**) represents the features supported by the server. Prior to calling this procedure, the structure should have been initialized by **globus\_ftp\_client\_features\_init()** (p. 23); afterwards, it should be destroyed by **globus\_ftp\_client\_features\_destroy()** (p. 23). After **globus\_ftp\_client\_feat()** (p. 23) returns, each feature in the list has one of the values: **GLOBUS\_FTP\_CLIENT\_TRUE**, **GLOBUS\_FTP\_CLIENT\_FALSE**, or **GLOBUS\_FTP\_CLIENT\_MAYBE**. The first two denote the server supporting, or not supporting, the given feature. The last one means that the test has not been performed. This is not necessarily caused by error; there might have been no reason to check for this particular feature.



**Parameters:**

- u\_handle* An FTP Client handle to use for the list operation.
- url* The URL to list. The URL may be an ftp or gsiftp URL.
- attr* Attributes for this file transfer.
- u\_features* A pointer to a `globus_ftp_client_features_t` to be filled with the feature set supported by the server.
- complete\_callback* Callback to be invoked once the size check is completed.
- callback\_arg* Argument to be passed to the `complete_callback`.

**Returns:**

This function returns an error when any of these conditions are true:

- *u\_handle* is `GLOBUS_NULL`
- *source\_url* is `GLOBUS_NULL`
- *source\_url* cannot be parsed
- *source\_url* is not a ftp or gsiftp url
- *u\_features* is `GLOBUS_NULL` or badly initialized
- *complete\_callback* is `GLOBUS_NULL`
- handle already has an operation in progress

**5.5.4.5 `globus_result_t globus_ftp_client_is_feature_supported (const globus_ftp_client_features_t * u_features, globus_ftp_client_tristate_t * answer, globus_ftp_client_probed_feature_t feature)`**

Check if the feature is supported by the server.

After the function completes, parameter *answer* contains the state of the server support of the given function. It can have one of the values: `GLOBUS_FTP_CLIENT_TRUE`, `GLOBUS_FTP_CLIENT_FALSE`, or `GLOBUS_FTP_CLIENT_MAYBE`.

**Parameters:**

- u\_features* list of features, as returned by `globus_ftp_client_feat()` (p. 23)
- answer* this variable will contain the answer
- feature* feature number,  $0 \leq \text{feature} < \text{GLOBUS\_FTP\_CLIENT\_FEATURE\_MAX}$

**Returns:**

error when any of the parameters is null or badly initialized

**5.5.4.6 `globus_result_t globus_ftp_client_mkdir (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`**

Make a directory on an FTP server.

This function starts a `mkdir` operation on a FTP server.

When the response to the `mkdir` request has been received the `complete_callback` will be invoked with the result of the `mkdir` operation.

**Parameters:**

- u\_handle* An FTP Client handle to use for the `mkdir` operation.

***url*** The URL for the directory to create. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this operation.

***complete\_callback*** Callback to be invoked once the mkdir is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- *u\_handle* is GLOBUS\_NULL
- *url* is GLOBUS\_NULL
- *url* cannot be parsed
- *url* is not a ftp or gsiftp url
- *complete\_callback* is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.7** `globus_result_t globus_ftp_client_rmdir (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Make a directory on an FTP server.

This function starts a rmdir operation on a FTP server.

When the response to the rmdir request has been received the complete\_callback will be invoked with the result of the rmdir operation.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the rmdir operation.

***url*** The URL for the directory to create. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this operation.

***complete\_callback*** Callback to be invoked once the rmdir is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- *u\_handle* is GLOBUS\_NULL
- *url* is GLOBUS\_NULL
- *url* cannot be parsed
- *url* is not a ftp or gsiftp url
- *complete\_callback* is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.8** `globus_result_t globus_ftp_client_cwd (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_byte_t ** cwd_buffer, globus_size_t * cwd_buffer_length, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Make a directory on an FTP server.

This function starts a cwd operation on a FTP server.

When the response to the cwd request has been received the complete\_callback will be invoked with the result of the cwd operation.

**Parameters:**

- u\_handle*** An FTP Client handle to use for the cwd operation.
- url*** The URL for the directory to cd into. The URL may be an ftp or gsiftp URL.
- attr*** Attributes for this operation.
- cwd\_buffer*** A pointer to a globus\_byte\_t \* to be allocated and filled with the directory name returned by the CWD, if it succeeds. Otherwise, the value pointed to by it is undefined. It is up to the user to free this memory.
- cwd\_buffer\_length*** A pointer to a globus\_size\_t to be filled with the length of the data in cwd\_buffer.
- complete\_callback*** Callback to be invoked once the cwd is completed.
- callback\_arg*** Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- *u\_handle* is GLOBUS\_NULL
- *url* is GLOBUS\_NULL
- *url* cannot be parsed
- *url* is not a ftp or gsiftp url
- *cwd\_buffer* is GLOBUS\_NULL
- *cwd\_buffer\_length* is GLOBUS\_NULL
- *complete\_callback* is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.9 globus\_result\_t globus\_ftp\_client\_delete (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Delete a file on an FTP server.

This function starts a delete operation on a FTP server. Note that this functions will only delete files, not directories.

When the response to the delete request has been received the complete\_callback will be invoked with the result of the delete operation.

**Parameters:**

- u\_handle*** An FTP Client handle to use for the delete operation.
- url*** The URL for the file to delete. The URL may be an ftp or gsiftp URL.
- attr*** Attributes for this file transfer.
- complete\_callback*** Callback to be invoked once the delete is completed.
- callback\_arg*** Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- *u\_handle* is GLOBUS\_NULL
- *url* is GLOBUS\_NULL
- *url* cannot be parsed
- *url* is not a ftp or gsiftp url
- *complete\_callback* is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.10** `globus_result_t globus_ftp_client_list (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Get a file listing from an FTP server.

This function starts a "NLST" transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling `globus_ftp_client_register_read()` (p. 61) to retrieve the data associated with this listing.

When all of the data associated with the listing is retrieved, and all of the data callbacks have been called, or if the list request is aborted, the `complete_callback` will be invoked with the final status of the list.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL to list. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*complete\_callback* Callback to be invoked once the "list" is completed.

*callback\_arg* Argument to be passed to the `complete_callback`.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- `complete_callback` is GLOBUS\_NULL
- handle already has an operation in progress

**See also:**

`globus_ftp_client_register_read()` (p. 61)

**5.5.4.11** `globus_result_t globus_ftp_client_verbose_list (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Get a file listing from an FTP server.

This function starts a "LIST" transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling `globus_ftp_client_register_read()` (p. 61) to retrieve the data associated with this listing.

When all of the data associated with the listing is retrieved, and all of the data callbacks have been called, or if the list request is aborted, the `complete_callback` will be invoked with the final status of the list.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL to list. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*complete\_callback* Callback to be invoked once the "list" is completed.

*callback\_arg* Argument to be passed to the `complete_callback`.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsift url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**See also:**

`globus_ftp_client_register_read()` (p. 61)

**5.5.4.12** `globus_result_t globus_ftp_client_stat (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_byte_t ** stat_buffer, globus_size_t * stat_buffer_length, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Get information about a file or directory from a FTP server.

This function requests the STAT listing of a file or directory from an FTP server.

When the STAT request is completed or aborted, the complete\_callback will be invoked with the final status of the operation. If the callback returns without an error, the STAT fact string will be stored in the globus\_byte\_t \* pointed to by the stat\_buffer parameter to this function.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL of a file or directory to list. The URL may be an ftp or gsift URL.

*attr* Attributes for this file transfer.

*stat\_buffer* A pointer to a globus\_byte\_t \* to be allocated and filled with the STAT listing of the file, if it exists. Otherwise, the value pointed to by it is undefined. It is up to the user to free this memory.

*stat\_buffer\_length* A pointer to a globus\_size\_t to be filled with the length of the data in stat\_buffer.

*complete\_callback* Callback to be invoked once the STAT command is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsift url
- stat\_buffer is GLOBUS\_NULL
- stat\_buffer\_length is GLOBUS\_NULL
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.13** `globus_result_t globus_ftp_client_machine_list (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Get a machine parseable file listing from an FTP server.

This function starts a "MLSD" transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling `globus_ftp_client_register_read()` (p. 61) to retrieve the data associated with this listing.

When all of the data associated with the listing is retrieved, and all of the data callbacks have been called, or if the list request is aborted, the `complete_callback` will be invoked with the final status of the list.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL to list. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*complete\_callback* Callback to be invoked once the "list" is completed.

*callback\_arg* Argument to be passed to the `complete_callback`.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**See also:**

`globus_ftp_client_register_read()` (p. 61)

**5.5.4.14** `globus_result_t globus_ftp_client_recursive_list (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`

Get a machine parseable recursive file listing from an FTP server.

This function starts a "MLSR" transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling `globus_ftp_client_register_read()` (p. 61) to retrieve the data associated with this listing.

When all of the data associated with the listing is retrieved, and all of the data callbacks have been called, or if the list request is aborted, the `complete_callback` will be invoked with the final status of the list.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL to list. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*complete\_callback* Callback to be invoked once the "list" is completed.

*callback\_arg* Argument to be passed to the `complete_callback`.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsift url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**See also:**

**globus\_ftp\_client\_register\_read()** (p. 61)

**5.5.4.15** **globus\_result\_t globus\_ftp\_client\_mlst (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_byte\_t \*\* *mlst\_buffer*, globus\_size\_t \* *mlst\_buffer\_length*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Get information about a file or directory from a FTP server.

This function requests the MLST fact string of a file or directory from an FTP server.

When the MLST request is completed or aborted, the complete\_callback will be invoked with the final status of the operation. If the callback returns without an error, the MLST fact string will be stored in the globus\_byte\_t \* pointed to by the mlst\_buffer parameter to this function.

**Parameters:**

***u\_handle*** An FTP Client handle to use for the list operation.

***url*** The URL of a file or directory to list. The URL may be an ftp or gsift URL.

***attr*** Attributes for this file transfer.

***mlst\_buffer*** A pointer to a globus\_byte\_t \* to be allocated and filled with the MLST fact string of the file, if it exists. Otherwise, the value pointed to by it is undefined. It is up to the user to free this memory.

***mlst\_buffer\_length*** A pointer to a globus\_size\_t to be filled with the length of the data in mlst\_buffer.

***complete\_callback*** Callback to be invoked once the MLST command is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsift url
- mlst\_buffer is GLOBUS\_NULL
- mlst\_buffer\_length is GLOBUS\_NULL
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.16 globus\_result\_t globus\_ftp\_client\_move (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *source\_url*, const char \* *dest\_url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Move a file on an FTP server.

This function starts a move (rename) operation on an FTP server. Note that this function does not move files between FTP servers and that the host:port part of the destination url is ignored.

When the response to the move request has been received the *complete\_callback* will be invoked with the result of the move operation.

**Parameters:**

*u\_handle* An FTP Client handle to use for the move operation.

*source\_url* The URL for the file to move.

*dest\_url* The URL for the target of the move. The host:port part of this URL is ignored.

*attr* Attributes for this operation.

*complete\_callback* Callback to be invoked once the move is completed.

*callback\_arg* Argument to be passed to the *complete\_callback*.

**Returns:**

This function returns an error when any of these conditions are true:

- *handle* is GLOBUS\_NULL
- *source\_url* is GLOBUS\_NULL
- *source\_url* cannot be parsed
- *source\_url* is not a ftp or gsiftp url
- *complete\_callback* is GLOBUS\_NULL
- *handle* already has an operation in progress

**5.5.4.17 globus\_result\_t globus\_ftp\_client\_chmod (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, int *mode*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Change permissions on a file.

This function changes file permissions

When the response to the move request has been received the *complete\_callback* will be invoked with the result of the operation.

**Parameters:**

*u\_handle* An FTP Client handle to use for the move operation.

*url* The URL of the file to modify permissions.

*mode* The integer file mode to change to. Be sure that the integer is in the proper base, i.e. 0777 (octal) vs 777 (decimal).

*attr* Attributes for this operation.

*complete\_callback* Callback to be invoked once the move is completed.

*callback\_arg* Argument to be passed to the *complete\_callback*.

**Returns:**

This function returns an error when any of these conditions are true:



- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.18 globus\_result\_t globus\_ftp\_client\_chgrp (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, const char \* *group*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Change group membership on a file.

This function changes file group membership

When the response to the group change request has been received the complete\_callback will be invoked with the result of the operation.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the change operation.

***url*** The URL of the file to modify group membership.

***group*** The group name or numeric identifier for the group that the file should become a member of.

***attr*** Attributes for this operation.

***complete\_callback*** Callback to be invoked once the change is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.19 globus\_result\_t globus\_ftp\_client\_utime (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, const struct tm \* *utime\_time*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Set the modification time on a file.

This function changes file modification time

When the response to the modification time request has been received the complete\_callback will be invoked with the result of the operation.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the change operation.

***url*** The URL of the file to modify group membership.

***utime\_time*** The absolute time to which the file's modification time should be set.

***attr*** Attributes for this operation.

***complete\_callback*** Callback to be invoked once the change is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.20 globus\_result\_t globus\_ftp\_client\_symlink (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *source\_url*, const char \* *link\_url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Create a symbolic link on the FTP server.

This function creates a symbolic link on the FTP server pointing to a given path. Note that this function does not link files between FTP servers and that the host:port part of the link url is ignored.

When the response to the symlink request has been received the complete\_callback will be invoked with the result of the move operation.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the move operation.

***source\_url*** The URL for the symbolic link to create.

***link\_url*** The URL for the target of the link. The host:port part of this URL is ignored.

***attr*** Attributes for this operation.

***complete\_callback*** Callback to be invoked once the symlink is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- source\_url is GLOBUS\_NULL
- source\_url cannot be parsed
- source\_url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.21 globus\_result\_t globus\_ftp\_client\_get (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Get a file from an FTP server.

This function starts a "get" file transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_read()** (p. 61) to retrieve the data associated with this URL.

When all of the data associated with this URL is retrieved, and all of the data callbacks have been called, or if the get request is aborted, the *complete\_callback* will be invoked with the final status of the get.

**Parameters:**

*handle* An FTP Client handle to use for the get operation.

*url* The URL to download. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*restart* Pointer to a restart marker.

*complete\_callback* Callback to be invoked once the "get" is completed.

*callback\_arg* Argument to be passed to the *complete\_callback*.

**Returns:**

This function returns an error when any of these conditions are true:

- *handle* is GLOBUS\_NULL
- *url* is GLOBUS\_NULL
- *url* cannot be parsed
- *url* is not a ftp or gsiftp url
- *complete\_callback* is GLOBUS\_NULL
- *handle* already has an operation in progress

**See also:**

**globus\_ftp\_client\_register\_read()** (p. 61)

**5.5.4.22 globus\_result\_t globus\_ftp\_client\_partial\_get (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart*, globus\_off\_t *partial\_offset*, globus\_off\_t *partial\_end\_offset*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Get a file from an FTP server.

This function starts a "get" file transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_read()** (p. 61) to retrieve the data associated with this URL.

When all of the data associated with this URL is retrieved, and all of the data callbacks have been called, or if the get request is aborted, the *complete\_callback* will be invoked with the final status of the get.

**Parameters:**

*handle* An FTP Client handle to use for the get operation.

*url* The URL to download. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

***restart*** Pointer to a restart marker.

***partial\_offset*** Starting offset for a partial file get.

***partial\_end\_offset*** Ending offset for a partial file get. Use -1 for EOF.

***complete\_callback*** Callback to be invoked once the "get" is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.23 globus\_result\_t globus\_ftp\_client\_extended\_get (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart*, const char \* *eret\_alg\_str*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Get a file from an FTP server with server-side processing.

This function starts a "get" file transfer from an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_read()** (p. 61) to retrieve the data associated with this URL.

When all of the data associated with this URL is retrieved, and all of the data callbacks have been called, or if the get request is aborted, the complete\_callback will be invoked with the final status of the get.

This function differs from the **globus\_ftp\_client\_get()** (p. 34) function by allowing the user to invoke server-side data processing algorithms. GridFTP servers may support support algorithms for data reduction or other customized data storage requirements. There is no client-side verification done on the algorithm string provided by the user. If the server does not understand the requested algorithm, the transfer will fail.

#### Parameters:

***handle*** An FTP Client handle to use for the get operation.

***url*** The URL to download. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this file transfer.

***restart*** Pointer to a restart marker.

***eret\_alg\_str*** The ERET algorithm string. This string contains information needed to invoke a server-specific data reduction algorithm on the file being retrieved.

***complete\_callback*** Callback to be invoked once the "get" is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed

- url is not a ftp or gsiftp url
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

See also:

**globus\_ftp\_client\_register\_read()** (p. 61)

**5.5.4.24 globus\_result\_t globus\_ftp\_client\_put (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Store a file on an FTP server.

This function starts a "put" file transfer to an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_write()** (p. 61) to send the data associated with this URL.

When all of the data associated with this URL is sent, and all of the data callbacks have been called, or if the put request is aborted, the complete\_callback will be invoked with the final status of the put.

**Parameters:**

***handle*** An FTP Client handle to use for the put operation.

***url*** The URL to store the data to. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this file transfer.

***restart*** Pointer to a restart marker.

***complete\_callback*** Callback to be invoked once the "put" is completed.

***callback\_arg*** Argument to be passed to the complete\_callback.

See also:

**globus\_ftp\_client\_register\_write()** (p. 61)

**5.5.4.25 globus\_result\_t globus\_ftp\_client\_partial\_put (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart*, globus\_off\_t *partial\_offset*, globus\_off\_t *partial\_end\_offset*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)**

Store a file on an FTP server.

This function starts a "put" file transfer to an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_write()** (p. 61) to send the data associated with this URL.

When all of the data associated with this URL is sent, and all of the data callbacks have been called, or if the put request is aborted, the complete\_callback will be invoked with the final status of the put.

**Parameters:**

***handle*** An FTP Client handle to use for the put operation.

***url*** The URL to store the data to. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this file transfer.

***restart*** Pointer to a restart marker.

*partial\_offset* Starting offset for a partial file put.

*partial\_end\_offset* Ending offset for a partial file put.

*complete\_callback* Callback to be invoked once the "put" is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

See also:

**globus\_ftp\_client\_register\_write()** (p. 61)

**5.5.4.26 globus\_result\_t globus\_ftp\_client\_extended\_put (globus\_ftp\_client\_handle\_t \* handle, const char \* url, globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_client\_restart\_marker\_t \* restart, const char \* esto\_alg\_str, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \* callback\_arg)**

Store a file on an FTP server with server-side processing.

This function starts a "put" file transfer to an FTP server. If this function returns GLOBUS\_SUCCESS, then the user may immediately begin calling **globus\_ftp\_client\_register\_write()** (p. 61) to send the data associated with this URL.

When all of the data associated with this URL is sent, and all of the data callbacks have been called, or if the put request is aborted, the complete\_callback will be invoked with the final status of the put.

This function differs from the **globus\_ftp\_client\_put()** (p. 36) function by allowing the user to invoke server-side data processing algorithms. GridFTP servers may support algorithms for data reduction or other customized data storage requirements. There is no client-side verification done on the algo string provided by the user. If the server does not understand the requested algorithm, the transfer will fail.

**Parameters:**

*handle* An FTP Client handle to use for the put operation.

*url* The URL to store the data to. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*restart* Pointer to a restart marker.

*esto\_alg\_str*

*complete\_callback* Callback to be invoked once the "put" is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

See also:

**globus\_ftp\_client\_register\_write()** (p. 61)

**5.5.4.27 globus\_result\_t globus\_ftp\_client\_third\_party\_transfer (globus\_ftp\_client\_handle\_t \* handle, const char \* source\_url, globus\_ftp\_client\_operationattr\_t \* source\_attr, const char \* dest\_url, globus\_ftp\_client\_operationattr\_t \* dest\_attr, globus\_ftp\_client\_restart\_marker\_t \* restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \* callback\_arg)**

Transfer a file between two FTP servers.

This function starts up a third-party file transfer between FTP server. This function returns immediately.

When the transfer is completed or if the transfer is aborted, the complete\_callback will be invoked with the final status of the transfer.

**Parameters:**

*handle* An FTP Client handle to use for the get operation.

*source\_url* The URL to transfer. The URL may be an ftp or gsiftp URL.

*source\_attr* Attributes for the source URL.

*dest\_url* The destination URL for the transfer. The URL may be an ftp or gsiftp URL.

*dest\_attr* Attributes for the destination URL.

*restart* Pointer to a restart marker.

*complete\_callback* Callback to be invoked once the "put" is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

**Note:**

The source\_attr and dest\_attr MUST be compatible. For example, the MODE and TYPE should match for both the source and destination.

**5.5.4.28 globus\_result\_t globus\_ftp\_client\_partial\_third\_party\_transfer** (globus\_ftp\_client\_handle\_t \* handle, const char \* source\_url, globus\_ftp\_client\_operationattr\_t \* source\_attr, const char \* dest\_url, globus\_ftp\_client\_operationattr\_t \* dest\_attr, globus\_ftp\_client\_restart\_marker\_t \* restart, globus\_off\_t partial\_offset, globus\_off\_t partial\_end\_offset, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \* callback\_arg)

Transfer a file between two FTP servers.

This function starts up a third-party file transfer between FTP server. This function returns immediately.

When the transfer is completed or if the transfer is aborted, the complete\_callback will be invoked with the final status of the transfer.

**Parameters:**

*handle* An FTP Client handle to use for the get operation.

*source\_url* The URL to transfer. The URL may be an ftp or gsiftp URL.

*source\_attr* Attributes for the source URL.

*dest\_url* The destination URL for the transfer. The URL may be an ftp or gsiftp URL.

*dest\_attr* Attributes for the destination URL.

*restart* Pointer to a restart marker.

*partial\_offset* Starting offset for a partial file get.

*partial\_end\_offset* Ending offset for a partial file get. Use -1 for EOF.

*complete\_callback* Callback to be invoked once the "put" is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

**Note:**

The source\_attr and dest\_attr MUST be compatible. For example, the MODE and TYPE should match for both the source and destination.

**5.5.4.29 globus\_result\_t globus\_ftp\_client\_extended\_third\_party\_transfer** (globus\_ftp\_client\_handle\_t \* handle, const char \* source\_url, globus\_ftp\_client\_operationattr\_t \* source\_attr, const char \* eret\_alg\_str, const char \* dest\_url, globus\_ftp\_client\_operationattr\_t \* dest\_attr, const char \* esto\_alg\_str, globus\_ftp\_client\_restart\_marker\_t \* restart, globus\_ftp\_client\_complete\_callback\_t complete\_callback, void \* callback\_arg)

Transfer a file between two FTP servers with server-side processing.

This function starts up a third-party file transfer between FTP server. This function returns immediately.

When the transfer is completed or if the transfer is aborted, the `complete_callback` will be invoked with the final status of the transfer.

This function differs from the `globus_ftp_client_third_party_transfer()` (p. 37) function by allowing the user to invoke server-side data processing algorithms. GridFTP servers may support algorithms for data reduction or other customized data storage requirements. There is no client-side verification done on the algorithm string provided by the user. If the server does not understand the requested algorithm, the transfer will fail.

#### Parameters:

***handle*** An FTP Client handle to use for the get operation.

***source\_url*** The URL to transfer. The URL may be an ftp or gsiftp URL.

***source\_attr*** Attributes for the source URL.

***reset\_alg\_str***

***dest\_url*** The destination URL for the transfer. The URL may be an ftp or gsiftp URL.

***dest\_attr*** Attributes for the destination URL.

***reset\_alg\_str***

***restart*** Pointer to a restart marker.

***complete\_callback*** Callback to be invoked once the "put" is completed.

***callback\_arg*** Argument to be passed to the `complete_callback`.

#### Note:

The `source_attr` and `dest_attr` MUST be compatible. For example, the MODE and TYPE should match for both the source and destination.

**5.5.4.30 `globus_result_t globus_ftp_client_modification_time (globus_ftp_client_handle_t * u_handle, const char * url, globus_ftp_client_operationattr_t * attr, globus_abstime_t * modification_time, globus_ftp_client_complete_callback_t complete_callback, void * callback_arg)`**

Get a file's modification time from an FTP server.

This function requests the modification time of a file from an FTP server.

When the modification time request is completed or aborted, the `complete_callback` will be invoked with the final status of the operation. If the callback returns without an error, the modification time will be stored in the `globus_abstime_t` value pointed to by the `modification_time` parameter to this function.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the list operation.

***url*** The URL to list. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this file transfer.

***modification\_time*** A pointer to a `globus_abstime_t` to be filled with the modification time of the file, if it exists. Otherwise, the value pointed to by it is undefined.

***complete\_callback*** Callback to be invoked once the modification time check is completed.

***callback\_arg*** Argument to be passed to the `complete_callback`.

#### Returns:

This function returns an error when any of these conditions are true:

- `handle` is `GLOBUS_NULL`



- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- modification time is GLOBUS\_NULL
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.31 globus\_result\_t globus\_ftp\_client\_size** (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_off\_t \* *size*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)

Get a file's size from an FTP server.

This function requests the size of a file from an FTP server.

When the size request is completed or aborted, the complete\_callback will be invoked with the final status of the operation. If the callback returns without an error, the size will be stored in the globus\_off\_t value pointed to by the size parameter to this function.

**Note:**

In ASCII mode, the size will be the size of the file after conversion to ASCII mode. The actual amount of data which is returned in the data callbacks may be less than this amount.

**Parameters:**

*u\_handle* An FTP Client handle to use for the list operation.

*url* The URL to list. The URL may be an ftp or gsiftp URL.

*attr* Attributes for this file transfer.

*size* A pointer to a globus\_off\_t to be filled with the total size of the file, if it exists. Otherwise, the value pointed to by it is undefined.

*complete\_callback* Callback to be invoked once the size check is completed.

*callback\_arg* Argument to be passed to the complete\_callback.

**Returns:**

This function returns an error when any of these conditions are true:

- handle is GLOBUS\_NULL
- url is GLOBUS\_NULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- size is GLOBUS\_NULL
- complete\_callback is GLOBUS\_NULL
- handle already has an operation in progress

**5.5.4.32 globus\_result\_t globus\_ftp\_client\_cksm** (globus\_ftp\_client\_handle\_t \* *u\_handle*, const char \* *url*, globus\_ftp\_client\_operationattr\_t \* *attr*, char \* *cksm*, globus\_off\_t *offset*, globus\_off\_t *length*, const char \* *algorithm*, globus\_ftp\_client\_complete\_callback\_t *complete\_callback*, void \* *callback\_arg*)

Get a file's checksum from an FTP server.

This function requests the checksum of a file from an FTP server.

When the request is completed or aborted, the `complete_callback` will be invoked with the final status of the operation. If the callback returns without an error, the checksum will be stored in the `char *` buffer provided in the `'checksum'` parameter to this function. The buffer must be large enough to hold the expected checksum result.

#### Parameters:

***u\_handle*** An FTP Client handle to use for the list operation.

***url*** The URL to list. The URL may be an ftp or gsiftp URL.

***attr*** Attributes for this file transfer.

***cksm*** A pointer to a buffer to be filled with the checksum of the file. On error the buffer contents are undefined.

***offset*** File offset to start calculating checksum.

***length*** Length of data to read from the starting offset. Use -1 to read the entire file.

***algorithm*** A pointer to a string to specifying the desired algorithm. Currently, GridFTP servers only support the MD5 algorithm.

***complete\_callback*** Callback to be invoked once the checksum is completed.

***callback\_arg*** Argument to be passed to the `complete_callback`.

#### Returns:

This function returns an error when any of these conditions are true:

- handle is `GLOBUS_NULL`
- url is `GLOBUS_NULL`
- url cannot be parsed
- url is not a ftp or gsiftp url
- size is `GLOBUS_NULL`
- `complete_callback` is `GLOBUS_NULL`
- handle already has an operation in progress

#### 5.5.4.33 `globus_result_t globus_ftp_client_abort (globus_ftp_client_handle_t * u_handle)`

Abort the operation currently in progress.

#### Parameters:

***u\_handle*** Handle which to abort.

## 5.6 FTP Operation Attributes

Operation attributes are used to control the security and performance of an FTP operation.

#### Storage Module

- `globus_result_t globus_ftp_client_operationattr_set_storage_module (globus_ftp_client_operationattr_t *attr, const char *module_name, const char *module_args)`
- `globus_result_t globus_ftp_client_operationattr_get_storage_module (const globus_ftp_client_operationattr_t *attr, char **module_name, char **module_args)`

## Custom Data Channel Driver Stack

- `globus_result_t globus_ftp_client_operationattr_set_net_stack (globus_ftp_client_operationattr_t *attr, const char *driver_list)`
- `globus_result_t globus_ftp_client_operationattr_get_net_stack (const globus_ftp_client_operationattr_t *attr, char **driver_list)`

## Custom Server File Driver Stack

- `globus_result_t globus_ftp_client_operationattr_set_disk_stack (globus_ftp_client_operationattr_t *attr, const char *driver_list)`
- `globus_result_t globus_ftp_client_operationattr_get_disk_stack (const globus_ftp_client_operationattr_t *attr, char **driver_list)`

## Parallelism

- `globus_result_t globus_ftp_client_operationattr_set_parallelism (globus_ftp_client_operationattr_t *attr, const globus_ftp_control_parallelism_t *parallelism)`
- `globus_result_t globus_ftp_client_operationattr_get_parallelism (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_parallelism_t *parallelism)`

## allocate

- `globus_result_t globus_ftp_client_operationattr_set_allocate (globus_ftp_client_operationattr_t *attr, const globus_off_t allocated_size)`
- `globus_result_t globus_ftp_client_operationattr_get_allocate (const globus_ftp_client_operationattr_t *attr, globus_off_t *allocated_size)`

## authz\_assert

- `globus_result_t globus_ftp_client_operationattr_set_authz_assert (globus_ftp_client_operationattr_t *attr, const char *authz_assert, globus_bool_t cache_authz_assert)`
- `globus_result_t globus_ftp_client_operationattr_get_authz_assert (const globus_ftp_client_operationattr_t *attr, char **authz_assert, globus_bool_t *cache_authz_assert)`

## Striped Data Movement

- `globus_result_t globus_ftp_client_operationattr_set_striped (globus_ftp_client_operationattr_t *attr, globus_bool_t striped)`
- `globus_result_t globus_ftp_client_operationattr_get_striped (const globus_ftp_client_operationattr_t *attr, globus_bool_t *striped)`

## Striped File Layout

- `globus_result_t globus_ftp_client_operationattr_set_layout (globus_ftp_client_operationattr_t *attr, const globus_ftp_control_layout_t *layout)`
- `globus_result_t globus_ftp_client_operationattr_get_layout (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_layout_t *layout)`

## TCP Buffer

- `globus_result_t globus_ftp_client_operationattr_set_tcp_buffer (globus_ftp_client_operationattr_t *attr, const globus_ftp_control_tcpbuffer_t *tcp_buffer)`
- `globus_result_t globus_ftp_client_operationattr_get_tcp_buffer (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_tcpbuffer_t *tcp_buffer)`

## File Type

- `globus_result_t globus_ftp_client_operationattr_set_type (globus_ftp_client_operationattr_t *attr, globus_ftp_control_type_t type)`
- `globus_result_t globus_ftp_client_operationattr_get_type (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_type_t *type)`

## Transfer Mode

- `globus_result_t globus_ftp_client_operationattr_set_mode (globus_ftp_client_operationattr_t *attr, globus_ftp_control_mode_t mode)`
- `globus_result_t globus_ftp_client_operationattr_get_mode (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_mode_t *mode)`
- `globus_result_t globus_ftp_client_operationattr_set_list_uses_data_mode (const globus_ftp_client_operationattr_t *attr, globus_bool_t list_uses_data_mode)`
- `globus_result_t globus_ftp_client_operationattr_get_list_uses_data_mode (const globus_ftp_client_operationattr_t *attr, globus_bool_t *list_uses_data_mode)`

## Authorization

- `globus_result_t globus_ftp_client_operationattr_set_authorization (globus_ftp_client_operationattr_t *attr, gss_cred_id_t credential, const char *user, const char *password, const char *account, const char *subject)`
- `globus_result_t globus_ftp_client_operationattr_get_authorization (const globus_ftp_client_operationattr_t *attr, gss_cred_id_t *credential, char **user, char **password, char **account, char **subject)`

## Data Channel Authentication

- `globus_result_t globus_ftp_client_operationattr_set_dcau (globus_ftp_client_operationattr_t *attr, const globus_ftp_control_dcau_t *dcau)`
- `globus_result_t globus_ftp_client_operationattr_get_dcau (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_dcau_t *dcau)`

## Data Channel Protection

- `globus_result_t globus_ftp_client_operationattr_set_data_protection (globus_ftp_client_operationattr_t *attr, globus_ftp_control_protection_t protection)`
- `globus_result_t globus_ftp_client_operationattr_get_data_protection (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_protection_t *protection)`

## Data Channel Security Context

- `globus_result_t globus_ftp_client_operationattr_set_data_security (globus_ftp_client_operationattr_t *attr, int type, void *credential)`
- `globus_result_t globus_ftp_client_operationattr_get_data_security (const globus_ftp_client_operationattr_t *attr, int *type, void **credential)`

## Control Channel Protection

- `globus_result_t globus_ftp_client_operationattr_set_control_protection (globus_ftp_client_operationattr_t *attr, globus_ftp_control_protection_t protection)`
- `globus_result_t globus_ftp_client_operationattr_get_control_protection (const globus_ftp_client_operationattr_t *attr, globus_ftp_control_protection_t *protection)`

## Append

- `globus_result_t globus_ftp_client_operationattr_set_append (globus_ftp_client_operationattr_t *attr, globus_bool_t append)`
- `globus_result_t globus_ftp_client_operationattr_get_append (const globus_ftp_client_operationattr_t *attr, globus_bool_t *append)`

## IPv6

- `globus_result_t globus_ftp_client_operationattr_set_allow_ipv6 (globus_ftp_client_operationattr_t *attr, globus_bool_t allow_ipv6)`
- `globus_result_t globus_ftp_client_operationattr_get_allow_ipv6 (const globus_ftp_client_operationattr_t *attr, globus_bool_t *allow_ipv6)`

## Read into a Single Buffer

- `globus_result_t globus_ftp_client_operationattr_set_read_all (globus_ftp_client_operationattr_t *attr, globus_bool_t read_all, globus_ftp_client_data_callback_t intermediate_callback, void *intermediate_callback_arg)`
- `globus_result_t globus_ftp_client_operationattr_get_read_all (const globus_ftp_client_operationattr_t *attr, globus_bool_t *read_all, globus_ftp_client_data_callback_t *intermediate_callback, void **intermediate_callback_arg)`

## Typedefs

- `typedef globus_i_ftp_client_operationattr_t * globus_ftp_client_operationattr_t`

## Functions

- `globus_result_t globus_ftp_client_operationattr_init (globus_ftp_client_operationattr_t *attr)`
- `globus_result_t globus_ftp_client_operationattr_destroy (globus_ftp_client_operationattr_t *attr)`
- `globus_result_t globus_ftp_client_operationattr_set_delayed_pasv (const globus_ftp_client_operationattr_t *attr, globus_bool_t delayed_pasv)`
- `globus_result_t globus_ftp_client_operationattr_copy (globus_ftp_client_operationattr_t *dst, const globus_ftp_client_operationattr_t *src)`

### 5.6.1 Detailed Description

Operation attributes are used to control the security and performance of an FTP operation.

These features are often dependent on the capabilities of the FTP server which you are going to access.

### 5.6.2 Typedef Documentation

#### 5.6.2.1 typedef struct globus\_i\_ftp\_client\_operationattr\_t\* globus\_ftp\_client\_operationattr\_t

Operation Attributes.

FTP Client attributes are used to control the parameters needed to access an URL using the FTP protocol. Attributes are created and manipulated using the functions in the **attributes** (p. 41) section of the library.

See also:

**globus\_ftp\_client\_operationattr\_init()** (p. 45), **globus\_ftp\_client\_operationattr\_destroy()** (p. 45)

### 5.6.3 Function Documentation

#### 5.6.3.1 globus\_result\_t globus\_ftp\_client\_operationattr\_init (globus\_ftp\_client\_operationattr\_t \* *attr*)

Initialize an FTP client attribute set.

**Parameters:**

*attr* A pointer to the new attribute set.

#### 5.6.3.2 globus\_result\_t globus\_ftp\_client\_operationattr\_destroy (globus\_ftp\_client\_operationattr\_t \* *attr*)

Destroy an FTP client attribute set.

**Parameters:**

*attr* A pointer to the attribute to destroy.

#### 5.6.3.3 globus\_result\_t globus\_ftp\_client\_operationattr\_set\_storage\_module (globus\_ftp\_client\_operationattr\_t \* *attr*, const char \* *module\_name*, const char \* *module\_args*)

Set/Get the gridftp storage module (DSI).

This attribute allows the user to control what backend module they use with the gridftp server. The module **MUST** be implemented by the server or the transfer/get/put will result in an error.

This attribute is ignored in stream mode.

**Parameters:**

*attr* The attribute set to query or modify.

*module\_name* The backend storage module name

*module\_args* The backend storage module parameters

**Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.

#### 5.6.3.4 `globus_result_t globus_ftp_client_operationattr_set_net_stack (globus_ftp_client_operationattr_t * attr, const char * driver_list)`

Set/Get the gridftp xio driver stack used for the data channel.

This attribute allows the user to control which xio drivers will be used for data transport. The driver MUST be installed and allowed by the server or the transfer/get/put will result in an error.

##### Parameters:

*attr* The attribute set to query or modify.

*driver\_list* driver list in the following format: driver1[:driver1opts][,driver2[:driver2opts]][...]. The string "default" will reset the stack list to the server default.

##### Note:

This is a GridFTP extension, and may not be supported on all FTP servers.

#### 5.6.3.5 `globus_result_t globus_ftp_client_operationattr_set_disk_stack (globus_ftp_client_operationattr_t * attr, const char * driver_list)`

Set/Get the gridftp xio driver stack used for the file storage.

This attribute allows the user to control which xio drivers will be used for file DSI only. This is an experimental feature of the gridftp server. Only works for third party transfers.

##### Parameters:

*attr* The attribute set to query or modify.

*driver\_list* driver list in the following format: driver1[:driver1opts][,driver2[:driver2opts]][...]. The string "default" will reset the stack list to the server default.

##### Note:

This is a GridFTP extension, and may not be supported on all FTP servers.

#### 5.6.3.6 `globus_result_t globus_ftp_client_operationattr_set_parallelism (globus_ftp_client_operationattr_t * attr, const globus_ftp_control_parallelism_t * parallelism)`

Set/Get the parallelism attribute for an ftp client attribute set.

This attribute allows the user to control the level of parallelism to be used on an extended block mode file transfer. Currently, only a "fixed" parallelism level is supported. This is interpreted by the FTP server as the number of parallel data connections to be allowed for each stripe of data. Currently, only the "fixed" parallelism type is

This attribute is ignored in stream mode.

##### Parameters:

*attr* The attribute set to query or modify.

*parallelism* The value of parallelism attribute.

##### See also:

`globus_ftp_client_operationattr_set_layout()` (p. 48), `globus_ftp_client_operationattr_set_mode()` (p. 49)

##### Note:

This is a Grid-FTP extension, and may not be supported on all FTP servers.

#### 5.6.3.7 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_allocate (globus\_ftp\_client\_operationattr\_t \* attr, const globus\_off\_t allocated\_size)**

Set/Get the allocate attribute for an ftp client attribute set.

This attribute lets the user set a size to be passed to the server before a put operation.

This attribute is ignored for get operations.

##### **Parameters:**

*attr* The attribute set to query or modify.

*allocated\_size* The size to direct server to allocate.

#### 5.6.3.8 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_authz\_assert (globus\_ftp\_client\_operationattr\_t \* attr, const char \* authz\_assert, globus\_bool\_t cache\_authz\_assert)**

Set/Get the authz\_assert attribute for an ftp client attribute set.

This attribute lets the user set an AUTHORIZATION assertion to be passed to the server

##### **Parameters:**

*attr* The attribute set to query or modify.

*authz\_assert* The AUTHORIZATION assertion.

*cache\_authz\_assert* Boolean that specifies whether to cache this assertion for subsequent operations

#### 5.6.3.9 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_striped (globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t striped)**

Set/Get the striped attribute for an ftp client attribute set.

This attribute allows the user to force the client library to use the FTP commands to do a striped data transfer, even when the user has not requested a specific file layout via the layout attribute. This is useful when transferring files between servers which use the server side processing commands ERET or ESTO to transform data and send it to particular stripes on the destination server.

The layout attribute is used only when the data is being stored the server (on a put or 3rd party transfer). This attribute is ignored for stream mode data transfers.

##### **Parameters:**

*attr* The attribute set to query or modify.

*striped* The value of striped attribute.

##### **See also:**

**globus\_ftp\_client\_operationattr\_set\_parallelism()** (p. 46), **globus\_ftp\_client\_operationattr\_set\_layout()** (p. 48) **globus\_ftp\_client\_operationattr\_set\_mode()** (p. 49)

##### **Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.



#### 5.6.3.10 `globus_result_t globus_ftp_client_operationattr_set_layout (globus_ftp_client_operationattr_t * attr, const globus_ftp_control_layout_t * layout)`

Set/Get the layout attribute for an ftp client attribute set.

This attribute allows the user to control the layout of a file being transferred to a striped Grid-FTP server. The striping layout defines what regions of a file will be stored on each stripe of a multiple-striped ftp server.

The layout attribute is used only when the data is being stored on the server (on a put or 3rd party transfer). This attribute is ignored for stream mode data transfers.

##### Parameters:

*attr* The attribute set to query or modify.

*layout* The value of layout attribute.

##### See also:

`globus_ftp_client_operationattr_set_parallelism()` (p. 46), `globus_ftp_client_operationattr_set_mode()` (p. 49)

##### Note:

This is a Grid-FTP extension, and may not be supported on all FTP servers.

#### 5.6.3.11 `globus_result_t globus_ftp_client_operationattr_set_tcp_buffer (globus_ftp_client_operationattr_t * attr, const globus_ftp_control_tcpbuffer_t * tcp_buffer)`

Set/Get the TCP buffer attribute for an ftp client attribute set.

This attribute allows the user to control the TCP buffer size used for all data channels used in a file transfer. The size of the TCP buffer can make a significant impact on the performance of a file transfer. The user may set the buffer to either a system-dependent default value, or to a fixed value.

The actual implementation of this attribute is designed to be as widely interoperable as possible. In addition to supporting the SBUF command described in the GridFTP protocol extensions document, it also supports other commands and site commands which are used by other servers to set TCP buffer sizes. These are

- SITE RETRBUFSIZE
- SITE RBUFSZ
- SITE RBUFSIZ
- SITE STORBUFSIZE
- SITE SBUFSZ
- SITE SBUFSIZ
- SITE BUFSIZE

This attribute affects any type of data transfer done with the ftp client library.

##### Parameters:

*attr* The attribute set to query or modify.

*tcp\_buffer* The value of tcp\_buffer attribute.

#### 5.6.3.12 `globus_result_t globus_ftp_client_operationattr_set_type (globus_ftp_client_operationattr_t * attr, globus_ftp_control_type_t type)`

Set/Get the file representation type attribute for an ftp client attribute set.

This attribute allows the user to choose the file type used for an FTP file transfer. The file may be transferred as either ASCII or a binary image.

When transferring an ASCII file, the data will be transformed in the following way

- the high-order bit will be set to zero
- end-of line characters will be converted to a CRLF pair for the data transfer, and then converted to native format before being returned to the user's data callbacks.

The default type for the ftp client library is binary.

##### Parameters:

*attr* The attribute set to query or modify.

*type* The value of type attribute.

#### 5.6.3.13 `globus_result_t globus_ftp_client_operationattr_set_mode (globus_ftp_client_operationattr_t * attr, globus_ftp_control_mode_t mode)`

Set/Get the file transfer mode attribute for an ftp client attribute set.

This attribute allows the user to choose the data channel protocol used to transfer a file. There are two modes supported by this library: stream mode and extended block mode.

Stream mode is a file transfer mode where all data is sent over a single TCP socket, without any data framing. In stream mode, data will arrive in sequential order. This mode is supported by nearly all FTP servers.

Extended block mode is a file transfer mode where data can be sent over multiple parallel connections and to multiple data storage nodes to provide a high-performance data transfer. In extended block mode, data may arrive out-of-order. ASCII type files are not supported in extended block mode.

##### Parameters:

*attr* The attribute set to query or modify.

*mode* The value of mode attribute

##### See also:

`globus_ftp_client_operationattr_set_parallelism()` (p. 46), `globus_ftp_client_operationattr_set_layout()` (p. 48)

##### Note:

Extended block mode is a Grid-FTP extension, and may not be supported on all FTP servers.

#### 5.6.3.14 `globus_result_t globus_ftp_client_operationattr_set_list_uses_data_mode (const globus_ftp_client_operationattr_t * attr, globus_bool_t list_uses_data_mode)`

Set/Get whether or not list data will use the current data mode

This attribute allows the user to allow list data to be transferred using the current data channel mode.

##### Parameters:

*attr* The attribute set to query or modify.

*list\_uses\_data\_mode* globus\_bool\_t

**Note:**

List data transfers in nonstandard modes is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.15** *globus\_result\_t globus\_ftp\_client\_operationattr\_set\_delayed\_pasv (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t delayed\_pasv)*

Set/Get whether or not delayed passive should be used

This attribute allows the user to enable delayed passive so the server can provide the passive address after it knows the filename to be transferred.

**Parameters:**

*attr* The attribute set to query or modify.

*delayed\_pasv* globus\_bool\_t

**Note:**

Delayed passive is a GridFTP extension, and may not be supported on all FTP servers.

**5.6.3.16** *globus\_result\_t globus\_ftp\_client\_operationattr\_set\_authorization (globus\_ftp\_client\_operationattr\_t \* attr, gss\_cred\_id\_t credential, const char \* user, const char \* password, const char \* account, const char \* subject)*

Set/Get the authorization attribute for an ftp client attribute set.

This attribute allows the user to pass authentication information to the ftp client library. This information is used to authenticate with the ftp server.

The Globus FTP client library supports authentication using either the GSSAPI, or standard plaintext username and passwords. The type of authentication is determined by the URL scheme which is used for the individual get, put, or 3rd party transfer calls.

**Parameters:**

*attr* The attribute set to query or modify.

*credential* The credential to use for authenticating with a GSIFTP server. This may be GSS\_C\_NO\_CREDENTIAL to use the default credential.

*user* The user name to send to the FTP server. When doing a gsiftp transfer, this may be set to NULL, and the default globusmap entry for the user's GSI identity will be used.

*password* The password to send to the FTP server. When doing a gsiftp transfer, this may be set to NULL.

*account* The account to use for the data transfer. Most FTP servers do not require this.

*subject* The subject name of the FTP server. This is only used when doing a gsiftp transfer, and then only when the security subject name does not match the hostname of the server (ie, when the server is being run by a user).

**5.6.3.17** *globus\_result\_t globus\_ftp\_client\_operationattr\_set\_dcac (globus\_ftp\_client\_operationattr\_t \* attr, const globus\_ftp\_control\_dcac\_t \* dcac)*

Set/Get the data channel authentication attribute for an ftp client attribute set.

Data channel authentication is a GridFTP extension, and may not be supported by all servers. If a server supports it, then the default is to delegate a credential to the server, and authenticate all data channels with that delegated credential.

**Parameters:**

*attr* The attribute set to query or modify.  
*dcau* The value of data channel authentication attribute.

#### 5.6.3.18 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_data\_protection (globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_control\_protection\_t *protection*)**

Set/Get the data channel protection attribute for an ftp client attribute set.

**Parameters:**

*attr* The attribute set to query or modify.  
*protection* The value of data channel protection attribute.

**Bug**

Only safe and private protection levels are supported by gsiftp.

#### 5.6.3.19 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_data\_security (globus\_ftp\_client\_operationattr\_t \* *attr*, int *type*, void \* *credential*)**

Set/Get the data channel security context type and credential.

**Parameters:**

*attr* The attribute set to query or modify.  
*type* The type of credential. Currently only 'P' is supported, and requires a gss\_cred\_id\_t for the credential.  
*credential* A credential appropriate for the type specified.

#### 5.6.3.20 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_control\_protection (globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_control\_protection\_t *protection*)**

Set/Get the control channel protection attribute for an ftp client attribute set.

The control channel protection attribute allows the user to decide whether to encrypt or integrity check the command session between the client and the FTP server. This attribute is only relevant if used with a gsiftp URL.

**Parameters:**

*attr* The attribute set to query or modify.  
*protection* The value of control channel protection attribute.

**Bug**

The clear and safe protection levels are treated identically, with the client integrity checking all commands.  
The confidential and private protection levels are treated identically, with the client encrypting all commands.

#### 5.6.3.21 **globus\_result\_t globus\_ftp\_client\_operationattr\_set\_append (globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_bool\_t *append*)**

Set/Get the append attribute for an ftp client attribute set.

This attribute allows the user to append to a file on an FTP server, instead of replacing the existing file when doing a **globus\_ftp\_client\_put()** (p. 36) or **globus\_ftp\_client\_transfer()**.

This attribute is ignored on the retrieving side of a transfer, or a **globus\_ftp\_client\_get()** (p. 34).

**Parameters:**

*attr* The attribute set to query or modify.

*append* The value of append attribute.

**5.6.3.22 globus\_result\_t globus\_ftp\_client\_operationattr\_set\_allow\_ipv6 (globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t allow\_ipv6)**

Set/Get the allow ipv6 attribute for an ftp client attribute set.

This attribute allows client library to make use of ipv6 when possible.

Use of this is currently very experimental.

**Parameters:**

*attr* The attribute set to query or modify.

*allow\_ipv6* GLOBUS\_TRUE to allow ipv6 or GLOBUS\_FALSE to disallow(default)

**5.6.3.23 globus\_result\_t globus\_ftp\_client\_operationattr\_set\_read\_all (globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t read\_all, globus\_ftp\_client\_data\_callback\_t intermediate\_callback, void \* intermediate\_callback\_arg)**

Set/Get the read\_all attribute for an ftp client attribute set.

This attribute allows the user to pass in a single buffer to receive all of the data for the current transfer. This buffer must be large enough to hold all of the data for the transfer. Only one buffer may be registered with **globus\_ftp\_client\_register\_read()** (p. 61) when this attribute is used for a get.

In extended block mode, this attribute will cause data to be stored directly into the buffer from multiple streams without any extra data copies.

If the user sets the intermediate callback to a non-null value, this function will be called whenever an intermediate sub-section of the data is received into the buffer.

This attribute is ignored for **globus\_ftp\_client\_put()** (p. 36) or **globus\_ftp\_client\_third\_party\_transfer()** (p. 37) operations.

**Parameters:**

*attr* The attribute set to query or modify.

*read\_all* The value of read\_all attribute.

*intermediate\_callback* Callback to be invoked when a subsection of the data has been retrieved. This callback may be GLOBUS\_NULL, if the user only wants to be notified when the data transfer is completed.

*intermediate\_callback\_arg* User data to be passed to the intermediate callback function.

**5.6.3.24 globus\_result\_t globus\_ftp\_client\_operationattr\_copy (globus\_ftp\_client\_operationattr\_t \* dst, const globus\_ftp\_client\_operationattr\_t \* src)**

Create a duplicate of an attribute set.

The duplicated attribute set has a deep copy of all data in the attribute set, so the original may be destroyed, while the copy is still valid.

**Parameters:**

*dst* The attribute set to be initialized to the same values as src.

*src* The original attribute set to duplicate.

#### 5.6.3.25 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_storage\_module** (const globus\_ftp\_client\_operationattr\_t \* *attr*, char \*\* *module\_name*, char \*\* *module\_args*)

Set/Get the gridftp storage module (DSI).

This attribute allows the user to control what backend module they use with the gridftp server. The module **MUST** be implemented by the server or the transfer/get/put will result in an error.

This attribute is ignored in stream mode.

##### **Parameters:**

*attr* The attribute set to query or modify.

*module\_name* The backend storage module name

*module\_args* The backend storage module parameters

##### **Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.

#### 5.6.3.26 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_net\_stack** (const globus\_ftp\_client\_operationattr\_t \* *attr*, char \*\* *driver\_list*)

Set/Get the gridftp xio driver stack used for the data channel.

This attribute allows the user to control which xio drivers will be used for data transport. The driver **MUST** be installed and allowed by the server or the transfer/get/put will result in an error.

##### **Parameters:**

*attr* The attribute set to query or modify.

*driver\_list* driver list in the following format: driver1[:driver1opts][,driver2[:driver2opts]][...]. The string "default" will reset the stack list to the server default.

##### **Note:**

This is a GridFTP extension, and may not be supported on all FTP servers.

#### 5.6.3.27 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_disk\_stack** (const globus\_ftp\_client\_operationattr\_t \* *attr*, char \*\* *driver\_list*)

Set/Get the gridftp xio driver stack used for the file storage.

This attribute allows the user to control which xio drivers will be used for file DSI only. This is an experimental feature of the gridftp server. Only works for third party transfers.

##### **Parameters:**

*attr* The attribute set to query or modify.

*driver\_list* driver list in the following format: driver1[:driver1opts][,driver2[:driver2opts]][...]. The string "default" will reset the stack list to the server default.

##### **Note:**

This is a GridFTP extension, and may not be supported on all FTP servers.

**5.6.3.28 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_parallelism (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_parallelism\_t \* parallelism)**

Set/Get the parallelism attribute for an ftp client attribute set.

This attribute allows the user to control the level of parallelism to be used on an extended block mode file transfer. Currently, only a "fixed" parallelism level is supported. This is interpreted by the FTP server as the number of parallel data connections to be allowed for each stripe of data. Currently, only the "fixed" parallelism type is

This attribute is ignored in stream mode.

**Parameters:**

*attr* The attribute set to query or modify.

*parallelism* The value of parallelism attribute.

**See also:**

**globus\_ftp\_client\_operationattr\_set\_layout()** (p. 48), **globus\_ftp\_client\_operationattr\_set\_mode()** (p. 49)

**Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.29 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_allocate (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_off\_t \* allocated\_size)**

Set/Get the allocate attribute for an ftp client attribute set.

This attribute lets the user set a size to be passed to the server before a put operation.

This attribute is ignored for get operations.

**Parameters:**

*attr* The attribute set to query or modify.

*allocated\_size* The size to direct server to allocate.

**5.6.3.30 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_authz\_assert (const globus\_ftp\_client\_operationattr\_t \* attr, char \*\* authz\_assert, globus\_bool\_t \* cache\_authz\_assert)**

Set/Get the authz\_assert attribute for an ftp client attribute set.

This attribute lets the user set an AUTHORIZATION assertion to be passed to the server

**Parameters:**

*attr* The attribute set to query or modify.

*authz\_assert* The AUTHORIZATION assertion.

*cache\_authz\_assert* Boolean that specifies whether to cache this assertion for subsequent operations

**5.6.3.31 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_striped (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t \* striped)**

Set/Get the striped attribute for an ftp client attribute set.

This attribute allows the user to force the client library to use the FTP commands to do a striped data transfer, even when the user has not requested a specific file layout via the layout attribute. This is useful when transferring files between servers which use the server side processing commands ERET or ESTO to transform data and send it to particular stripes on the destination server.

The layout attribute is used only when the data is being stored on the server (on a put or 3rd party transfer). This attribute is ignored for stream mode data transfers.

**Parameters:**

*attr* The attribute set to query or modify.

*striped* The value of striped attribute.

**See also:**

`globus_ftp_client_operationattr_set_parallelism()` (p. 46), `globus_ftp_client_operationattr_set_layout()` (p. 48) `globus_ftp_client_operationattr_set_mode()` (p. 49)

**Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.32 `globus_result_t globus_ftp_client_operationattr_get_layout (const globus_ftp_client_operationattr_t * attr, globus_ftp_control_layout_t * layout)`**

Set/Get the layout attribute for an ftp client attribute set.

This attribute allows the user to control the layout of a file being transferred to a striped Grid-FTP server. The striping layout defines what regions of a file will be stored on each stripe of a multiple-striped ftp server.

The layout attribute is used only when the data is being stored on the server (on a put or 3rd party transfer). This attribute is ignored for stream mode data transfers.

**Parameters:**

*attr* The attribute set to query or modify.

*layout* The value of layout attribute.

**See also:**

`globus_ftp_client_operationattr_set_parallelism()` (p. 46), `globus_ftp_client_operationattr_set_mode()` (p. 49)

**Note:**

This is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.33 `globus_result_t globus_ftp_client_operationattr_get_tcp_buffer (const globus_ftp_client_operationattr_t * attr, globus_ftp_control_tcpbuffer_t * tcp_buffer)`**

Set/Get the TCP buffer attribute for an ftp client attribute set.

This attribute allows the user to control the TCP buffer size used for all data channels used in a file transfer. The size of the TCP buffer can make a significant impact on the performance of a file transfer. The user may set the buffer to either a system-dependent default value, or to a fixed value.

The actual implementation of this attribute is designed to be as widely interoperable as possible. In addition to supporting the SBUF command described in the GridFTP protocol extensions document, it also supports other commands and site commands which are used by other servers to set TCP buffer sizes. These are



- SITE RETRBUFSIZE
- SITE RBUFSZ
- SITE RBUFSIZ
- SITE STORBUFSIZE
- SITE SBUFSZ
- SITE SBUFSIZ
- SITE BUFSIZE

This attribute affects any type of data transfer done with the ftp client library.

**Parameters:**

*attr* The attribute set to query or modify.

*tcp\_buffer* The value of tcp\_buffer attribute.

#### 5.6.3.34 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_type (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_type\_t \* type)**

Set/Get the file representation type attribute for an ftp client attribute set.

This attribute allows the user to choose the file type used for an FTP file transfer. The file may be transferred as either ASCII or a binary image.

When transferring an ASCII file, the data will be transformed in the following way

- the high-order bit will be set to zero
- end-of line characters will be converted to a CRLF pair for the data transfer, and then converted to native format before being returned to the user's data callbacks.

The default type for the ftp client library is binary.

**Parameters:**

*attr* The attribute set to query or modify.

*type* The value of type attribute.

#### 5.6.3.35 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_mode (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_mode\_t \* mode)**

Set/Get the file transfer mode attribute for an ftp client attribute set.

This attribute allows the user to choose the data channel protocol used to transfer a file. There are two modes supported by this library: stream mode and extended block mode.

Stream mode is a file transfer mode where all data is sent over a single TCP socket, without any data framing. In stream mode, data will arrive in sequential order. This mode is supported by nearly all FTP servers.

Extended block mode is a file transfer mode where data can be sent over multiple parallel connections and to multiple data storage nodes to provide a high-performance data transfer. In extended block mode, data may arrive out-of-order. ASCII type files are not supported in extended block mode.

**Parameters:**

*attr* The attribute set to query or modify.

*mode* The value of mode attribute

**See also:**

**globus\_ftp\_client\_operationattr\_set\_parallelism()** (p. 46), **globus\_ftp\_client\_operationattr\_set\_layout()** (p. 48)

**Note:**

Extended block mode is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.36 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_list\_uses\_data\_mode (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t \* list\_uses\_data\_mode)**

Set/Get whether or not list data will use the current data mode

This attribute allows the user to allow list data to be transferred using the current data channel mode.

**Parameters:**

*attr* The attribute set to query or modify.

*list\_uses\_data\_mode* globus\_bool\_t

**Note:**

List data transfers in nonstandard modes is a Grid-FTP extension, and may not be supported on all FTP servers.

**5.6.3.37 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_authorization (const globus\_ftp\_client\_operationattr\_t \* attr, gss\_cred\_id\_t \* credential, char \*\* user, char \*\* password, char \*\* account, char \*\* subject)**

Set/Get the authorization attribute for an ftp client attribute set.

This attribute allows the user to pass authentication information to the ftp client library. This information is used to authenticate with the ftp server.

The Globus FTP client library supports authentication using either the GSSAPI, or standard plaintext username and passwords. The type of authentication is determined by the URL scheme which is used for the individual get, put, or 3rd party transfer calls.

**Parameters:**

*attr* The attribute set to query or modify.

*credential* The credential to use for authenticating with a GSIFTP server. This may be GSS\_C\_NO\_CREDENTIAL to use the default credential.

*user* The user name to send to the FTP server. When doing a gsiftp transfer, this may be set to NULL, and the default globusmap entry for the user's GSI identity will be used.

*password* The password to send to the FTP server. When doing a gsiftp transfer, this may be set to NULL.

*account* The account to use for the data transfer. Most FTP servers do not require this.

*subject* The subject name of the FTP server. This is only used when doing a gsiftp transfer, and then only when the security subject name does not match the hostname of the server (ie, when the server is being run by a user).

**5.6.3.38 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_dcau (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_dcau\_t \* dcau)**

Set/Get the data channel authentication attribute for an ftp client attribute set.

Data channel authentication is a GridFTP extension, and may not be supported by all servers. If a server supports it, then the default is to delegate a credential to the server, and authenticate all data channels with that delegated credential.

**Parameters:**

*attr* The attribute set to query or modify.

*dcau* The value of data channel authentication attribute.

**5.6.3.39 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_data\_protection (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_protection\_t \* protection)**

Set/Get the data channel protection attribute for an ftp client attribute set.

**Parameters:**

*attr* The attribute set to query or modify.

*protection* The value of data channel protection attribute.

**Bug**

Only safe and private protection levels are supported by gsiftp.

**5.6.3.40 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_data\_security (const globus\_ftp\_client\_operationattr\_t \* attr, int \* type, void \*\* credential)**

Set/Get the data channel security context type and credential.

**Parameters:**

*attr* The attribute set to query or modify.

*type* The type of credential. Currently only 'P' is supported, and requires a gss\_cred\_id\_t for the credential.

*credential* A credential appropriate for the type specified.

**5.6.3.41 globus\_result\_t globus\_ftp\_client\_operationattr\_get\_control\_protection (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_ftp\_control\_protection\_t \* protection)**

Set/Get the control channel protection attribute for an ftp client attribute set.

The control channel protection attribute allows the user to decide whether to encrypt or integrity check the command session between the client and the FTP server. This attribute is only relevant if used with a gsiftp URL.

**Parameters:**

*attr* The attribute set to query or modify.

*protection* The value of control channel protection attribute.

**Bug**

The clear and safe protection levels are treated identically, with the client integrity checking all commands. The confidential and private protection levels are treated identically, with the client encrypting all commands.

#### 5.6.3.42 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_append (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t \* append)**

Set/Get the append attribute for an ftp client attribute set.

This attribute allows the user to append to a file on an FTP server, instead of replacing the existing file when doing a **globus\_ftp\_client\_put()** (p. 36) or **globus\_ftp\_client\_transfer()**.

This attribute is ignored on the retrieving side of a transfer, or a **globus\_ftp\_client\_get()** (p. 34).

##### **Parameters:**

*attr* The attribute set to query or modify.

*append* The value of append attribute.

#### 5.6.3.43 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_allow\_ipv6 (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t \* allow\_ipv6)**

Set/Get the allow ipv6 attribute for an ftp client attribute set.

This attribute allows client library to make use of ipv6 when possible.

Use of this is currently very experimental.

##### **Parameters:**

*attr* The attribute set to query or modify.

*allow\_ipv6* GLOBUS\_TRUE to allow ipv6 or GLOBUS\_FALSE to disallow(default)

#### 5.6.3.44 **globus\_result\_t globus\_ftp\_client\_operationattr\_get\_read\_all (const globus\_ftp\_client\_operationattr\_t \* attr, globus\_bool\_t \* read\_all, globus\_ftp\_client\_data\_callback\_t \* intermediate\_callback, void \*\* intermediate\_callback\_arg)**

Set/Get the read\_all attribute for an ftp client attribute set.

This attribute allows the user to pass in a single buffer to receive all of the data for the current transfer. This buffer must be large enough to hold all of the data for the transfer. Only one buffer may be registered with **globus\_ftp\_client\_register\_read()** (p. 61) when this attribute is used for a get.

In extended block mode, this attribute will cause data to be stored directly into the buffer from multiple streams without any extra data copies.

If the user sets the intermediate callback to a non-null value, this function will be called whenever an intermediate sub-section of the data is received into the buffer.

This attribute is ignored for **globus\_ftp\_client\_put()** (p. 36) or **globus\_ftp\_client\_third\_party\_transfer()** (p. 37) operations.

##### **Parameters:**

*attr* The attribute set to query or modify.

*read\_all* The value of read\_all attribute.

*intermediate\_callback* Callback to be invoked when a subsection of the data has been retrieved. This callback may be GLOBUS\_NULL, if the user only wants to be notified when the data transfer is completed.

*intermediate\_callback\_arg* User data to be passed to the intermediate callback function.

## **5.7 Reading and Writing Data**

Certain FTP client operations require the user to supply buffers for reading or writing data to an FTP server.

## Typedefs

- typedef void(\*) **globus\_ftp\_client\_data\_callback\_t** (void \*user\_arg, globus\_ftp\_client\_handle\_t \*handle, globus\_object\_t \*error, globus\_byte\_t \*buffer, globus\_size\_t length, globus\_off\_t offset, globus\_bool\_t eof)

## Functions

- globus\_result\_t **globus\_ftp\_client\_register\_read** (globus\_ftp\_client\_handle\_t \*handle, globus\_byte\_t \*buffer, globus\_size\_t buffer\_length, globus\_ftp\_client\_data\_callback\_t callback, void \*callback\_arg)
- globus\_result\_t **globus\_ftp\_client\_register\_write** (globus\_ftp\_client\_handle\_t \*handle, globus\_byte\_t \*buffer, globus\_size\_t buffer\_length, globus\_off\_t offset, globus\_bool\_t eof, globus\_ftp\_client\_data\_callback\_t callback, void \*callback\_arg)

### 5.7.1 Detailed Description

Certain FTP client operations require the user to supply buffers for reading or writing data to an FTP server.

These operations are **globus\_ftp\_client\_get()** (p. 34), **globus\_ftp\_client\_partial\_get()** (p. 34), **globus\_ftp\_client\_put()** (p. 36), **globus\_ftp\_client\_partial\_put()** (p. 36), **globus\_ftp\_client\_list()** (p. 27), **globus\_ftp\_client\_machine\_list()** (p. 29), **globus\_ftp\_client\_recursive\_list()** (p. 29), and **globus\_ftp\_client\_verbose\_list()** (p. 27).

When doing these operations, the user must pass data buffers to the FTP Client library. Data is read or written directly from the data buffers, without any internal copies being done.

The functions in this section of the manual may be called as soon as the operation function has returned. Multiple data blocks may be registered with the FTP Client Library at once, and may be sent in parallel to or from the FTP server if the GridFTP protocol extensions are being used.

### 5.7.2 Typedef Documentation

**5.7.2.1 typedef void(\*) globus\_ftp\_client\_data\_callback\_t(void \*user\_arg, globus\_ftp\_client\_handle\_t \*handle, globus\_object\_t \*error, globus\_byte\_t \*buffer, globus\_size\_t length, globus\_off\_t offset, globus\_bool\_t eof)**

Data Callback.

Each read or write operation in the FTP Client library is asynchronous. A callback of this type is passed to each of the data operation function calls to let the user know when the data block has been processed.

#### Parameters:

**user\_arg** The user\_arg parameter passed to the read or write function.

**handle** The handle on which the data block operation was done.

**error** A Globus error object indicating any problem which occurred processing this data block, or or GLOBUS\_SUCCESS if the operation completed successfully.

**buffer** The data buffer passed to the original read or write call.

**length** The amount of data in the data buffer. When reading data, this may be smaller than original buffer's length.

**offset** The offset into the file which this data block contains.

**eof** GLOBUS\_TRUE if EOF has been reached on this data transfer, GLOBUS\_FALSE otherwise. This may be set to GLOBUS\_TRUE for multiple callbacks.

### 5.7.3 Function Documentation

**5.7.3.1** `globus_result_t globus_ftp_client_register_read (globus_ftp_client_handle_t * handle, globus_byte_t * buffer, globus_size_t buffer_length, globus_ftp_client_data_callback_t callback, void * callback_arg)`

Register a data buffer to handle a part of the FTP data transfer.

The data buffer will be associated with the current get being performed on this client handle.

#### Parameters:

*handle* A pointer to a FTP Client handle which contains state information about the get operation.

*buffer* A user-supplied buffer into which data from the FTP server will be stored.

*buffer\_length* The maximum amount of data that can be stored into the buffer.

*callback* The function to be called once the data has been read.

*callback\_arg* Argument passed to the callback function

#### See also:

`globus_ftp_client_operationattr_set_read_all()` (p. 52)

**5.7.3.2** `globus_result_t globus_ftp_client_register_write (globus_ftp_client_handle_t * handle, globus_byte_t * buffer, globus_size_t buffer_length, globus_off_t offset, globus_bool_t eof, globus_ftp_client_data_callback_t callback, void * callback_arg)`

Register a data buffer to handle a part of the FTP data transfer.

The data buffer will be associated with the current "put" being performed on this client handle. Multiple data buffers may be registered on a handle at once. There is no guaranteed ordering of the data callbacks in extended block mode.

#### Parameters:

*handle* A pointer to a FTP Client handle which contains state information about the put operation.

*buffer* A user-supplied buffer containing the data to write to the server.

*buffer\_length* The size of the buffer to be written.

*offset* The offset of the buffer to be written. In extended-block mode, the data does not need to be sent in order. In stream mode (the default), data must be sent in sequential order. The behavior is undefined if multiple writes overlap.

*eof*

*callback* The function to be called once the data has been written.

*callback\_arg* Argument passed to the callback function

## 5.8 Debugging Plugin

Collaboration diagram for Debugging Plugin:



The FTP Debugging plugin provides a way for the user to trace FTP protocol messages which occur while the GridFTP client library processes an FTP operation.

## Defines

- `#define GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_MODULE (&globus_i_ftp_client_debug_plugin_module)`

## Functions

- `globus_result_t globus_ftp_client_debug_plugin_init (globus_ftp_client_plugin_t *plugin, FILE *stream, const char *text)`
- `globus_result_t globus_ftp_client_debug_plugin_destroy (globus_ftp_client_plugin_t *plugin)`

### 5.8.1 Detailed Description

The FTP Debugging plugin provides a way for the user to trace FTP protocol messages which occur while the GridFTP client library processes an FTP operation.

This may be useful for debugging FTP configuration problems.

When this plugin is used for a GridFTP Client operation, information will be printed to the file stream associated with the plugin when a user begins an operation, for all data buffers which pass through while handling a data transfer, and for all protocol messages which are sent and received.

#### Example Usage:

The following example illustrates a typical use of the debug plugin. In this case, we configure a plugin instance to output log messages preceded by the process name and pid to a file named `gridftp.log`.

```
int main(int argc, char *argv[])
{
    globus_ftp_client_plugin_t restart_plugin;
    globus_ftp_client_handleattr_t handleattr;
    globus_ftp_client_handle_t handle;
    FILE * log;
    char text[256];

    /* Activate the necessary modules */
    globus_module_activate (GLOBUS_FTP_CLIENT_MODULE);
    globus_module_activate (GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_MODULE);

    /* Configure plugin to show custom text, and send plugin data to
     * a custom log file
     */
    log = fopen("gridftp.log", "a");
    sprintf(text, "%s:%ld", argv[0], (long) getpid());

    globus_ftp_client_debug_plugin_init(&debug_plugin, log, text);

    /* Set up our client handle to use the new plugin */
    globus_ftp_client_handleattr_init(&handleattr);
    globus_ftp_client_handleattr_add_plugin(&handleattr, &debug_plugin);
    globus_ftp_client_handle_init(&handle, &handleattr);

    /* As this get is processed, data will be appended to our gridftp.log
     * file
     */
    globus_ftp_client_get(&handle,
                        "ftp://ftp.globus.org/pub/globus/README",
                        GLOBUS_NULL,
                        GLOBUS_NULL,
                        callback_fn,
                        GLOBUS_NULL);
}
```

## 5.8.2 Define Documentation

### 5.8.2.1 `#define GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_MODULE (&globus_i_ftp_client_debug_plugin_module)`

Module descriptor.

## 5.8.3 Function Documentation

### 5.8.3.1 `globus_result_t globus_ftp_client_debug_plugin_init (globus_ftp_client_plugin_t * plugin, FILE * stream, const char * text)`

Initialize an instance of the GridFTP debugging plugin

This function will initialize the debugging plugin-specific instance data for this plugin, and will make the plugin usable for ftp client handle attribute and handle creation.

#### Parameters:

*plugin* A pointer to an uninitialized plugin. The plugin will be configured as a debugging plugin, with the default of sending debugging messages to stderr.

*stream*

*text*

#### Returns:

This function returns an error if

- *plugin* is null

#### See also:

`globus_ftp_client_debug_plugin_destroy()` (p. 63), `globus_ftp_client_handleattr_add_plugin()` (p. 15), `globus_ftp_client_handleattr_remove_plugin()` (p. 18), `globus_ftp_client_handle_init()` (p. 8)

### 5.8.3.2 `globus_result_t globus_ftp_client_debug_plugin_destroy (globus_ftp_client_plugin_t * plugin)`

Destroy an instance of the GridFTP debugging plugin

This function will free all debugging plugin-specific instance data from this plugin, and will make the plugin unusable for further ftp handle creation.

Existing FTP client handles and handle attributes will not be affected by destroying a plugin associated with them, as a local copy of the plugin is made upon handle initialization.

#### Parameters:

*plugin* A pointer to a GridFTP debugging plugin, previously initialized by calling `globus_ftp_client_debug_plugin_init()` (p. 63)

#### Returns:

This function returns an error if

- *plugin* is null
- *plugin* is not a debugging plugin

#### See also:

`globus_ftp_client_debug_plugin_init()` (p. 63), `globus_ftp_client_handleattr_add_plugin()` (p. 15), `globus_ftp_client_handleattr_remove_plugin()` (p. 18), `globus_ftp_client_handle_init()` (p. 8)



## 5.9 Performance Marker Plugin

Collaboration diagram for Performance Marker Plugin:



The FTP Performance Marker plugin allows the user to obtain performance markers for all types of transfers except a third party transfer in which Extended Block mode is not enabled.

### Defines

- `#define GLOBUS_FTP_CLIENT_PERF_PLUGIN_MODULE (&globus_i_ftp_client_perf_plugin_module)`

### Typedefs

- `typedef void(*) globus_ftp_client_perf_plugin_begin_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url, globus_bool_t restart)`
- `typedef void(*) globus_ftp_client_perf_plugin_marker_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, long time_stamp_int, char time_stamp_tlength, int stripe_ndx, int num_stripes, globus_off_t nbytes)`
- `typedef void(*) globus_ftp_client_perf_plugin_complete_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, globus_bool_t success)`
- `typedef void(*) globus_ftp_client_perf_plugin_user_copy_cb_t (void *user_specific)`
- `typedef void(*) globus_ftp_client_perf_plugin_user_destroy_cb_t (void *user_specific)`

### Functions

- `globus_result_t globus_ftp_client_perf_plugin_init (globus_ftp_client_plugin_t *plugin, globus_ftp_client_perf_plugin_begin_cb_t begin_cb, globus_ftp_client_perf_plugin_marker_cb_t marker_cb, globus_ftp_client_perf_plugin_complete_cb_t complete_cb, void *user_specific)`
- `globus_result_t globus_ftp_client_perf_plugin_set_copy_destroy (globus_ftp_client_plugin_t *plugin, globus_ftp_client_perf_plugin_user_copy_cb_t copy_cb, globus_ftp_client_perf_plugin_user_destroy_cb_t destroy_cb)`
- `globus_result_t globus_ftp_client_perf_plugin_destroy (globus_ftp_client_plugin_t *plugin)`
- `globus_result_t globus_ftp_client_perf_plugin_get_user_specific (globus_ftp_client_plugin_t *plugin, void **user_specific)`

#### 5.9.1 Detailed Description

The FTP Performance Marker plugin allows the user to obtain performance markers for all types of transfers except a third party transfer in which Extended Block mode is not enabled.

These markers may be generated internally, or they may be received from a server ('put' or third\_party\_transfer' only).

Copy constructor and destructor callbacks are also provided to allow one to more easily layer other plugins on top of this one.

## 5.9.2 Define Documentation

### 5.9.2.1 `#define GLOBUS_FTP_CLIENT_PERF_PLUGIN_MODULE (&globus_i_ftp_client_perf_plugin_module)`

Module descriptor.

## 5.9.3 Typedef Documentation

### 5.9.3.1 `typedef void(*) globus_ftp_client_perf_plugin_begin_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url, globus_bool_t restart)`

Transfer begin callback

This callback is called when a get, put, or third party transfer is started.

Note that it is possible for this callback to be made multiple times before ever receiving the complete callback... this would be the case if a transfer was restarted. The 'restart' will indicate whether or not we have been restarted.

#### Parameters:

- handle* this the client handle that this transfer will be occurring on
- user\_specific* this is user specific data either created by the copy method, or, if a copy method was not specified, the value passed to init
- source\_url* source of the transfer (GLOBUS\_NULL if 'put')
- dest\_url* dest of the transfer (GLOBUS\_NULL if 'get')
- restart* boolean indicating whether this callback is result of a restart

#### Returns:

- n/a

### 5.9.3.2 `typedef void(*) globus_ftp_client_perf_plugin_marker_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, long time_stamp_int, char time_stamp_tlength, int stripe_ndx, int num_stripes, globus_off_t nbytes)`

Performance marker received callback

This callback is called for all types of transfers except a third party in which extended block mode is not used (because 112 perf markers wont be sent in that case).

For extended mode 'put' and '3pt', actual 112 perf markers will be used and the frequency of this callback is dependent upon the frequency those messages are received. For 'put' in which extended block mode is not enabled and 'get' transfers, the information in this callback will be determined locally and the frequency of this callback will be at a maximum of one per second.

#### Parameters:

- handle* this the client handle that this transfer is occurring on
- user\_specific* this is user specific data either created by the copy method, or, if a copy method was not specified, the value passed to init
- time\_stamp* the timestamp at which the number of bytes is valid
- stripe\_ndx* the stripe index this data refers to
- num\_stripes* total number of stripes involved in this transfer
- nbytes* the total bytes transfered on this stripe

#### Returns:

- n/a

### 5.9.3.3 `typedef void(*) globus_ftp_client_perf_plugin_complete_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, globus_bool_t success)`

Transfer complete callback

This callback will be called upon transfer completion (successful or otherwise).

#### Parameters:

- handle* this the client handle that this transfer was occurring on
- user\_specific* this is user specific data either created by the copy method, or, if a copy method was not specified, the value passed to init
- success* indicates whether this transfer completed successfully or was interrupted (by error or abort)

#### Returns:

- n/a

### 5.9.3.4 `typedef void*(*) globus_ftp_client_perf_plugin_user_copy_cb_t(void *user_specific)`

Copy constructor

This callback will be called when a copy of this plugin is made, it is intended to allow initialization of a new *user\_specific* data.

#### Parameters:

- user\_specific* this is user specific data either created by this copy method, or the value passed to init

#### Returns:

- a pointer to a user specific piece of data
- GLOBUS\_NULL (does not indicate error)

### 5.9.3.5 `typedef void(*) globus_ftp_client_perf_plugin_user_destroy_cb_t(void *user_specific)`

Destructor

This callback will be called when a copy of this plugin is destroyed, it is intended to allow the user to free up any memory associated with the user specific data.

#### Parameters:

- user\_specific* this is user specific data created by the copy method

#### Returns:

- n/a

## 5.9.4 Function Documentation

### 5.9.4.1 `globus_result_t globus_ftp_client_perf_plugin_init (globus_ftp_client_plugin_t * plugin, globus_ftp_client_perf_plugin_begin_cb_t begin_cb, globus_ftp_client_perf_plugin_marker_cb_t marker_cb, globus_ftp_client_perf_plugin_complete_cb_t complete_cb, void * user_specific)`

Initialize a perf plugin

This function initializes a performance marker plugin.

Any params except for the plugin may be GLOBUS\_NULL

**Parameters:**

*plugin* a pointer to a plugin type to be initialized

*user\_specific* a pointer to some user specific data that will be provided to all callbacks

*begin\_cb* the callback to be called upon the start of a transfer

*marker\_cb* the callback to be called with every performance marker received

*complete\_cb* the callback to be called to indicate transfer completion

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin
- Error on init internal plugin

**5.9.4.2** `globus_result_t globus_ftp_client_perf_plugin_set_copy_destroy (globus_ftp_client_plugin_t * plugin, globus_ftp_client_perf_plugin_user_copy_cb_t copy_cb, globus_ftp_client_perf_plugin_user_destroy_cb_t destroy_cb)`

Set user copy and destroy callbacks

Use this to have the plugin make callbacks any time a copy of this plugin is being made.

This will allow the user to keep state for different handles.

**Parameters:**

*plugin* plugin previously initialized with init (above)

*copy\_cb* func to be called when a copy is needed

*destroy\_cb* func to be called when a copy is to be destroyed

**Returns:**

- Error on NULL arguments
- GLOBUS\_SUCCESS

**5.9.4.3** `globus_result_t globus_ftp_client_perf_plugin_destroy (globus_ftp_client_plugin_t * plugin)`

Destroy performance marker plugin

Frees up memory associated with plugin.

**Parameters:**

*plugin* plugin previously initialized with init (above)

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin

#### 5.9.4.4 globus\_result\_t globus\_ftp\_client\_perf\_plugin\_get\_user\_specific (globus\_ftp\_client\_plugin\_t \* plugin, void \*\* user\_specific)

Retrieve user specific pointer.

#### Parameters:

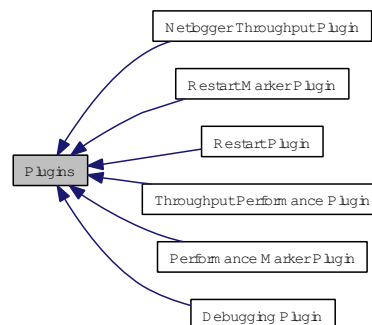
- plugin* plugin previously initialized with init (above)
- user\_specific* pointer to storage for user\_specific pointer

#### Returns:

- GLOBUS\_SUCCESS
- Error on NULL plugin
- Error on NULL user\_specific

## 5.10 Plugins

Collaboration diagram for Plugins:



Plugin API.

#### Modules

- Debugging Plugin
- Performance Marker Plugin
- Restart Marker Plugin
- Restart Plugin
- Netlogger Throughput Plugin
- Throughput Performance Plugin

#### Typedefs

- typedef globus\_i\_ftp\_client\_plugin\_t \* globus\_ftp\_client\_plugin\_t
- typedef globus\_ftp\_client\_plugin\_t \*(\*) globus\_ftp\_client\_plugin\_copy\_t (globus\_ftp\_client\_plugin\_t \*plugin\_template, void \*plugin\_specific)
- typedef void(\*) globus\_ftp\_client\_plugin\_destroy\_t (globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific)
- typedef void(\*) globus\_ftp\_client\_plugin\_connect\_t (globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific, globus\_ftp\_client\_handle\_t \*handle, const char \*url)

- [illegible]

- `typedef void(*) globus_ftp_client_plugin_put_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`
- `typedef void(*) globus_ftp_client_plugin_third_party_transfer_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *source_url, const globus_ftp_client_operationattr_t *source_attr, const char *dest_url, const globus_ftp_client_operationattr_t *dest_attr, globus_bool_t restart)`
- `typedef void(*) globus_ftp_client_plugin_modification_time_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`
- `typedef void(*) globus_ftp_client_plugin_size_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`
- `typedef void(*) globus_ftp_client_plugin_abort_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle)`
- `typedef void(*) globus_ftp_client_plugin_read_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const globus_byte_t *buffer, globus_size_t buffer_length)`
- `typedef void(*) globus_ftp_client_plugin_write_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const globus_byte_t *buffer, globus_size_t buffer_length, globus_off_t offset, globus_bool_t eof)`
- `typedef void(*) globus_ftp_client_plugin_data_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, globus_object_t *error, const globus_byte_t *buffer, globus_size_t length, globus_off_t offset, globus_bool_t eof)`
- `typedef void(*) globus_ftp_client_plugin_command_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const char *command)`
- `typedef void(*) globus_ftp_client_plugin_response_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, globus_object_t *error, const globus_ftp_control_response_t *ftp_response)`
- `typedef void(*) globus_ftp_client_plugin_fault_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, globus_object_t *error)`
- `typedef void(*) globus_ftp_client_plugin_complete_t (globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle)`

## Enumerations

- `enum globus_ftp_client_plugin_command_mask_t { ,  
GLOBUS_FTP_CLIENT_CMD_MASK_CONTROL_ESTABLISHMENT = 1<<0,  
GLOBUS_FTP_CLIENT_CMD_MASK_DATA_ESTABLISHMENT = 1<<1,  
GLOBUS_FTP_CLIENT_CMD_MASK_TRANSFER_PARAMETERS = 1<<2,  
GLOBUS_FTP_CLIENT_CMD_MASK_TRANSFER_MODIFIERS = 1<<3,  
GLOBUS_FTP_CLIENT_CMD_MASK_FILE_ACTIONS = 1<<4,  
GLOBUS_FTP_CLIENT_CMD_MASK_INFORMATION = 1<<5,  
GLOBUS_FTP_CLIENT_CMD_MASK_MISC = 1<<6,  
GLOBUS_FTP_CLIENT_CMD_MASK_BUFFER = 1<<7,  
GLOBUS_FTP_CLIENT_CMD_MASK_ALL = 0x7fffffff }`

## Functions

- `globus_result_t globus_ftp_client_plugin_restart_list (globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, const globus_abstime_t *when)`

- [illegible]



- **globus\_result\_t globus\_ftp\_client\_plugin\_add\_data\_channels** (globus\_ftp\_client\_handle\_t \*handle, unsigned int num\_channels, unsigned int stripe)
- **globus\_result\_t globus\_ftp\_client\_plugin\_remove\_data\_channels** (globus\_ftp\_client\_handle\_t \*handle, unsigned int num\_channels, unsigned int stripe)

### 5.10.1 Detailed Description

Plugin API.

A plugin is a way to implement application-independent reliability and performance tuning behavior. Plugins are written using the API described in this document.

A plugin is created by defining a `globus_ftp_client_plugin_t` which contains the function pointers and plugin-specific data needed for the plugin's operation. It is recommended that a plugin define a `globus_module_descriptor_t` and plugin initialization functions, to ensure that the plugin is properly initialized.

The functions pointed to in a plugin are called when significant events in the life of an FTP Client operation occur. Note that plugins will only be called when the plugin has the function pointer for both the operation (get, put, list, etc), and the event (connect, authenticate, command, etc), are defined. The command and response functions are filtered based on the `command_mask` defined in the plugin structure.

Every plugin must define **copy** (p. 72) and **destroy** (p. 73) functions. The copy function is called when the plugin is added to an attribute set or a handle is initialized with an attribute set containing the plugin. The destroy function is called when the handle or attribute set is destroyed.

### 5.10.2 Typedef Documentation

#### 5.10.2.1 typedef struct globus\_i\_ftp\_client\_plugin\_t\* globus\_ftp\_client\_plugin\_t

FTP Client plugin

An FTP Client plugin is used to add restart, monitoring, and performance tuning operations to the FTP Client library, without modifying the base API.

Multiple plugins may be associated with a `globus_ftp_client_handle_t`.

See also:

**globus\_ftp\_client\_handle\_init()** (p. 8), **globus\_ftp\_client\_handle\_destroy()** (p. 9), **globus\_ftp\_client\_handleattr\_t** (p. 12), **Debugging Plugin** (p. 61)

#### 5.10.2.2 typedef globus\_ftp\_client\_plugin\_t\*(\*) globus\_ftp\_client\_plugin\_copy\_t(globus\_ftp\_client\_plugin\_t \*plugin\_template, void \*plugin\_specific)

Plugin copy function.

This function is used to create a new copy or reference count a plugin. This function is called by the FTP Client library when a plugin is added to a handle attribute set, or when a handle is initialized with an attribute which contains the plugin.

A plugin may not call any of the plugin API functions from its instantiate method.

Parameters:

**plugin\_template** A plugin previously initialized by a call to the plugin-specific initialization function. by the user.

**plugin\_specific** Plugin-specific data.

**Returns:**

A pointer to a plugin. This plugin copy must remain valid until the copy's **destroy** (p. 73) function is called on the copy.

**See also:**

**globus\_ftp\_client\_plugin\_destroy\_t** (p. 73)

**5.10.2.3 typedef void(\*) globus\_ftp\_client\_plugin\_destroy\_t(globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific)**

Plugin destroy function.

This function is used to free or unreference a copy of a plugin which was allocated by calling the **globus\_ftp\_client\_plugin\_instantiate\_t** function from the plugin.

**Parameters:**

**plugin** The plugin, created by the **globus\_ftp\_client\_plugin\_instantiate\_t** function, which is to be destroyed.

**plugin\_specific** Plugin-specific data.

**5.10.2.4 typedef void(\*) globus\_ftp\_client\_plugin\_connect\_t(globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific, globus\_ftp\_client\_handle\_t \*handle, const char \*url)**

Plugin connection begin function.

This callback is used to notify a plugin that connection establishment is being done for this client handle. This notification can occur when a new request is made or when a restart is done by a plugin.

If a **response\_callback** is defined by a plugin, then that will be once the connection establishment has completed (successfully or unsuccessfully).

**Parameters:**

**plugin** The plugin which is being notified.

**plugin\_specific** Plugin-specific data.

**handle** The handle associated with the connection.

**Note:**

This function will not be called for a get, put, or third-party transfer operation when a cached connection is used.

**5.10.2.5 typedef void(\*) globus\_ftp\_client\_plugin\_authenticate\_t(globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific, globus\_ftp\_client\_handle\_t \*handle, const char \*url, const globus\_ftp\_control\_auth\_info\_t \*auth\_info)**

Plugin authentication notification callback.

This callback is used to notify a plugin that an authentication handshake is being done for this client handle. This notification can occur when a new request is made or when a hard restart is done by a plugin.

If a **response\_callback** is defined by a plugin, then that will be once the authentication has completed (successfully or unsuccessfully).

**Parameters:**

**plugin** The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the connection.

*url* The URL of the server to connect to.

*auth\_info* Authentication and authorization info being used to authenticate with the FTP or GridFTP server.

**5.10.2.6** `typedef void(*) globus_ftp_client_plugin_chmod_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, int mode, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin chmod notification callback.

This callback is used to notify a plugin that a chmod is being requested on a client handle. This notification happens both when the user requests a chmod, and when a plugin restarts the currently active chmod request.

If this function is not defined by the plugin, then no plugin callbacks associated with the chmod will be called.

#### Parameters:

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the delete operation.

*url* The url to chmod.

*mode* The file mode to be applied.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current delete operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.7** `typedef void(*) globus_ftp_client_plugin_chgrp_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const char *group, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin chgrp notification callback.

This callback is used to notify a plugin that a chgrp is being requested on a client handle. This notification happens both when the user requests a chgrp, and when a plugin restarts the currently active chgrp request.

If this function is not defined by the plugin, then no plugin callbacks associated with the chgrp will be called.

#### Parameters:

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the delete operation.

*url* The url to chgrp.

*group* The group name or ID to change to.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current delete operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.8** `typedef void(*) globus_ftp_client_plugin_utime_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const struct tm *utime_time, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin utime notification callback.

This callback is used to notify a plugin that a utime is being requested on a client handle. This notification happens both when the user requests a utime, and when a plugin restarts the currently active utime request.

If this function is not defined by the plugin, then no plugin callbacks associated with the utime will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the utime operation.

*url* The url to utime.

*utime\_time* The modification time to change the file to.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current utime operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.9** `typedef void(*) globus_ftp_client_plugin_symlink_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const char *utime_time, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin symlink notification callback.

This callback is used to notify a plugin that a symlink is being requested on a client handle. This notification happens both when the user requests a symlink, and when a plugin restarts the currently active symlink request.

If this function is not defined by the plugin, then no plugin callbacks associated with the symlink will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the symlink operation.

*url* The url of the new link.

*link\_url* The url to which the new link should point.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current utime operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.10** `typedef void(*) globus_ftp_client_plugin_cksm_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, globus_off_t offset, globus_off_t length, const char *algorithm, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin cksum notification callback.

This callback is used to notify a plugin that a cksum is being requested on a client handle. This notification happens both when the user requests a cksum, and when a plugin restarts the currently active cksum request.

If this function is not defined by the plugin, then no plugin callbacks associated with the cksum will be called.

#### Parameters:

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the delete operation.

*url* The url to cksm.

*offset* File offset to start calculating checksum.

*length* Length of data to read from the starting offset. Use -1 to read the entire file.

*algorithm* A pointer to a string to be filled with the checksum of the file. On error the value pointed to by it is undefined.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current delete operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.11** `typedef void(*) globus_ftp_client_plugin_delete_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin delete notification callback.

This callback is used to notify a plugin that a delete is being requested on a client handle. This notification happens both when the user requests a delete, and when a plugin restarts the currently active delete request.

If this function is not defined by the plugin, then no plugin callbacks associated with the delete will be called.

#### Parameters:

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the delete operation.

*url* The url to be deleted.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current delete operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.12** `typedef void(*) globus_ftp_client_plugin_feat_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin feat notification callback.

This callback is used to notify a plugin that a feat is being requested on a client handle. This notification happens both when the user requests a feat, and when a plugin restarts the currently active feat request.

If this function is not defined by the plugin, then no plugin callbacks associated with the feat will be called.

#### Parameters:

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the feat operation.

*url* The url to be feat'd.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current feat operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.13** `typedef void(*) globus_ftp_client_plugin_mkdir_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin mkdir notification callback.

This callback is used to notify a plugin that a mkdir is being requested on a client handle. This notification happens both when the user requests a mkdir, and when a plugin restarts the currently active mkdir request.

If this function is not defined by the plugin, then no plugin callbacks associated with the mkdir will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the mkdir operation.

*url* The url of the directory to create.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current mkdir operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.14** `typedef void(*) globus_ftp_client_plugin_rmdir_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin rmdir notification callback.

This callback is used to notify a plugin that a rmdir is being requested on a client handle. This notification happens both when the user requests a rmdir, and when a plugin restarts the currently active rmdir request.

If this function is not defined by the plugin, then no plugin callbacks associated with the rmdir will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the rmdir operation.

*url* The url of the rmdir operation.

*attr* The attributes to be used during this operation.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current rmdir operation; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.15** `typedef void(*) globus_ftp_client_plugin_list_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin list notification callback.

This callback is used to notify a plugin that a list is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.16** `typedef void(*) globus_ftp_client_plugin_verbose_list_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin verbose list notification callback.

This callback is used to notify a plugin that a list is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.17** `typedef void(*) globus_ftp_client_plugin_machine_list_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin machine list notification callback.

This callback is used to notify a plugin that a list is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.18** `typedef void(*) globus_ftp_client_plugin_recursive_list_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin recursive list notification callback.

This callback is used to notify a plugin that a list is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.19** `typedef void(*) globus_ftp_client_plugin_mlst_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin mlst notification callback.

This callback is used to notify a plugin that a mlst is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.20** `typedef void(*) globus_ftp_client_plugin_stat_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin stat notification callback.

This callback is used to notify a plugin that a stat is being requested on a client handle. This notification happens both when the user requests a list, and when a plugin restarts the currently active list request.

If this function is not defined by the plugin, then no plugin callbacks associated with the list will be called.

**Parameters:**

*plugin* The plugin which is being notified.



***plugin\_specific*** Plugin-specific data.

***handle*** The handle associated with the list operation.

***url*** The url of the list operation.

***attr*** The attributes to be used during this transfer.

***restart*** This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.21** `typedef void(*) globus_ftp_client_plugin_move_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin move notification callback.

This callback is used to notify a plugin that a move is being requested on a client handle. This notification happens both when the user requests a move, and when a plugin restarts the currently active move request.

If this function is not defined by the plugin, then no plugin callbacks associated with the move will be called.

**Parameters:**

***plugin*** The plugin which is being notified.

***plugin\_specific*** Plugin-specific data.

***handle*** The handle associated with the move operation.

***source\_url*** The source url of the move operation.

***dest\_url*** The destination url of the move operation.

***attr*** The attributes to be used during this move.

***restart*** This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current move transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.22** `typedef void(*) globus_ftp_client_plugin_get_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin get notification callback.

This callback is used to notify a plugin that a get is being requested on a client handle. This notification happens both when the user requests a get, and when a plugin restarts the currently active get request.

If this function is not defined by the plugin, then no plugin callbacks associated with the get will be called.

**Parameters:**

***plugin*** The plugin which is being notified.

***plugin\_specific*** Plugin-specific data.

***handle*** The handle associated with the get operation.

***url*** The url of the get operation.

***attr*** The attributes to be used during this transfer.

***restart*** This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current get transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.23** `typedef void(*) globus_ftp_client_plugin_put_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin put notification callback.

This callback is used to notify a plugin that a put is being requested on a client handle. This notification happens both when the user requests a put, and when a plugin restarts the currently active put request.

If this function is not defined by the plugin, then no plugin callbacks associated with the put will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the put operation.

*url* The url of the put operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current put transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.24** `typedef void(*) globus_ftp_client_plugin_third_party_transfer_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *source_url, const globus_ftp_client_operationattr_t *source_attr, const char *dest_url, const globus_ftp_client_operationattr_t *dest_attr, globus_bool_t restart)`

Plugin third-party transfer notification callback.

This callback is used to notify a plugin that a transfer is being requested on a client handle. This notification happens both when the user requests a transfer, and when a plugin restarts the currently active transfer request.

If this function is not defined by the plugin, then no plugin callbacks associated with the third-party transfer will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the transfer operation.

*source\_url* The source url of the transfer operation.

*source\_attr* The attributes to be used during this transfer on the source.

*dest\_url* The destination url of the third-party transfer operation.

*dest\_attr* The attributes to be used during this transfer on the destination.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current transfer transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.25** `typedef void(*) globus_ftp_client_plugin_modification_time_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin modification time notification callback.

This callback is used to notify a plugin that a modification time check is being requested on a client handle. This notification happens both when the user requests the modification time of a file, and when a plugin restarts the currently active request.

If this function is not defined by the plugin, then no plugin callbacks associated with the modification time request will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.26** `typedef void(*) globus_ftp_client_plugin_size_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const globus_ftp_client_operationattr_t *attr, globus_bool_t restart)`

Plugin size notification callback.

This callback is used to notify a plugin that a size check is being requested on a client handle. This notification happens both when the user requests the size of a file, and when a plugin restarts the currently active request.

If this function is not defined by the plugin, then no plugin callbacks associated with the size request will be called.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the list operation.

*url* The url of the list operation.

*attr* The attributes to be used during this transfer.

*restart* This value is set to GLOBUS\_TRUE when this callback is caused by a plugin restarting the current list transfer; otherwise, this is set to GLOBUS\_FALSE.

**5.10.2.27** `typedef void(*) globus_ftp_client_plugin_abort_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle)`

Plugin abort notification callback.

This callback is used to notify a plugin that an abort is being requested on a client handle. This notification happens both when the user aborts a request and when a plugin aborts the currently active request.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

**5.10.2.28** `typedef void(*) globus_ftp_client_plugin_read_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const globus_byte_t *buffer, globus_size_t buffer_length)`

Plugin read registration callback.

This callback is used to notify a plugin that the client API has registered a buffer with the FTP control API for reading when processing a get.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*buffer* The data buffer to read into.

*buffer\_length* The maximum amount of data to read into the buffer.

**5.10.2.29** `typedef void(*) globus_ftp_client_plugin_write_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const globus_byte_t *buffer, globus_size_t buffer_length, globus_off_t offset, globus_bool_t eof)`

Plugin write registration callback.

This callback is used to notify a plugin that the client API has registered a buffer with the FTP control API for writing when processing a put.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*buffer* The buffer which is being written.

*buffer\_length* The amount of data in the buffer.

*offset* The offset within the file where the buffer is to be written.

*eof* This value is set to GLOBUS\_TRUE if this is the last data buffer to be sent for this put request.

**5.10.2.30** `typedef void(*) globus_ftp_client_plugin_data_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, globus_object_t *error, const globus_byte_t *buffer, globus_size_t length, globus_off_t offset, globus_bool_t eof)`

Plugin data callback handler.

This callback is used to notify a plugin that a read or write operation previously registered has completed. The buffer pointer will match that of a previous plugin read or write registration callback.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*buffer* The buffer which was successfully transferred over the network.

*length* The amount of data to read or written.

*offset* The offset into the file where this data buffer belongs.

*eof* This value is set to GLOBUS\_TRUE if end-of-file is being processed for this transfer.

**5.10.2.31** `typedef void(*) globus_ftp_client_plugin_command_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, const char *command)`

Command callback.

This callback is used to notify a plugin that a FTP control command is being sent. The client library will only call this function for response callbacks associated with a command which is in the plugin's command mask, and associated with one of the other ftp operations with a defined callback in the plugin.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*url* The URL which this command is being sent to.

*command* A string containing the command which is being sent to the server (TYPE I, for example).

**5.10.2.32** `typedef void(*) globus_ftp_client_plugin_response_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, globus_object_t *error, const globus_ftp_control_response_t *ftp_response)`

Response callback.

This callback is used to notify a plugin that a FTP control response has occurred on a control connection. FTP response callbacks will come back to the user in the order which the commands were executed. The client library will only call this function for response callbacks associated with a command which is in the plugin's command mask, or associated with one of the other ftp operations with a defined callback in the plugin.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*url* The URL which this response came from.

*error* An error which occurred while processing this command/response pair.

*ftp\_response* The response structure from the ftp control library.

**5.10.2.33** `typedef void(*) globus_ftp_client_plugin_fault_t(globus_ftp_client_plugin_t *plugin, void *plugin_specific, globus_ftp_client_handle_t *handle, const char *url, globus_object_t *error)`

Fault notification callback.

This callback is used to notify a plugin that a fault occurred while processing the request. The fault may be internally generated, or come from a call to another library.

**Parameters:**

*plugin* The plugin which is being notified.

*plugin\_specific* Plugin-specific data.

*handle* The handle associated with the request.

*url* The url being processed when the fault occurred.

*error* An error object describing the fault.

**5.10.2.34** typedef void(\*) globus\_ftp\_client\_plugin\_complete\_t(globus\_ftp\_client\_plugin\_t \*plugin, void \*plugin\_specific, globus\_ftp\_client\_handle\_t \*handle)

Completion notification callback.

This callback is used to notify a plugin that an operation previously begun has completed. The plugin may not call any other plugin operation on this handle after this has occurred. This is the final callback for the plugin while processing the operation. The plugin may free any internal state associated with the operation at this point.

**Parameters:**

*plugin* The plugin which is being notified.  
*plugin\_specific* Plugin-specific data.  
*handle* The handle associated with the operation.

### 5.10.3 Enumeration Type Documentation

#### 5.10.3.1 enum globus\_ftp\_client\_plugin\_command\_mask\_t

Command Mask.

This enumeration includes the types of commands which the plugin is interested in.

**Enumerator:**

**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_CONTROL\_ESTABLISHMENT** connect, authenticate  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_DATA\_ESTABLISHMENT** PASV, PORT, SPOR, SPAS.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_TRANSFER\_PARAMETERS** MODE, TYPE, STRU, OPTS  
RETR, DCAU.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_TRANSFER\_MODIFIERS** ALLO, REST.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_FILE\_ACTIONS** STOR, RETR, ESTO, ERET, APPE, LIST,  
NLST, MLSD, MLRS, GET, PUT.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_INFORMATION** HELP, SITE HELP, FEAT, STAT, SYST, SIZE.  
  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_MISC** SITE, NOOP.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_BUFFER** SBUF, ABUF.  
**GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_ALL** All possible commands.

### 5.10.4 Function Documentation

**5.10.4.1** globus\_result\_t globus\_ftp\_client\_plugin\_restart\_list (globus\_ftp\_client\_handle\_t \*handle, const char \*url, const globus\_ftp\_client\_operationattr\_t \*attr, const globus\_abstime\_t \*when)

Restart an existing list.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

*handle* The handle which is associated with the list.

- url*** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.
- attr*** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.
- when*** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before ***when***, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

#### 5.10.4.2 **globus\_result\_t globus\_ftp\_client\_plugin\_restart\_verbose\_list (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const globus\_ftp\_client\_operationattr\_t \* *attr*, const globus\_abstime\_t \* *when*)**

Restart an existing verbose list.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

#### **Parameters:**

- handle*** The handle which is associated with the list.
- url*** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.
- attr*** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.
- when*** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before ***when***, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

#### 5.10.4.3 **globus\_result\_t globus\_ftp\_client\_plugin\_restart\_machine\_list (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const globus\_ftp\_client\_operationattr\_t \* *attr*, const globus\_abstime\_t \* *when*)**

Restart an existing machine list.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

#### **Parameters:**

- handle*** The handle which is associated with the list.
- url*** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.
- attr*** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.

**when** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.4 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_recursive\_list (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing recursive list.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the list.

**source\_url** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**source\_attr** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.

**when** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.5 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_mlst (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing MLST.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the list.

**url** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.

**when** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.



**5.10.4.6 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_stat (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const globus\_ftp\_client\_operationattr\_t \* *attr*, const globus\_abstime\_t \* *when*)**

Restart an existing STAT.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in list events will receive a list callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the list.

***url*** The destination URL of the transfer. This may be different than the original list's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original list's attribute set.

***when*** Absolute time for when to restart the list. The current control and data connections will be stopped immediately. If this completes before ***when***, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.7 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_chmod (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, int *mode*, const globus\_ftp\_client\_operationattr\_t \* *attr*, const globus\_abstime\_t \* *when*)**

Restart an existing chmod.

This function will cause the currently executing chmod operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in chmod events will receive a chmod callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the chmod.

***url*** The destination URL of the transfer. This may be different than the original chmod's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

***mode*** The file mode that will be applied. Must be an octal number representing the bit pattern for the new permissions.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original chmod's attribute set.

***when*** Absolute time for when to restart the chmod. The current control and data connections will be stopped immediately. If this completes before ***when***, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.8 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_chgrp (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const char \* *group*, const globus\_ftp\_client\_operationattr\_t \* *attr*, const globus\_abstime\_t \* *when*)**

Restart an existing chgrp.

This function will cause the currently executing chgrp operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in chgrp events will receive a chgrp callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the chgrp.

**url** The destination URL of the transfer. This may be different than the original chgrp's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**group** The group name or ID to change the file to.

**attr** The attributes to use for the new transfer. This may be a modified version of the original chgrp's attribute set.

**when** Absolute time for when to restart the chgrp. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.9 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_utime (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const struct tm \* utime\_time, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing utime.

This function will cause the currently executing utime operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in utime events will receive a utime callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the chgrp.

**url** The destination URL of the transfer. This may be different than the original utime's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**utime\_time** The time value to change the file to.

**attr** The attributes to use for the new transfer. This may be a modified version of the original utime's attribute set.

**when** Absolute time for when to restart the utime. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.10 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_symlink (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const char \* link\_url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing symlink.

This function will cause the currently executing symlink operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in chgrp events will receive a chgrp callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the chgrp.

**url** The destination URL of the transfer. This may be different than the original symlink's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**link\_url** The URL to symbolically link the file to.

**attr** The attributes to use for the new transfer. This may be a modified version of the original symlink's attribute set.

**when** Absolute time for when to restart the symlink. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.11 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_cksm (globus\_ftp\_client\_handle\_t \* handle, const char \* url, globus\_off\_t offset, globus\_off\_t length, const char \* algorithm, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing cksum.

This function will cause the currently executing cksum operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in cksum events will receive a cksum callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the cksum.

**url** The destination URL of the transfer. This may be different than the original cksum's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**offset** File offset to start calculating checksum.

**length** Length of data to read from the starting offset. Use -1 to read the entire file.

**algorithm** A pointer to a string to be filled with the checksum of the file. On error the value pointed to by it is undefined.

**attr** The attributes to use for the new transfer. This may be a modified version of the original cksum's attribute set.

**when** Absolute time for when to restart the cksum. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.12 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_delete (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing delete.

This function will cause the currently executing delete operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in delete events will receive a delete callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the delete.

**url** The destination URL of the transfer. This may be different than the original delete's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new transfer. This may be a modified version of the original delete's attribute set.

**when** Absolute time for when to restart the delete. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.13 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_feat (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing feat.

This function will cause the currently executing feat operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in feat events will receive a feat callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the feat.

**url** The destination URL of the transfer. This may be different than the original feat's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new transfer. This may be a modified version of the original feat's attribute set.

**when** Absolute time for when to restart the feat. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.14 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_mkdir (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart an existing mkdir.

This function will cause the currently executing operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in mkdir events will receive a mkdir callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the mkdir.

**url** The destination URL of the transfer. This may be different than the original mkdir's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new transfer. This may be a modified version of the original mkdir's attribute set.

**when** Absolute time for when to restart the mkdir. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.15** `globus_result_t globus_ftp_client_plugin_restart_rmdir (globus_ftp_client_handle_t * handle, const char * url, const globus_ftp_client_operationattr_t * attr, const globus_abstime_t * when)`

Restart an existing rmdir.

This function will cause the currently executing operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in rmdir events will receive a rmdir callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the rmdir.

***url*** The destination URL of the transfer. This may be different than the original rmdir's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original rmdir's attribute set.

**when** Absolute time for when to restart the rmdir. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.16** `globus_result_t globus_ftp_client_plugin_restart_move (globus_ftp_client_handle_t * handle, const char * source_url, const char * dest_url, const globus_ftp_client_operationattr_t * attr, const globus_abstime_t * when)`

Restart an existing move.

This function will cause the currently executing move operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially new URLs and attributes.

The user will not receive any notification that a restart has happened. Each plugin which is interested in get events will receive a move callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the move.

***source\_url*** The source URL of the move. This may be different than the original get's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

***dest\_url*** The destination URL of the move. This may be different than the original get's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL. Note that only the path component of this URL is used.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original move's attribute set. This may be useful when the plugin wishes to send restart markers to the FTP server to prevent re-sending the data which has already been sent.

**when** Absolute time for when to restart the move. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.17 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_get (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart\_marker*, const globus\_abstime\_t \* *when*)**

Restart an existing get.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued will be cleared and reused once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in get events will receive a get callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the get.

***url*** The source URL of the transfer. This may be different than the original get's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original get's attribute set. This may be useful when the plugin wishes to send restart markers to the FTP server to prevent re-sending the data which has already been sent.

***restart\_marker*** Plugin-provided restart marker for resuming at a non-default restart point. This may be used to implement a persistent restart across process invocations. The default behavior if this is NULL is to use any restart information which has been received by the ftp client library while processing this operation when restarted.

***when*** Absolute time for when to restart the get. The current control and data connections will be stopped immediately. If this completes before ***when***, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.18 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_put (globus\_ftp\_client\_handle\_t \* *handle*, const char \* *url*, const globus\_ftp\_client\_operationattr\_t \* *attr*, globus\_ftp\_client\_restart\_marker\_t \* *restart\_marker*, const globus\_abstime\_t \* *when*)**

Restart an existing put.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes. Any data buffers which are currently queued but not called back will be resent once the connection is re-established.

The user will not receive any notification that a restart has happened. Each plugin which is interested in get events will receive a put callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

***handle*** The handle which is associated with the put.

***url*** The URL of the transfer. This may be different than the original put's URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL. If the put is restarted with a different URL, the plugin must re-send any data which has already been acknowledged by it's callback.

***attr*** The attributes to use for the new transfer. This may be a modified version of the original put's attribute set. This may be useful when the plugin wishes to send restart markers to the FTP server to prevent re-sending the data which has already been sent.

***restart\_marker*** Plugin-provided restart marker for resuming at a non-default restart point. This may be used to implement a persistent restart across process invocations. The default behavior if this is NULL is to use any restart information which has been received by the ftp client library while processing this operation when restarted.

**when** Absolute time for when to restart the put. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.19 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_third\_party\_transfer (globus\_ftp\_client\_handle\_t \* handle, const char \* source\_url, const globus\_ftp\_client\_operationattr\_t \* source\_attr, const char \* dest\_url, const globus\_ftp\_client\_operationattr\_t \* dest\_attr, globus\_ftp\_client\_restart\_marker\_t \* restart\_marker, const globus\_abstime\_t \* when)**

Restart an existing third-party transfer.

This function will cause the currently executing transfer operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URLs and attributes.

The user will not receive any notification that a restart has happened. Each plugin which is interested in third-party transfer events will receive a transfer callback with the restart boolean set to GLOBUS\_TRUE.

#### Parameters:

**handle** The handle which is associated with the transfer.

**source\_url** The source URL of the transfer. This may be different than the original URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL

**source\_attr** The attributes to use for the new transfer. This may be a modified version of the original transfer's attribute set. This may be useful when the plugin wishes to send restart markers to the FTP server to prevent re-sending the data which has already been sent.

**dest\_url** The destination URL of the transfer. This may be different than the original destination URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**dest\_attr** The attributes to use for the new transfer. This may be a modified version of the original transfer's attribute set. This may be useful when the plugin wishes to send restart markers to the FTP server to prevent re-sending the data which has already been sent.

**restart\_marker** Plugin-provided restart marker for resuming at a non-default restart point. This may be used to implement a persistent restart across process invocations. The default behavior if this is NULL is to use any restart information which has been received by the ftp client library while processing this operation when restarted.

**when** Absolute time for when to restart the transfer. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.20 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_size (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart a size check operation.

This function will cause the currently executing size check operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes.

The user will not receive any notification that a restart has happened. Each plugin which is interested in size operations will receive a size callback with the restart boolean set to GLOBUS\_TRUE.

#### Parameters:

**handle** The handle which is associated with the operation.

**url** The source URL of the size check. This may be different than the original operations URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new operation. This may be a modified version of the original operations's attribute set.

**when** Absolute time for when to restart the size check. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.21 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_modification\_time (globus\_ftp\_client\_handle\_t \* handle, const char \* url, const globus\_ftp\_client\_operationattr\_t \* attr, const globus\_abstime\_t \* when)**

Restart a modification time check operation.

This function will cause the currently executing modification time check operation to be restarted. When a restart happens, the operation will be silently aborted, and then restarted with potentially a new URL and attributes.

The user will not receive any notification that a restart has happened. Each plugin which is interested in modification time operations will receive a modification time callback with the restart boolean set to GLOBUS\_TRUE.

**Parameters:**

**handle** The handle which is associated with the operation.

**url** The source URL of the modification time check. This may be different than the original operations URL, if the plugin decides to redirect to another FTP server due to performance or reliability problems with the original URL.

**attr** The attributes to use for the new operation. This may be a modified version of the original operations's attribute set.

**when** Absolute time for when to restart the modification time check. The current control and data connections will be stopped immediately. If this completes before **when**, then the restart will be delayed until that time. Otherwise, it will be immediately restarted.

**5.10.4.22 globus\_result\_t globus\_ftp\_client\_plugin\_restart\_get\_marker (globus\_ftp\_client\_handle\_t \* handle, globus\_ftp\_client\_restart\_marker\_t \* marker)**

Get restart marker

This function will allow this user to get the restart marker associated with a restarted file transfer.

This function may only be called within the get, put, or third party transfer callback in which the 'restart' argument is GLOBUS\_TRUE

**Parameters:**

**handle** The handle which is associated with the transfer.

**marker** Pointer to an uninitialized restart marker type

**Returns:**

- Error on NULL handle or marker
- Error on invalid use of function
- GLOBUS\_SUCCESS (marker will be populated)



#### 5.10.4.23 `globus_result_t globus_ftp_client_plugin_abort (globus_ftp_client_handle_t * handle)`

Abort a transfer operation.

This function will cause the currently executing transfer operation to be aborted. When this happens, all plugins will be notified by their abort callbacks. Once those are processed, the complete callback will be called for all plugins, and then for the user's callback.

The complete callback will indicate that the transfer did not complete successfully.

##### Parameters:

*handle* The handle which is associated with the transfer.

#### 5.10.4.24 `globus_result_t globus_ftp_client_plugin_add_data_channels (globus_ftp_client_handle_t * handle, unsigned int num_channels, unsigned int stripe)`

Add data channels to an existing put transfer.

This function will cause the currently executing transfer operation to have additional data channels acquired if the attribute set allows it.

##### Parameters:

*handle* The handle which is associated with the transfer.

*num\_channels* The number of channels to add to the transfer.

*stripe* The stripe number to have the channels added to.

##### Note:

Do the plugins need to be notified when this happens?

#### 5.10.4.25 `globus_result_t globus_ftp_client_plugin_remove_data_channels (globus_ftp_client_handle_t * handle, unsigned int num_channels, unsigned int stripe)`

Remove data channels from an existing put transfer.

This function will cause the currently executing transfer operation to have data channels removed, if the attribute set allows it.

##### Parameters:

*handle* The handle which is associated with the transfer.

*num\_channels* The number of channels to remove from the transfer.

*stripe* The stripe number to have the channels removed from.

##### Note:

Do the plugins need to be notified when this happens?

## 5.11 Restart Marker Plugin

Collaboration diagram for Restart Marker Plugin:



This plugin is intended to allow users to make restart markers persistent.

## Defines

- `#define GLOBUS_FTP_CLIENT_RESTART_MARKER_PLUGIN_MODULE (&globus_i_ftp_client_restart_marker_plugin_module)`

## Typedefs

- `typedef globus_bool_t(*) globus_ftp_client_restart_marker_plugin_begin_cb_t (void *user_arg, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url, globus_ftp_client_restart_marker_t *user_saved_marker)`
- `typedef void(*) globus_ftp_client_restart_marker_plugin_marker_cb_t (void *user_arg, globus_ftp_client_handle_t *handle, globus_ftp_client_restart_marker_t *marker)`
- `typedef void(*) globus_ftp_client_restart_marker_plugin_complete_cb_t (void *user_arg, globus_ftp_client_handle_t *handle, globus_object_t *error, const char *error_url)`

## Functions

- `globus_result_t globus_ftp_client_restart_marker_plugin_init (globus_ftp_client_plugin_t *plugin, globus_ftp_client_restart_marker_plugin_begin_cb_t begin_cb, globus_ftp_client_restart_marker_plugin_marker_cb_t marker_cb, globus_ftp_client_restart_marker_plugin_complete_cb_t complete_cb, void *user_arg)`
- `globus_result_t globus_ftp_client_restart_marker_plugin_destroy (globus_ftp_client_plugin_t *plugin)`

### 5.11.1 Detailed Description

This plugin is intended to allow users to make restart markers persistent.

During a transfer, every marker received will result in the user's 'marker' callback being called with the new restart marker that can be stored. If the application were to prematurely terminate (while transferring), the user (after restarting the application) could pass this stored marker back to the plugin via the 'begin' callback to force the transfer to be restarted from the last marked point.

### 5.11.2 Define Documentation

#### 5.11.2.1 `#define GLOBUS_FTP_CLIENT_RESTART_MARKER_PLUGIN_MODULE (&globus_i_ftp_client_restart_marker_plugin_module)`

Module descriptor.

### 5.11.3 Typedef Documentation

#### 5.11.3.1 `typedef globus_bool_t(*) globus_ftp_client_restart_marker_plugin_begin_cb_t(void *user_arg, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url, globus_ftp_client_restart_marker_t *user_saved_marker)`

Transfer begin callback

This callback is called when a get, put, or third party transfer is started.

The intended use for this callback is for the user to use the transfer urls to locate a restart marker in some persistent storage. If one is found, it should be copied into 'user\_saved\_marker' and the callback should return GLOBUS\_TRUE. This will cause the transfer to be restarted using that restart marker. If one is not found, return GLOBUS\_FALSE to indicate that the transfer should proceed from the beginning.

In any case, this is also an opportunity for the user to set up any storage in anticipation of restart markers for this transfer.

**Parameters:**

*handle* this the client handle that this transfer will be occurring on  
*user\_arg* this is the user\_arg passed to the init func  
*source\_url* source of the transfer (GLOBUS\_NULL if 'put')  
*dest\_url* dest of the transfer (GLOBUS\_NULL if 'get')  
*user\_saved\_marker* pointer to an uninitialized restart marker

**Returns:**

- GLOBUS\_TRUE to indicate that the plugin should use 'user\_saved\_marker' to restart the transfer (and subsequently, destroy the marker)
- GLOBUS\_FALSE to indicate that 'user\_saved\_marker' has not been modified, and that the transfer should proceed normally

**5.11.3.2** `typedef void(*) globus_ftp_client_restart_marker_plugin_marker_cb_t(void *user_arg, globus_ftp_client_handle_t *handle, globus_ftp_client_restart_marker_t *marker)`

Restart marker received callback

This callback will be called every time a restart marker is available.

To receive restart markers in a 'put' or 'third\_party\_transfer', the transfer must be in Extended Block mode. 'get' transfers will have their markers generated internally. Markers generated internally will be 'sent' at most, once per second.

The intended use for this callback is to allow the user to store this marker (most likely in place of any previous marker) in a format that the 'begin\_cb' can parse and pass back.

**Parameters:**

*handle* this the client handle that this transfer is occurring on  
*user\_arg* this is the user\_arg passed to the init func  
*marker* the restart marker that has been received. This marker is owned by the caller. The user must use the copy method to keep it. Note: this restart marker currently contains all ranges received as of yet. Should I instead only pass a marker with the ranges just made available? If so, the user may need a way to combine restart markers (globus\_ftp\_client\_restart\_marker\_combine)

**Returns:**

- n/a

**5.11.3.3** `typedef void(*) globus_ftp_client_restart_marker_plugin_complete_cb_t(void *user_arg, globus_ftp_client_handle_t *handle, globus_object_t *error, const char *error_url)`

Transfer complete callback

This callback will be called upon transfer completion (successful or otherwise).

**Parameters:**

*handle* this the client handle that this transfer was occurring on  
*user\_arg* this is the user\_arg passed to the init func

**error** the error object indicating what went wrong (GLOBUS\_NULL on success)  
**error\_url** the url which is the source of the above error (GLOBUS\_NULL on success)

**Returns:**

- n/a

#### 5.11.4 Function Documentation

**5.11.4.1 globus\_result\_t globus\_ftp\_client\_restart\_marker\_plugin\_init (globus\_ftp\_client\_plugin\_t \* *plugin*, globus\_ftp\_client\_restart\_marker\_plugin\_begin\_cb\_t *begin\_cb*, globus\_ftp\_client\_restart\_marker\_plugin\_marker\_cb\_t *marker\_cb*, globus\_ftp\_client\_restart\_marker\_plugin\_complete\_cb\_t *complete\_cb*, void \* *user\_arg*)**

Initialize a restart marker plugin

This function initializes a restart marker plugin.

Any params except for the plugin may be GLOBUS\_NULL

**Parameters:**

***plugin*** a pointer to a plugin type to be initialized  
***user\_arg*** a pointer to some user specific data that will be provided to all callbacks  
***begin\_cb*** the callback to be called upon the start of a transfer  
***marker\_cb*** the callback to be called with every restart marker received  
***complete\_cb*** the callback to be called to indicate transfer completion

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin
- Error on init internal plugin

**5.11.4.2 globus\_result\_t globus\_ftp\_client\_restart\_marker\_plugin\_destroy (globus\_ftp\_client\_plugin\_t \* *plugin*)**

Destroy restart marker plugin

Frees up memory associated with plugin.

**Parameters:**

***plugin*** plugin previously initialized with init (above)

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin

## 5.12 Restart Plugin

Collaboration diagram for Restart Plugin:



The restart plugin implements one scheme for providing reliability functionality for the FTP Client library.

## Defines

- `#define GLOBUS_FTP_CLIENT_RESTART_PLUGIN_MODULE (&globus_i_ftp_client_restart_plugin_module)`

## Functions

- `globus_result_t globus_ftp_client_restart_plugin_init (globus_ftp_client_plugin_t *plugin, int max_retries, globus_reltime_t *interval, globus_abstime_t *deadline)`
- `globus_result_t globus_ftp_client_restart_plugin_destroy (globus_ftp_client_plugin_t *plugin)`

### 5.12.1 Detailed Description

The restart plugin implements one scheme for providing reliability functionality for the FTP Client library.

Other plugins may be developed to provide other methods of reliability.

The specific functionality of this plugin is to restart any FTP operation when a fault occurs. The plugin's operation is parameterized to control how often and when to attempt to restart the operation.

This restart plugin will restart an FTP operation if a noticeable fault has occurred—a connection timing out, a failure by the server to process a command, a protocol error, an authentication error.

This plugin has three user-configurable parameters; these are the maximum number of retries to attempt, the interval to wait between retries, and the deadline after which no further retries will be attempted. These are set by initializing a restart plugin instance with the function `globus_ftp_client_restart_plugin_init()` (p. 101).

**Example Usage** The following example illustrates a typical use of the restart plugin. In this case, we configure a plugin instance to restart the operation for up to an hour, using an exponential back-off between retries.

```
#include "globus_ftp_client.h"
#include "globus_ftp_client_restart_plugin.h"
#include "globus_time.h"

int
main(int argc, char *argv[])
{
    globus_ftp_client_plugin_t restart_plugin;
    globus_ftp_client_handleattr_t handleattr;
    globus_ftp_client_handle_t handle;
    globus_abstime_t deadline;

    globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
    globus_module_activate(GLOBUS_FTP_CLIENT_RESTART_PLUGIN_MODULE);

    /* Set a deadline to be now + 1 hour */
    GlobusAbstimeSet(&deadline, 60 * 60, 0);

    /* initialize a plugin with this deadline */
    globus_ftp_client_restart_plugin_init(
        &restart_plugin,
        0, /* # retry limit (0 means don't limit) */
        GLOBUS_NULL, /* interval between retries--null means
                     * exponential backoff
                     */
        &deadline);

    /* Set up our handle to use the new plugin */
    globus_ftp_client_handleattr_init(&handleattr);
    globus_ftp_client_handleattr_add_plugin(&handleattr, &restart_plugin);
    globus_ftp_client_handle_init(&handle, &handleattr);
}
```

```

/*
 * Now, if a fault occurs processing this get, the plugin will restart
 * it with an exponential back-off, and will bail if a fault occurs
 * after 1 hour of retrying
 */
globus_ftp_client_get(&handle,
                     "ftp://ftp.globus.org/pub/globus/README",
                     GLOBUS_NULL,
                     GLOBUS_NULL,
                     callback_fn,
                     GLOBUS_NULL);
}

```

## 5.12.2 Define Documentation

### 5.12.2.1 `#define GLOBUS_FTP_CLIENT_RESTART_PLUGIN_MODULE (&globus_i_ftp_client_restart_plugin_module)`

Module descriptor.

## 5.12.3 Function Documentation

### 5.12.3.1 `globus_result_t globus_ftp_client_restart_plugin_init (globus_ftp_client_plugin_t * plugin, int max_retries, globus_reftime_t * interval, globus_abstime_t * deadline)`

Initialize an instance of the GridFTP restart plugin

This function will initialize the plugin-specific instance data for this plugin, and will make the plugin usable for ftp client handle attribute and handle creation.

#### Parameters:

***plugin*** A pointer to an uninitialized plugin. The plugin will be configured as a restart plugin.

***max\_retries*** The maximum number of times to retry the operation before giving up on the transfer. If this value is less than or equal to 0, then the restart plugin will keep trying to restart the operation until it completes or the deadline is reached with an unsuccessful operation.

***interval*** The interval to wait after a failures before retrying the transfer. If the interval is 0 seconds or GLOBUS\_NULL, then an exponential backoff will be used.

***deadline*** An absolute timeout. If the deadline is GLOBUS\_NULL then the retry will never timeout.

#### Returns:

This function returns an error if

- plugin is null

#### See also:

`globus_ftp_client_restart_plugin_destroy()` (p. 101), `globus_ftp_client_handleattr_add_plugin()` (p. 15), `globus_ftp_client_handleattr_remove_plugin()` (p. 18), `globus_ftp_client_handle_init()` (p. 8)

### 5.12.3.2 `globus_result_t globus_ftp_client_restart_plugin_destroy (globus_ftp_client_plugin_t * plugin)`

Destroy an instance of the GridFTP restart plugin

This function will free all restart plugin-specific instance data from this plugin, and will make the plugin unusable for further ftp handle creation.

Existing FTP client handles and handle attributes will not be affected by destroying a plugin associated with them, as a local copy of the plugin is made upon handle initialization.

## Parameters:

*plugin* A pointer to a GridFTP restart plugin, previously initialized by calling **globus\_ftp\_client\_restart\_plugin\_init()** (p. 101)

## Returns:

This function returns an error if

- plugin is null
- plugin is not a restart plugin

## See also:

**globus\_ftp\_client\_restart\_plugin\_init()** (p. 101), **globus\_ftp\_client\_handleattr\_add\_plugin()** (p. 15), **globus\_ftp\_client\_handleattr\_remove\_plugin()** (p. 18), **globus\_ftp\_client\_handle\_init()** (p. 8)

## 5.13 Netlogger Throughput Plugin

Collaboration diagram for Netlogger Throughput Plugin:



This plugin allows a user to easily use the throughput plugin to log performance data vi Netlogger.

## Defines

- **#define GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_NL\_PLUGIN\_MODULE (&globus\_i\_ftp\_client\_throughput\_nl\_plugin\_module)**

## Functions

- **globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_init (globus\_ftp\_client\_plugin\_t \*plugin, const char \*nl\_url, const char \*prog\_name, const char \*opaque\_string)**
- **globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_init\_with\_handle (globus\_ftp\_client\_plugin\_t \*plugin, NLhandle \*nl\_handle, const char \*opaque\_string)**
- **globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_destroy (globus\_ftp\_client\_plugin\_t \*plugin)**
- **globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_set\_callbacks (globus\_ftp\_client\_plugin\_t \*plugin, globus\_ftp\_client\_throughput\_plugin\_begin\_cb\_t begin\_cb, globus\_ftp\_client\_throughput\_plugin\_stripe\_cb\_t per\_stripe\_cb, globus\_ftp\_client\_throughput\_plugin\_total\_cb\_t total\_cb, globus\_ftp\_client\_throughput\_plugin\_complete\_cb\_t complete\_cb, void \*user\_specific)**

### 5.13.1 Detailed Description

This plugin allows a user to easily use the throughput plugin to log performance data vi Netlogger.

The plugin will log the following Event Types with its coressponding info

TransferPerfTotal : This event type will be sent everytime a throughput plugin total callback is received.

- URL.SOURCE <string> Source url of transfer
- URL.DEST <string> Dest url of transfer

- BYTES <int> Total bytes transfered thus far
- BW.CURRENT <float> Current (instantaneous) bandwidth
- BW.AVG <float> Average (instantaneous) bandwidth

TransferPerfStripe : This event type will be sent everytime a throughput plugin stripe callback is received.

- URL.SOURCE <string> Source url of transfer
- URL.DEST <string> Dest url of transfer
- INDEX <int> The stripe index the event applies to
- BYTES <int> Total bytes transfered thus far on this stripe
- BW.CURRENT <float> Current (instantaneous) bandwidth on this stripe
- BW.AVG <float> Average (instantaneous) bandwidth on this stripe

TransferBegin : This event type will be sent everytime a throughput plugin begin callback is received.

- URL.SOURCE <string> Source url of transfer
- URL.DEST <string> Dest url of transfer

TransferEnd : This event type will be sent everytime a throughput plugin complete callback is received.

- SUCCESS <bool> Completion status

### 5.13.2 Define Documentation

#### 5.13.2.1 #define GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_NL\_PLUGIN\_MODULE (&globus\_i\_ftp\_client\_throughput\_nl\_plugin\_module)

Module descriptor.

### 5.13.3 Function Documentation

#### 5.13.3.1 globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_init (globus\_ftp\_client\_plugin\_t \* *plugin*, const char \* *nl\_url*, const char \* *prog\_name*, const char \* *opaque\_string*)

Initialize netlogger wrapped throughput plugin

This will initialize a netlogger wrapped throughput plugin.

Note that the *nl\_url* may be NULL. Regardless of what *nl\_host* is set to, if the env variable NL\_DEST\_ENV is set, logging will always occur to that location.

#### Parameters:

*plugin* a plugin to be initialized

*nl\_url* the url to log to (May be NULL) Valid urls are: file://tmp/netlog.log  
x-netlog://host[:port] x-syslog://localhost

*prog\_name* This is used as the prog name in the NetLoggerOpen call

*opaque\_string* this is an opaque string that will be inserted into all logged statements. (may be NULL)



**Returns:**

- Error on NULL plugin or failure to init throughput plugin
- Error on NetLogger open
- GLOBUS\_SUCCESS

**5.13.3.2 globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_init\_with\_handle (globus\_ftp\_client\_plugin\_t \* *plugin*, NLhandle \* *nl\_handle*, const char \* *opaque\_string*)**

Initialize netlogger wrapped throughput plugin

This will initialize a netlogger wrapped throughput plugin.

Instead of passing a NetLogger url as in the plain init func, you can pass in a previously 'Open'ed NLhandle. This handle will not be destroyed by this plugin.

**Parameters:**

*plugin* a plugin to be initialized

*nl\_handle* a previously opened NetLogger handle

*opaque\_string* this is an opaque string that will be inserted into all logged statements. (may be NULL)

**Returns:**

- Error on NULL plugin or failure to init throughput plugin
- Error on NetLogger open
- GLOBUS\_SUCCESS

**5.13.3.3 globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_destroy (globus\_ftp\_client\_plugin\_t \* *plugin*)**

Destroy netlogger wrapped throughput plugin

Frees up memory associated with plugin.

**Parameters:**

*plugin* plugin previously initialized with init (above)

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin

**5.13.3.4 globus\_result\_t globus\_ftp\_client\_throughput\_nl\_plugin\_set\_callbacks (globus\_ftp\_client\_plugin\_t \* *plugin*, globus\_ftp\_client\_throughput\_plugin\_begin\_cb\_t *begin\_cb*, globus\_ftp\_client\_throughput\_plugin\_stripe\_cb\_t *per\_stripe\_cb*, globus\_ftp\_client\_throughput\_plugin\_total\_cb\_t *total\_cb*, globus\_ftp\_client\_throughput\_plugin\_complete\_cb\_t *complete\_cb*, void \* *user\_specific*)**

Receive throughput callbacks

You can still get the automatic netlogging of throughput along with receiving the same throughput callbacks that the throughput plugin provides by using this function to set these callbacks.

Note that the callbacks are defined the same as in the throughput plugin

## Parameters:

*plugin*

*begin\_cb* the callback to be called upon the start of a transfer

*per\_stripe\_cb* the callback to be called every time updated throughput info is available for a given stripe

*total\_cb* the callback to be called every time updated throughput info is available for any stripe

*complete\_cb* the callback to be called to indicate transfer completion

*user\_specific* a pointer to some user specific data that will be provided to all callbacks

## Returns:

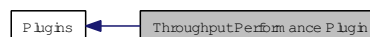
- Error on NULL or invalid plugin
- GLOBUS\_SUCCESS

## See also:

Throughput Performance Plugin (p. 105)

## 5.14 Throughput Performance Plugin

Collaboration diagram for Throughput Performance Plugin:



The FTP Throughput Performance plugin allows the user to obtain calculated performance information for all types of transfers except a third party transfer in which Extended Block mode is not enabled.

## Defines

- `#define GLOBUS_FTP_CLIENT_THROUGHPUT_PLUGIN_MODULE (&globus_i_ftp_client_throughput_plugin_module)`

## Typedefs

- `typedef void(*) globus_ftp_client_throughput_plugin_begin_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url)`
- `typedef void(*) globus_ftp_client_throughput_plugin_stripe_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, int stripe_ndx, globus_off_t bytes, float instantaneous_throughput, float avg_throughput)`
- `typedef void(*) globus_ftp_client_throughput_plugin_total_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, globus_off_t bytes, float instantaneous_throughput, float avg_throughput)`
- `typedef void(*) globus_ftp_client_throughput_plugin_complete_cb_t (void *user_specific, globus_ftp_client_handle_t *handle, globus_bool_t success)`
- `typedef void(*) globus_ftp_client_throughput_plugin_user_copy_cb_t (void *user_specific)`
- `typedef void(*) globus_ftp_client_throughput_plugin_user_destroy_cb_t (void *user_specific)`

## Functions

- `globus_result_t globus_ftp_client_throughput_plugin_init (globus_ftp_client_plugin_t *plugin, globus_ftp_client_throughput_plugin_begin_cb_t begin_cb, globus_ftp_client_throughput_plugin_stripe_cb_t per_stripe_cb, globus_ftp_client_throughput_plugin_total_cb_t total_cb, globus_ftp_client_throughput_plugin_complete_cb_t complete_cb, void *user_specific)`

- `globus_result_t globus_ftp_client_throughput_plugin_set_copy_destroy (globus_ftp_client_plugin_t *plugin, globus_ftp_client_throughput_plugin_user_copy_cb_t copy_cb, globus_ftp_client_throughput_plugin_user_destroy_cb_t destroy_cb)`
- `globus_result_t globus_ftp_client_throughput_plugin_destroy (globus_ftp_client_plugin_t *plugin)`
- `globus_result_t globus_ftp_client_throughput_plugin_get_user_specific (globus_ftp_client_plugin_t *plugin, void **user_specific)`

### 5.14.1 Detailed Description

The FTP Throughput Performance plugin allows the user to obtain calculated performance information for all types of transfers except a third party transfer in which Extended Block mode is not enabled.

Note: Since this plugin is built on top of the Performance Marker Plugin, it is not possible to associate both plugins with a handle

### 5.14.2 Define Documentation

#### 5.14.2.1 `#define GLOBUS_FTP_CLIENT_THROUGHPUT_PLUGIN_MODULE (&globus_i_ftp_client_throughput_plugin_module)`

Module descriptor.

### 5.14.3 Typedef Documentation

#### 5.14.3.1 `typedef void(*) globus_ftp_client_throughput_plugin_begin_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, const char *source_url, const char *dest_url)`

Transfer begin callback

This callback will be called when a transfer begins.

#### Parameters:

*handle* The client handle associated with this transfer

*user\_specific* User argument passed to `globus_ftp_client_throughput_plugin_init`

*source\_url* source of the transfer (GLOBUS\_NULL if 'put')

*dest\_url* dest of the transfer (GLOBUS\_NULL if 'get')

#### Returns:

- n/a

#### 5.14.3.2 `typedef void(*) globus_ftp_client_throughput_plugin_stripe_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, int stripe_ndx, globus_off_t bytes, float instantaneous_throughput, float avg_throughput)`

Stripe performance throughput callback

This callback will be called with every performance callback that is received by the perf plugin.

The first callback for each `stripe_ndx` will have an `instantaneous_throughput` based from the time the command was sent.

#### Parameters:

*handle* The client handle associated with this transfer

*user\_specific* User argument passed to globus\_ftp\_client\_throughput\_plugin\_init

*bytes* The total number of bytes received on this stripe

*instantaneous\_throughput* Instantaneous throughput on this stripe (bytes / sec)

*avg\_throughput* Average throughput on this stripe (bytes / sec)

*stripe\_ndx* This stripe's index

**5.14.3.3** `typedef void(*) globus_ftp_client_throughput_plugin_total_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, globus_off_t bytes, float instantaneous_throughput, float avg_throughput)`

Total performance throughput callback

This callback will be called with every performance callback that is received by the perf plugin.

The first callback for will have an instantaneous\_throughput based from the time the command was sent. This callback will be called after the per\_stripe\_cb

**Parameters:**

*handle* The client handle associated with this transfer

*user\_specific* User argument passed to globus\_ftp\_client\_throughput\_plugin\_init

*bytes* The total number of bytes received on all stripes

*instantaneous\_throughput* Total instantaneous throughput on all stripes (bytes / sec)

*avg\_throughput* Average total throughput on all stripes (bytes / sec)

**5.14.3.4** `typedef void(*) globus_ftp_client_throughput_plugin_complete_cb_t(void *user_specific, globus_ftp_client_handle_t *handle, globus_bool_t success)`

Transfer complete callback

This callback will be called upon transfer completion (successful or otherwise).

**Parameters:**

*handle* The client handle associated with this transfer

*user\_specific* User argument passed to globus\_ftp\_client\_throughput\_plugin\_init

*success* indicates whether this transfer completed successfully or was interrupted (by error or abort)

**Returns:**

- n/a

**5.14.3.5** `typedef void(*) globus_ftp_client_throughput_plugin_user_copy_cb_t(void *user_specific)`

Copy constructor

This callback will be called when a copy of this plugin is made, it is intended to allow initialization of a new user\_specific data.

**Parameters:**

*user\_specific* this is user specific data either created by this copy method, or the value passed to init

**Returns:**

- a pointer to a user specific piece of data
- GLOBUS\_NULL (does not indicate error)

#### 5.14.3.6 `typedef void(*) globus_ftp_client_throughput_plugin_user_destroy_cb_t(void *user_specific)`

Destructor

This callback will be called when a copy of this plugin is destroyed, it is intended to allow the user to free up any memory associated with the user specific data.

##### Parameters:

*user\_specific* this is user specific data created by the copy method

##### Returns:

- n/a

#### 5.14.4 Function Documentation

**5.14.4.1** `globus_result_t globus_ftp_client_throughput_plugin_init (globus_ftp_client_plugin_t * plugin, globus_ftp_client_throughput_plugin_begin_cb_t begin_cb, globus_ftp_client_throughput_plugin_stripe_cb_t per_stripe_cb, globus_ftp_client_throughput_plugin_total_cb_t total_cb, globus_ftp_client_throughput_plugin_complete_cb_t complete_cb, void * user_specific)`

Throughput plugin init

Use this function to initialize a throughput plugin.

The throughput plugin sits on top of the perf\_plugin. The only required param is 'plugin', all others may be GLOBUS\_NULL

##### Parameters:

*plugin* a pointer to a plugin type to be initialized

*begin\_cb* the callback to be called upon the start of a transfer

*per\_stripe\_cb* the callback to be called every time updated throughput info is available for a given stripe

*total\_cb* the callback to be called every time updated throughput info is available for any stripe

*complete\_cb* the callback to be called to indicate transfer completion

*user\_specific* a pointer to some user specific data that will be provided to all callbacks

##### Returns:

- GLOBUS\_SUCCESS
- Error on NULL plugin
- Error on init perf plugin

**5.14.4.2** `globus_result_t globus_ftp_client_throughput_plugin_set_copy_destroy (globus_ftp_client_plugin_t * plugin, globus_ftp_client_throughput_plugin_user_copy_cb_t copy_cb, globus_ftp_client_throughput_plugin_user_destroy_cb_t destroy_cb)`

Set user copy and destroy callbacks

Use this to have the plugin make callbacks any time a copy of this plugin is being made.

This will allow the user to keep state for different handles.

##### Parameters:

*plugin* plugin previously initialized with init (above)

*copy\_cb* func to be called when a copy is needed

*destroy\_cb* func to be called when a copy is to be destroyed

**Returns:**

- Error on NULL arguments
- GLOBUS\_SUCCESS

**5.14.4.3 globus\_result\_t globus\_ftp\_client\_throughput\_plugin\_destroy (globus\_ftp\_client\_plugin\_t \* *plugin*)**

Destroy throughput plugin

Frees up memory associated with plugin.

**Parameters:**

*plugin* plugin previously initialized with init (above)

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin

**5.14.4.4 globus\_result\_t globus\_ftp\_client\_throughput\_plugin\_get\_user\_specific (globus\_ftp\_client\_plugin\_t \* *plugin*, void \*\* *user\_specific*)**

Retrieve user specific pointer.

**Parameters:**

*plugin* plugin previously initialized with init (above)

*user\_specific* pointer to storage for user\_specific pointer

**Returns:**

- GLOBUS\_SUCCESS
- Error on NULL plugin
- Error on NULL user\_specific

## **6 globus ftp client Data Structure Documentation**

### **6.1 globus\_ftp\_client\_restart\_extended\_block\_t Struct Reference**

Extended block mode restart marker.

#### **6.1.1 Detailed Description**

Extended block mode restart marker.

### **6.2 globus\_ftp\_client\_restart\_marker\_t Union Reference**

Restart marker.

### 6.2.1 Detailed Description

Restart marker.

This structure is may be either a stream mode transfer offset, or an extended block mode byte range.

See also:

`globus_ftp_client_restart_marker_init()` (p. 4), `globus_ftp_client_restart_marker_destroy()` (p. 4),  
`globus_ftp_client_restart_marker_copy()` (p. 4), `globus_ftp_client_restart_marker_insert_range()`  
(p. 5), `globus_ftp_client_restart_marker_set_offset()` (p. 6)

## 6.3 `globus_ftp_client_restart_stream_t` Struct Reference

Stream mode restart marker.

### 6.3.1 Detailed Description

Stream mode restart marker.

## 7 `globus_ftp_client` Page Documentation

### 7.1 Bug List

**Global `globus_ftp_client_operationattr_set_data_protection` (p. 51)** Only safe and private protection levels are supported by gsiftp.

**Global `globus_ftp_client_operationattr_set_control_protection` (p. 51)** The clear and safe protection levels are treated identically, with the client integrity checking all commands. The confidential and private protection levels are treated identically, with the client encrypting all commands.

## Index

Activation, 2

Debugging Plugin, 61

FTP Operation Attributes, 41

FTP Operations, 18

- globus\_ftp\_client\_abort
  - globus\_ftp\_client\_operations, 41
- globus\_ftp\_client\_activation
  - GLOBUS\_FTP\_CLIENT\_MODULE, 3
- globus\_ftp\_client\_chgrp
  - globus\_ftp\_client\_operations, 31
- globus\_ftp\_client\_chmod
  - globus\_ftp\_client\_operations, 31
- globus\_ftp\_client\_cksm
  - globus\_ftp\_client\_operations, 40
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_ALL
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_BUFFER
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_-  
CONTROL\_ESTABLISHMENT
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_DATA\_-  
ESTABLISHMENT
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_FILE\_-  
ACTIONS
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_-  
INFORMATION
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_MISC
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_-  
TRANSFER\_MODIFIERS
  - globus\_ftp\_client\_plugins, 85
- GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_-  
TRANSFER\_PARAMETERS
  - globus\_ftp\_client\_plugins, 85
- globus\_ftp\_client\_complete\_callback\_t
  - globus\_ftp\_client\_operations, 21
- globus\_ftp\_client\_cwd
  - globus\_ftp\_client\_operations, 25
- globus\_ftp\_client\_data
  - globus\_ftp\_client\_data\_callback\_t, 60
  - globus\_ftp\_client\_register\_read, 60
  - globus\_ftp\_client\_register\_write, 61
- globus\_ftp\_client\_data\_callback\_t
  - globus\_ftp\_client\_data, 60
- globus\_ftp\_client\_debug\_plugin
  - globus\_ftp\_client\_debug\_plugin\_destroy, 63

- globus\_ftp\_client\_debug\_plugin\_init, 62
- GLOBUS\_FTP\_CLIENT\_DEBUG\_PLUGIN\_-  
MODULE, 62
- globus\_ftp\_client\_debug\_plugin\_destroy
  - globus\_ftp\_client\_debug\_plugin, 63
- globus\_ftp\_client\_debug\_plugin\_init
  - globus\_ftp\_client\_debug\_plugin, 62
- GLOBUS\_FTP\_CLIENT\_DEBUG\_PLUGIN\_-  
MODULE
  - globus\_ftp\_client\_debug\_plugin, 62
- globus\_ftp\_client\_delete
  - globus\_ftp\_client\_operations, 26
- globus\_ftp\_client\_exists
  - globus\_ftp\_client\_operations, 22
- globus\_ftp\_client\_extended\_get
  - globus\_ftp\_client\_operations, 34
- globus\_ftp\_client\_extended\_put
  - globus\_ftp\_client\_operations, 36
- globus\_ftp\_client\_extended\_third\_party\_transfer
  - globus\_ftp\_client\_operations, 38
- globus\_ftp\_client\_feat
  - globus\_ftp\_client\_operations, 23
- globus\_ftp\_client\_features\_destroy
  - globus\_ftp\_client\_operations, 23
- globus\_ftp\_client\_features\_init
  - globus\_ftp\_client\_operations, 22
- globus\_ftp\_client\_features\_t
  - globus\_ftp\_client\_operations, 21
- globus\_ftp\_client\_get
  - globus\_ftp\_client\_operations, 33
- globus\_ftp\_client\_handle
  - globus\_ftp\_client\_handle\_add\_plugin, 9
  - globus\_ftp\_client\_handle\_cache\_url\_state, 8
  - globus\_ftp\_client\_handle\_destroy, 8
  - globus\_ftp\_client\_handle\_flush\_url\_state, 9
  - globus\_ftp\_client\_handle\_get\_user\_pointer, 10
  - globus\_ftp\_client\_handle\_init, 8
  - globus\_ftp\_client\_handle\_remove\_plugin, 9
  - globus\_ftp\_client\_handle\_set\_user\_pointer, 9
  - globus\_ftp\_client\_handle\_t, 8
- globus\_ftp\_client\_handle\_add\_plugin
  - globus\_ftp\_client\_handle, 9
- globus\_ftp\_client\_handle\_cache\_url\_state
  - globus\_ftp\_client\_handle, 8
- globus\_ftp\_client\_handle\_destroy
  - globus\_ftp\_client\_handle, 8
- globus\_ftp\_client\_handle\_flush\_url\_state
  - globus\_ftp\_client\_handle, 9
- globus\_ftp\_client\_handle\_get\_user\_pointer
  - globus\_ftp\_client\_handle, 10
- globus\_ftp\_client\_handle\_init
  - globus\_ftp\_client\_handle, 8
- globus\_ftp\_client\_handle\_remove\_plugin



- globus\_ftp\_client\_handle, 9
- globus\_ftp\_client\_handle\_set\_user\_pointer
  - globus\_ftp\_client\_handle, 9
- globus\_ftp\_client\_handle\_t
  - globus\_ftp\_client\_handle, 8
- globus\_ftp\_client\_handleattr
  - globus\_ftp\_client\_handleattr\_add\_cached\_url, 14
  - globus\_ftp\_client\_handleattr\_add\_plugin, 15
  - globus\_ftp\_client\_handleattr\_copy, 13
  - globus\_ftp\_client\_handleattr\_destroy, 12
  - globus\_ftp\_client\_handleattr\_get\_cache\_all, 15
  - globus\_ftp\_client\_handleattr\_get\_clientinfo, 16
  - globus\_ftp\_client\_handleattr\_get\_gridftp2, 16
  - globus\_ftp\_client\_handleattr\_get\_pipeline, 17
  - globus\_ftp\_client\_handleattr\_get\_rfc1738\_url, 16
  - globus\_ftp\_client\_handleattr\_init, 12
  - globus\_ftp\_client\_handleattr\_remove\_cached\_url, 15
  - globus\_ftp\_client\_handleattr\_remove\_plugin, 17
  - globus\_ftp\_client\_handleattr\_set\_cache\_all, 13
  - globus\_ftp\_client\_handleattr\_set\_clientinfo, 13
  - globus\_ftp\_client\_handleattr\_set\_gridftp2, 14
  - globus\_ftp\_client\_handleattr\_set\_netlogger, 15
  - globus\_ftp\_client\_handleattr\_set\_netlogger\_ftp\_io, 17
  - globus\_ftp\_client\_handleattr\_set\_pipeline, 14
  - globus\_ftp\_client\_handleattr\_set\_rfc1738\_url, 13
  - globus\_ftp\_client\_handleattr\_t, 12
- globus\_ftp\_client\_handleattr\_add\_cached\_url
  - globus\_ftp\_client\_handleattr, 14
- globus\_ftp\_client\_handleattr\_add\_plugin
  - globus\_ftp\_client\_handleattr, 15
- globus\_ftp\_client\_handleattr\_copy
  - globus\_ftp\_client\_handleattr, 13
- globus\_ftp\_client\_handleattr\_destroy
  - globus\_ftp\_client\_handleattr, 12
- globus\_ftp\_client\_handleattr\_get\_cache\_all
  - globus\_ftp\_client\_handleattr, 15
- globus\_ftp\_client\_handleattr\_get\_clientinfo
  - globus\_ftp\_client\_handleattr, 16
- globus\_ftp\_client\_handleattr\_get\_gridftp2
  - globus\_ftp\_client\_handleattr, 16
- globus\_ftp\_client\_handleattr\_get\_pipeline
  - globus\_ftp\_client\_handleattr, 17
- globus\_ftp\_client\_handleattr\_get\_rfc1738\_url
  - globus\_ftp\_client\_handleattr, 16
- globus\_ftp\_client\_handleattr\_init
  - globus\_ftp\_client\_handleattr, 12
- globus\_ftp\_client\_handleattr\_remove\_cached\_url
  - globus\_ftp\_client\_handleattr, 15
- globus\_ftp\_client\_handleattr\_remove\_plugin
  - globus\_ftp\_client\_handleattr, 17
- globus\_ftp\_client\_handleattr\_set\_cache\_all
  - globus\_ftp\_client\_handleattr, 13
- globus\_ftp\_client\_handleattr\_set\_clientinfo
  - globus\_ftp\_client\_handleattr, 13
- globus\_ftp\_client\_handleattr\_set\_gridftp2
  - globus\_ftp\_client\_handleattr, 14
- globus\_ftp\_client\_handleattr\_set\_netlogger
  - globus\_ftp\_client\_handleattr, 15
- globus\_ftp\_client\_handleattr\_set\_netlogger\_ftp\_io
  - globus\_ftp\_client\_handleattr, 17
- globus\_ftp\_client\_handleattr\_set\_pipeline
  - globus\_ftp\_client\_handleattr, 14
- globus\_ftp\_client\_handleattr\_set\_rfc1738\_url
  - globus\_ftp\_client\_handleattr, 13
- globus\_ftp\_client\_handleattr\_t
  - globus\_ftp\_client\_handleattr, 12
- globus\_ftp\_client\_is\_feature\_supported
  - globus\_ftp\_client\_operations, 24
- globus\_ftp\_client\_list
  - globus\_ftp\_client\_operations, 26
- globus\_ftp\_client\_machine\_list
  - globus\_ftp\_client\_operations, 28
- globus\_ftp\_client\_mkdir
  - globus\_ftp\_client\_operations, 24
- globus\_ftp\_client\_mlst
  - globus\_ftp\_client\_operations, 29
- globus\_ftp\_client\_modification\_time
  - globus\_ftp\_client\_operations, 39
- GLOBUS\_FTP\_CLIENT\_MODULE
  - globus\_ftp\_client\_activation, 3
- globus\_ftp\_client\_move
  - globus\_ftp\_client\_operations, 30
- globus\_ftp\_client\_operationattr
  - globus\_ftp\_client\_operationattr\_copy, 52
  - globus\_ftp\_client\_operationattr\_destroy, 44
  - globus\_ftp\_client\_operationattr\_get\_allocate, 53
  - globus\_ftp\_client\_operationattr\_get\_allow\_ipv6, 58
  - globus\_ftp\_client\_operationattr\_get\_append, 58
  - globus\_ftp\_client\_operationattr\_get\_authorization, 57
  - globus\_ftp\_client\_operationattr\_get\_authz\_assert, 54
  - globus\_ftp\_client\_operationattr\_get\_control\_protection, 58
  - globus\_ftp\_client\_operationattr\_get\_data\_protection, 57
  - globus\_ftp\_client\_operationattr\_get\_data\_security, 58
  - globus\_ftp\_client\_operationattr\_get\_dcav, 57
  - globus\_ftp\_client\_operationattr\_get\_disk\_stack, 53
  - globus\_ftp\_client\_operationattr\_get\_layout, 54
  - globus\_ftp\_client\_operationattr\_get\_list\_uses\_data\_mode, 56
  - globus\_ftp\_client\_operationattr\_get\_mode, 56
  - globus\_ftp\_client\_operationattr\_get\_net\_stack, 52
  - globus\_ftp\_client\_operationattr\_get\_parallelism, 53
  - globus\_ftp\_client\_operationattr\_get\_read\_all, 59

globus_ftp_client_operationattr_get_storage_- module, 52	globus_ftp_client_operationattr_get_control_- protection
globus_ftp_client_operationattr_get_stripped, 54	globus_ftp_client_operationattr, 58
globus_ftp_client_operationattr_get_tcp_buffer, 55	globus_ftp_client_operationattr_get_data_protection globus_ftp_client_operationattr, 57
globus_ftp_client_operationattr_get_type, 55	globus_ftp_client_operationattr_get_data_security globus_ftp_client_operationattr, 58
globus_ftp_client_operationattr_init, 44	globus_ftp_client_operationattr_get_dcau globus_ftp_client_operationattr, 57
globus_ftp_client_operationattr_set_allocate, 46	globus_ftp_client_operationattr_get_disk_stack globus_ftp_client_operationattr, 53
globus_ftp_client_operationattr_set_allow_ipv6, 51	globus_ftp_client_operationattr_get_layout globus_ftp_client_operationattr, 54
globus_ftp_client_operationattr_set_append, 51	globus_ftp_client_operationattr_get_list_uses_data_- mode
globus_ftp_client_operationattr_set_- authorization, 49	globus_ftp_client_operationattr, 56
globus_ftp_client_operationattr_set_authz_assert, 46	globus_ftp_client_operationattr_get_mode globus_ftp_client_operationattr, 56
globus_ftp_client_operationattr_set_control_- protection, 50	globus_ftp_client_operationattr_get_net_stack globus_ftp_client_operationattr, 52
globus_ftp_client_operationattr_set_data_- protection, 50	globus_ftp_client_operationattr_get_parallelism globus_ftp_client_operationattr, 53
globus_ftp_client_operationattr_set_data_- security, 50	globus_ftp_client_operationattr_get_read_all globus_ftp_client_operationattr, 59
globus_ftp_client_operationattr_set_dcau, 50	globus_ftp_client_operationattr_get_storage_module globus_ftp_client_operationattr, 52
globus_ftp_client_operationattr_set_delayed_- pasv, 49	globus_ftp_client_operationattr_get_stripped globus_ftp_client_operationattr, 54
globus_ftp_client_operationattr_set_disk_stack, 45	globus_ftp_client_operationattr_get_tcp_buffer globus_ftp_client_operationattr, 55
globus_ftp_client_operationattr_set_layout, 47	globus_ftp_client_operationattr_get_type globus_ftp_client_operationattr, 55
globus_ftp_client_operationattr_set_list_uses_- data_mode, 49	globus_ftp_client_operationattr_init globus_ftp_client_operationattr, 44
globus_ftp_client_operationattr_set_mode, 48	globus_ftp_client_operationattr_set_allocate globus_ftp_client_operationattr, 46
globus_ftp_client_operationattr_set_net_stack, 45	globus_ftp_client_operationattr_set_allow_ipv6 globus_ftp_client_operationattr, 51
globus_ftp_client_operationattr_set_parallelism, 46	globus_ftp_client_operationattr_set_append globus_ftp_client_operationattr, 51
globus_ftp_client_operationattr_set_read_all, 51	globus_ftp_client_operationattr_set_authorization globus_ftp_client_operationattr, 49
globus_ftp_client_operationattr_set_storage_- module, 45	globus_ftp_client_operationattr_set_authz_assert globus_ftp_client_operationattr, 46
globus_ftp_client_operationattr_set_stripped, 46	globus_ftp_client_operationattr_set_control_protection globus_ftp_client_operationattr, 50
globus_ftp_client_operationattr_set_tcp_buffer, 47	globus_ftp_client_operationattr_set_data_protection globus_ftp_client_operationattr, 50
globus_ftp_client_operationattr_set_type, 48	globus_ftp_client_operationattr_set_data_security globus_ftp_client_operationattr, 50
globus_ftp_client_operationattr_t, 44	globus_ftp_client_operationattr_set_dcau globus_ftp_client_operationattr, 50
globus_ftp_client_operationattr_copy globus_ftp_client_operationattr, 52	globus_ftp_client_operationattr_set_delayed_pasv globus_ftp_client_operationattr, 49
globus_ftp_client_operationattr_destroy globus_ftp_client_operationattr, 44	globus_ftp_client_operationattr_set_disk_stack globus_ftp_client_operationattr, 45
globus_ftp_client_operationattr_get_allocate globus_ftp_client_operationattr, 53	
globus_ftp_client_operationattr_get_allow_ipv6 globus_ftp_client_operationattr, 58	
globus_ftp_client_operationattr_get_append globus_ftp_client_operationattr, 58	
globus_ftp_client_operationattr_get_authorization globus_ftp_client_operationattr, 57	
globus_ftp_client_operationattr_get_authz_assert globus_ftp_client_operationattr, 54	

- globus\_ftp\_client\_operationattr\_set\_layout
  - globus\_ftp\_client\_operationattr, 47
- globus\_ftp\_client\_operationattr\_set\_list\_uses\_data\_-mode
  - globus\_ftp\_client\_operationattr, 49
- globus\_ftp\_client\_operationattr\_set\_mode
  - globus\_ftp\_client\_operationattr, 48
- globus\_ftp\_client\_operationattr\_set\_net\_stack
  - globus\_ftp\_client\_operationattr, 45
- globus\_ftp\_client\_operationattr\_set\_parallelism
  - globus\_ftp\_client\_operationattr, 46
- globus\_ftp\_client\_operationattr\_set\_read\_all
  - globus\_ftp\_client\_operationattr, 51
- globus\_ftp\_client\_operationattr\_set\_storage\_module
  - globus\_ftp\_client\_operationattr, 45
- globus\_ftp\_client\_operationattr\_set\_striped
  - globus\_ftp\_client\_operationattr, 46
- globus\_ftp\_client\_operationattr\_set\_tcp\_buffer
  - globus\_ftp\_client\_operationattr, 47
- globus\_ftp\_client\_operationattr\_set\_type
  - globus\_ftp\_client\_operationattr, 48
- globus\_ftp\_client\_operationattr\_t
  - globus\_ftp\_client\_operationattr, 44
- globus\_ftp\_client\_operations
  - globus\_ftp\_client\_abort, 41
  - globus\_ftp\_client\_chgrp, 31
  - globus\_ftp\_client\_chmod, 31
  - globus\_ftp\_client\_cksm, 40
  - globus\_ftp\_client\_complete\_callback\_t, 21
  - globus\_ftp\_client\_cwd, 25
  - globus\_ftp\_client\_delete, 26
  - globus\_ftp\_client\_exists, 22
  - globus\_ftp\_client\_extended\_get, 34
  - globus\_ftp\_client\_extended\_put, 36
  - globus\_ftp\_client\_extended\_third\_party\_transfer, 38
  - globus\_ftp\_client\_feat, 23
  - globus\_ftp\_client\_features\_destroy, 23
  - globus\_ftp\_client\_features\_init, 22
  - globus\_ftp\_client\_features\_t, 21
  - globus\_ftp\_client\_get, 33
  - globus\_ftp\_client\_is\_feature\_supported, 24
  - globus\_ftp\_client\_list, 26
  - globus\_ftp\_client\_machine\_list, 28
  - globus\_ftp\_client\_mkdir, 24
  - globus\_ftp\_client\_mlst, 29
  - globus\_ftp\_client\_modification\_time, 39
  - globus\_ftp\_client\_move, 30
  - globus\_ftp\_client\_partial\_get, 34
  - globus\_ftp\_client\_partial\_put, 36
  - globus\_ftp\_client\_partial\_third\_party\_transfer, 37
  - globus\_ftp\_client\_probed\_feature\_t, 22
  - globus\_ftp\_client\_put, 35
  - globus\_ftp\_client\_recursive\_list, 29
  - globus\_ftp\_client\_rmdir, 24
  - globus\_ftp\_client\_size, 39
  - globus\_ftp\_client\_stat, 27
  - globus\_ftp\_client\_symlink, 32
  - globus\_ftp\_client\_third\_party\_transfer, 37
  - globus\_ftp\_client\_tristate\_t, 22
  - globus\_ftp\_client\_utime, 32
  - globus\_ftp\_client\_verbose\_list, 27
- globus\_ftp\_client\_partial\_get
  - globus\_ftp\_client\_operations, 34
- globus\_ftp\_client\_partial\_put
  - globus\_ftp\_client\_operations, 36
- globus\_ftp\_client\_partial\_third\_party\_transfer
  - globus\_ftp\_client\_operations, 37
- globus\_ftp\_client\_perf\_plugin
  - globus\_ftp\_client\_perf\_plugin\_begin\_cb\_t, 64
  - globus\_ftp\_client\_perf\_plugin\_complete\_cb\_t, 65
  - globus\_ftp\_client\_perf\_plugin\_destroy, 67
  - globus\_ftp\_client\_perf\_plugin\_get\_user\_specific, 67
  - globus\_ftp\_client\_perf\_plugin\_init, 66
  - globus\_ftp\_client\_perf\_plugin\_marker\_cb\_t, 65
  - GLOBUS\_FTP\_CLIENT\_PERF\_PLUGIN\_MODULE, 64
  - globus\_ftp\_client\_perf\_plugin\_set\_copy\_destroy, 67
  - globus\_ftp\_client\_perf\_plugin\_user\_copy\_cb\_t, 66
  - globus\_ftp\_client\_perf\_plugin\_user\_destroy\_cb\_t, 66
- globus\_ftp\_client\_perf\_plugin\_begin\_cb\_t
  - globus\_ftp\_client\_perf\_plugin, 64
- globus\_ftp\_client\_perf\_plugin\_complete\_cb\_t
  - globus\_ftp\_client\_perf\_plugin, 65
- globus\_ftp\_client\_perf\_plugin\_destroy
  - globus\_ftp\_client\_perf\_plugin, 67
- globus\_ftp\_client\_perf\_plugin\_get\_user\_specific
  - globus\_ftp\_client\_perf\_plugin, 67
- globus\_ftp\_client\_perf\_plugin\_init
  - globus\_ftp\_client\_perf\_plugin, 66
- globus\_ftp\_client\_perf\_plugin\_marker\_cb\_t
  - globus\_ftp\_client\_perf\_plugin, 65
- GLOBUS\_FTP\_CLIENT\_PERF\_PLUGIN\_MODULE
  - globus\_ftp\_client\_perf\_plugin, 64
- globus\_ftp\_client\_perf\_plugin\_set\_copy\_destroy
  - globus\_ftp\_client\_perf\_plugin, 67
- globus\_ftp\_client\_perf\_plugin\_user\_copy\_cb\_t
  - globus\_ftp\_client\_perf\_plugin, 66
- globus\_ftp\_client\_perf\_plugin\_user\_destroy\_cb\_t
  - globus\_ftp\_client\_perf\_plugin, 66
- globus\_ftp\_client\_plugin\_abort
  - globus\_ftp\_client\_plugins, 95
- globus\_ftp\_client\_plugin\_abort\_t
  - globus\_ftp\_client\_plugins, 82
- globus\_ftp\_client\_plugin\_add\_data\_channels
  - globus\_ftp\_client\_plugins, 95
- globus\_ftp\_client\_plugin\_authenticate\_t
  - globus\_ftp\_client\_plugins, 73

globus_ftp_client_plugin_chgrp_t	globus_ftp_client_plugin_restart_delete
globus_ftp_client_plugins, 74	globus_ftp_client_plugins, 90
globus_ftp_client_plugin_chmod_t	globus_ftp_client_plugin_restart_feat
globus_ftp_client_plugins, 73	globus_ftp_client_plugins, 90
globus_ftp_client_plugin_cksm_t	globus_ftp_client_plugin_restart_get
globus_ftp_client_plugins, 75	globus_ftp_client_plugins, 92
globus_ftp_client_plugin_command_mask_t	globus_ftp_client_plugin_restart_get_marker
globus_ftp_client_plugins, 85	globus_ftp_client_plugins, 95
globus_ftp_client_plugin_command_t	globus_ftp_client_plugin_restart_list
globus_ftp_client_plugins, 83	globus_ftp_client_plugins, 85
globus_ftp_client_plugin_complete_t	globus_ftp_client_plugin_restart_machine_list
globus_ftp_client_plugins, 84	globus_ftp_client_plugins, 86
globus_ftp_client_plugin_connect_t	globus_ftp_client_plugin_restart_mkdir
globus_ftp_client_plugins, 73	globus_ftp_client_plugins, 91
globus_ftp_client_plugin_copy_t	globus_ftp_client_plugin_restart_mlst
globus_ftp_client_plugins, 72	globus_ftp_client_plugins, 87
globus_ftp_client_plugin_data_t	globus_ftp_client_plugin_restart_modification_time
globus_ftp_client_plugins, 83	globus_ftp_client_plugins, 94
globus_ftp_client_plugin_delete_t	globus_ftp_client_plugin_restart_move
globus_ftp_client_plugins, 75	globus_ftp_client_plugins, 92
globus_ftp_client_plugin_destroy_t	globus_ftp_client_plugin_restart_put
globus_ftp_client_plugins, 72	globus_ftp_client_plugins, 93
globus_ftp_client_plugin_fault_t	globus_ftp_client_plugin_restart_recursive_list
globus_ftp_client_plugins, 84	globus_ftp_client_plugins, 86
globus_ftp_client_plugin_feat_t	globus_ftp_client_plugin_restart_rmdir
globus_ftp_client_plugins, 76	globus_ftp_client_plugins, 91
globus_ftp_client_plugin_get_t	globus_ftp_client_plugin_restart_size
globus_ftp_client_plugins, 80	globus_ftp_client_plugins, 94
globus_ftp_client_plugin_list_t	globus_ftp_client_plugin_restart_stat
globus_ftp_client_plugins, 77	globus_ftp_client_plugins, 87
globus_ftp_client_plugin_machine_list_t	globus_ftp_client_plugin_restart_symlink
globus_ftp_client_plugins, 78	globus_ftp_client_plugins, 89
globus_ftp_client_plugin_mkdir_t	globus_ftp_client_plugin_restart_third_party_transfer
globus_ftp_client_plugins, 76	globus_ftp_client_plugins, 93
globus_ftp_client_plugin_mlst_t	globus_ftp_client_plugin_restart_utime
globus_ftp_client_plugins, 78	globus_ftp_client_plugins, 88
globus_ftp_client_plugin_modification_time_t	globus_ftp_client_plugin_restart_verbose_list
globus_ftp_client_plugins, 81	globus_ftp_client_plugins, 85
globus_ftp_client_plugin_move_t	globus_ftp_client_plugin_rmdir_t
globus_ftp_client_plugins, 79	globus_ftp_client_plugins, 76
globus_ftp_client_plugin_put_t	globus_ftp_client_plugin_size_t
globus_ftp_client_plugins, 80	globus_ftp_client_plugins, 81
globus_ftp_client_plugin_read_t	globus_ftp_client_plugin_stat_t
globus_ftp_client_plugins, 82	globus_ftp_client_plugins, 79
globus_ftp_client_plugin_recursive_list_t	globus_ftp_client_plugin_symlink_t
globus_ftp_client_plugins, 78	globus_ftp_client_plugins, 74
globus_ftp_client_plugin_remove_data_channels	globus_ftp_client_plugin_t
globus_ftp_client_plugins, 96	globus_ftp_client_plugins, 72
globus_ftp_client_plugin_response_t	globus_ftp_client_plugin_third_party_transfer_t
globus_ftp_client_plugins, 83	globus_ftp_client_plugins, 80
globus_ftp_client_plugin_restart_chgrp	globus_ftp_client_plugin_utime_t
globus_ftp_client_plugins, 88	globus_ftp_client_plugins, 74
globus_ftp_client_plugin_restart_chmod	globus_ftp_client_plugin_verbose_list_t
globus_ftp_client_plugins, 88	globus_ftp_client_plugins, 77
globus_ftp_client_plugin_restart_cksm	globus_ftp_client_plugin_write_t
globus_ftp_client_plugins, 89	globus_ftp_client_plugins, 82

- globus\_ftp\_client\_plugins
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_ALL, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_BUFFER, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_CONTROL\_ESTABLISHMENT, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_DATA\_ESTABLISHMENT, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_FILE\_ACTIONS, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_INFORMATION, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_MISC, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_TRANSFER\_MODIFIERS, 85
  - GLOBUS\_FTP\_CLIENT\_CMD\_MASK\_TRANSFER\_PARAMETERS, 85
  - globus\_ftp\_client\_plugin\_abort, 95
  - globus\_ftp\_client\_plugin\_abort\_t, 82
  - globus\_ftp\_client\_plugin\_add\_data\_channels, 95
  - globus\_ftp\_client\_plugin\_authenticate\_t, 73
  - globus\_ftp\_client\_plugin\_chgrp\_t, 74
  - globus\_ftp\_client\_plugin\_chmod\_t, 73
  - globus\_ftp\_client\_plugin\_cksm\_t, 75
  - globus\_ftp\_client\_plugin\_command\_mask\_t, 85
  - globus\_ftp\_client\_plugin\_command\_t, 83
  - globus\_ftp\_client\_plugin\_complete\_t, 84
  - globus\_ftp\_client\_plugin\_connect\_t, 73
  - globus\_ftp\_client\_plugin\_copy\_t, 72
  - globus\_ftp\_client\_plugin\_data\_t, 83
  - globus\_ftp\_client\_plugin\_delete\_t, 75
  - globus\_ftp\_client\_plugin\_destroy\_t, 72
  - globus\_ftp\_client\_plugin\_fault\_t, 84
  - globus\_ftp\_client\_plugin\_feat\_t, 76
  - globus\_ftp\_client\_plugin\_get\_t, 80
  - globus\_ftp\_client\_plugin\_list\_t, 77
  - globus\_ftp\_client\_plugin\_machine\_list\_t, 78
  - globus\_ftp\_client\_plugin\_mkdir\_t, 76
  - globus\_ftp\_client\_plugin\_mlst\_t, 78
  - globus\_ftp\_client\_plugin\_modification\_time\_t, 81
  - globus\_ftp\_client\_plugin\_move\_t, 79
  - globus\_ftp\_client\_plugin\_put\_t, 80
  - globus\_ftp\_client\_plugin\_read\_t, 82
  - globus\_ftp\_client\_plugin\_recursive\_list\_t, 78
  - globus\_ftp\_client\_plugin\_remove\_data\_channels, 96
  - globus\_ftp\_client\_plugin\_response\_t, 83
  - globus\_ftp\_client\_plugin\_restart\_chgrp, 88
  - globus\_ftp\_client\_plugin\_restart\_chmod, 88
  - globus\_ftp\_client\_plugin\_restart\_cksm, 89
  - globus\_ftp\_client\_plugin\_restart\_delete, 90
  - globus\_ftp\_client\_plugin\_restart\_feat, 90
  - globus\_ftp\_client\_plugin\_restart\_get, 92
  - globus\_ftp\_client\_plugin\_restart\_get\_marker, 95
  - globus\_ftp\_client\_plugin\_restart\_list, 85
  - globus\_ftp\_client\_plugin\_restart\_machine\_list, 86
  - globus\_ftp\_client\_plugin\_restart\_mkdir, 91
  - globus\_ftp\_client\_plugin\_restart\_mlst, 87
  - globus\_ftp\_client\_plugin\_restart\_modification\_time, 94
  - globus\_ftp\_client\_plugin\_restart\_move, 92
  - globus\_ftp\_client\_plugin\_restart\_put, 93
  - globus\_ftp\_client\_plugin\_restart\_recursive\_list, 86
  - globus\_ftp\_client\_plugin\_restart\_rmdir, 91
  - globus\_ftp\_client\_plugin\_restart\_size, 94
  - globus\_ftp\_client\_plugin\_restart\_stat, 87
  - globus\_ftp\_client\_plugin\_restart\_symlink, 89
  - globus\_ftp\_client\_plugin\_restart\_third\_party\_transfer, 93
  - globus\_ftp\_client\_plugin\_restart\_utime, 88
  - globus\_ftp\_client\_plugin\_restart\_verbose\_list, 85
  - globus\_ftp\_client\_plugin\_rmdir\_t, 76
  - globus\_ftp\_client\_plugin\_size\_t, 81
  - globus\_ftp\_client\_plugin\_stat\_t, 79
  - globus\_ftp\_client\_plugin\_symlink\_t, 74
  - globus\_ftp\_client\_plugin\_t, 72
  - globus\_ftp\_client\_plugin\_third\_party\_transfer\_t, 80
  - globus\_ftp\_client\_plugin\_utime\_t, 74
  - globus\_ftp\_client\_plugin\_verbose\_list\_t, 77
  - globus\_ftp\_client\_plugin\_write\_t, 82
  - globus\_ftp\_client\_probed\_feature\_t
  - globus\_ftp\_client\_operations, 22
  - globus\_ftp\_client\_put
    - globus\_ftp\_client\_operations, 35
  - globus\_ftp\_client\_recursive\_list
    - globus\_ftp\_client\_operations, 29
  - globus\_ftp\_client\_register\_read
    - globus\_ftp\_client\_data, 60
  - globus\_ftp\_client\_register\_write
    - globus\_ftp\_client\_data, 61
  - globus\_ftp\_client\_restart\_extended\_block\_t, 109
  - globus\_ftp\_client\_restart\_marker
    - globus\_ftp\_client\_restart\_marker\_copy, 4
    - globus\_ftp\_client\_restart\_marker\_destroy, 4
    - globus\_ftp\_client\_restart\_marker\_from\_string, 6
    - globus\_ftp\_client\_restart\_marker\_get\_total, 6
    - globus\_ftp\_client\_restart\_marker\_init, 3
    - globus\_ftp\_client\_restart\_marker\_insert\_range, 4
    - globus\_ftp\_client\_restart\_marker\_set\_ascii\_offset, 5
    - globus\_ftp\_client\_restart\_marker\_set\_offset, 5
    - globus\_ftp\_client\_restart\_marker\_to\_string, 6
  - globus\_ftp\_client\_restart\_marker\_copy
    - globus\_ftp\_client\_restart\_marker, 4
  - globus\_ftp\_client\_restart\_marker\_destroy
    - globus\_ftp\_client\_restart\_marker, 4
  - globus\_ftp\_client\_restart\_marker\_from\_string
    - globus\_ftp\_client\_restart\_marker, 6

- globus\_ftp\_client\_restart\_marker\_get\_total
  - globus\_ftp\_client\_restart\_marker, 6
- globus\_ftp\_client\_restart\_marker\_init
  - globus\_ftp\_client\_restart\_marker, 3
- globus\_ftp\_client\_restart\_marker\_insert\_range
  - globus\_ftp\_client\_restart\_marker, 4
- globus\_ftp\_client\_restart\_marker\_plugin
  - globus\_ftp\_client\_restart\_marker\_plugin\_begin\_-cb\_t, 97
  - globus\_ftp\_client\_restart\_marker\_plugin\_-complete\_cb\_t, 98
  - globus\_ftp\_client\_restart\_marker\_plugin\_destroy, 99
  - globus\_ftp\_client\_restart\_marker\_plugin\_init, 98
  - globus\_ftp\_client\_restart\_marker\_plugin\_-marker\_cb\_t, 98
  - GLOBUS\_FTP\_CLIENT\_RESTART\_-MARKER\_PLUGIN\_MODULE, 97
- globus\_ftp\_client\_restart\_marker\_plugin\_begin\_cb\_t
  - globus\_ftp\_client\_restart\_marker\_plugin, 97
- globus\_ftp\_client\_restart\_marker\_plugin\_complete\_-cb\_t
  - globus\_ftp\_client\_restart\_marker\_plugin, 98
- globus\_ftp\_client\_restart\_marker\_plugin\_destroy
  - globus\_ftp\_client\_restart\_marker\_plugin, 99
- globus\_ftp\_client\_restart\_marker\_plugin\_init
  - globus\_ftp\_client\_restart\_marker\_plugin, 98
- globus\_ftp\_client\_restart\_marker\_plugin\_marker\_cb\_t
  - globus\_ftp\_client\_restart\_marker\_plugin, 98
- GLOBUS\_FTP\_CLIENT\_RESTART\_MARKER\_-PLUGIN\_MODULE
  - globus\_ftp\_client\_restart\_marker\_plugin, 97
  - globus\_ftp\_client\_restart\_marker\_set\_ascii\_offset
    - globus\_ftp\_client\_restart\_marker, 5
  - globus\_ftp\_client\_restart\_marker\_set\_offset
    - globus\_ftp\_client\_restart\_marker, 5
  - globus\_ftp\_client\_restart\_marker\_t, 109
  - globus\_ftp\_client\_restart\_marker\_to\_string
    - globus\_ftp\_client\_restart\_marker, 6
  - globus\_ftp\_client\_restart\_plugin
    - globus\_ftp\_client\_restart\_plugin\_destroy, 101
    - globus\_ftp\_client\_restart\_plugin\_init, 101
  - GLOBUS\_FTP\_CLIENT\_RESTART\_PLUGIN\_-MODULE, 101
- globus\_ftp\_client\_restart\_plugin\_destroy
  - globus\_ftp\_client\_restart\_plugin, 101
- globus\_ftp\_client\_restart\_plugin\_init
  - globus\_ftp\_client\_restart\_plugin, 101
- GLOBUS\_FTP\_CLIENT\_RESTART\_PLUGIN\_-MODULE
  - globus\_ftp\_client\_restart\_plugin, 101
- globus\_ftp\_client\_restart\_stream\_t, 110
- globus\_ftp\_client\_rmdir
  - globus\_ftp\_client\_operations, 24
- globus\_ftp\_client\_size
  - globus\_ftp\_client\_operations, 39
- globus\_ftp\_client\_stat
  - globus\_ftp\_client\_operations, 27
- globus\_ftp\_client\_symlink
  - globus\_ftp\_client\_operations, 32
- globus\_ftp\_client\_third\_party\_transfer
  - globus\_ftp\_client\_operations, 37
- globus\_ftp\_client\_throughput\_nl\_plugin
  - globus\_ftp\_client\_throughput\_nl\_plugin\_destroy, 104
  - globus\_ftp\_client\_throughput\_nl\_plugin\_init, 103
  - globus\_ftp\_client\_throughput\_nl\_plugin\_init\_-with\_handle, 103
  - GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_-NL\_PLUGIN\_MODULE, 103
  - globus\_ftp\_client\_throughput\_nl\_plugin\_set\_-callbacks, 104
- globus\_ftp\_client\_throughput\_nl\_plugin\_destroy
  - globus\_ftp\_client\_throughput\_nl\_plugin, 104
- globus\_ftp\_client\_throughput\_nl\_plugin\_init
  - globus\_ftp\_client\_throughput\_nl\_plugin, 103
- globus\_ftp\_client\_throughput\_nl\_plugin\_init\_with\_-handle
  - globus\_ftp\_client\_throughput\_nl\_plugin, 103
- GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_NL\_-PLUGIN\_MODULE
  - globus\_ftp\_client\_throughput\_nl\_plugin, 103
  - globus\_ftp\_client\_throughput\_nl\_plugin\_set\_callbacks
    - globus\_ftp\_client\_throughput\_nl\_plugin, 104
- globus\_ftp\_client\_throughput\_plugin
  - globus\_ftp\_client\_throughput\_plugin\_begin\_cb\_t, 106
  - globus\_ftp\_client\_throughput\_plugin\_complete\_-cb\_t, 107
  - globus\_ftp\_client\_throughput\_plugin\_destroy, 109
  - globus\_ftp\_client\_throughput\_plugin\_get\_user\_-specific, 109
  - globus\_ftp\_client\_throughput\_plugin\_init, 108
  - GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_-PLUGIN\_MODULE, 106
  - globus\_ftp\_client\_throughput\_plugin\_set\_copy\_-destroy, 108
  - globus\_ftp\_client\_throughput\_plugin\_stripe\_cb\_t, 106
  - globus\_ftp\_client\_throughput\_plugin\_total\_cb\_t, 106
  - globus\_ftp\_client\_throughput\_plugin\_user\_-copy\_cb\_t, 107
  - globus\_ftp\_client\_throughput\_plugin\_user\_-destroy\_cb\_t, 107
- globus\_ftp\_client\_throughput\_plugin\_begin\_cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 106
- globus\_ftp\_client\_throughput\_plugin\_complete\_cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 107
- globus\_ftp\_client\_throughput\_plugin\_destroy
  - globus\_ftp\_client\_throughput\_plugin, 109

- globus\_ftp\_client\_throughput\_plugin\_get\_user\_-specific
  - globus\_ftp\_client\_throughput\_plugin, 109
- globus\_ftp\_client\_throughput\_plugin\_init
  - globus\_ftp\_client\_throughput\_plugin, 108
- GLOBUS\_FTP\_CLIENT\_THROUGHPUT\_PLUGIN\_MODULE
  - globus\_ftp\_client\_throughput\_plugin, 106
- globus\_ftp\_client\_throughput\_plugin\_set\_copy\_-destroy
  - globus\_ftp\_client\_throughput\_plugin, 108
- globus\_ftp\_client\_throughput\_plugin\_stripe\_cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 106
- globus\_ftp\_client\_throughput\_plugin\_total\_cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 106
- globus\_ftp\_client\_throughput\_plugin\_user\_copy\_cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 107
- globus\_ftp\_client\_throughput\_plugin\_user\_destroy\_-cb\_t
  - globus\_ftp\_client\_throughput\_plugin, 107
- globus\_ftp\_client\_tristate\_t
  - globus\_ftp\_client\_operations, 22
- globus\_ftp\_client\_utime
  - globus\_ftp\_client\_operations, 32
- globus\_ftp\_client\_verbose\_list
  - globus\_ftp\_client\_operations, 27

- Handle Attributes, 10
- Handle Management, 7

- Netlogger Throughput Plugin, 102

- Performance Marker Plugin, 63
- Plugins, 68

- Reading and Writing Data, 59
- Restart Marker Plugin, 96
- Restart Markers, 3
- Restart Plugin, 99

- Throughput Performance Plugin, 105