

globus ftp client Reference Manual

3.14

Generated by Doxygen 1.2.18

Fri Jun 26 15:51:47 2009

Contents

1	Globus FTP Client API	1
2	globus ftp client Module Index	1
3	globus ftp client Data Structure Index	2
4	globus ftp client Page Index	2
5	globus ftp client Module Documentation	2
6	globus ftp client Data Structure Documentation	97
7	globus ftp client Page Documentation	97

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](#) 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 "globusclient.h" and activate the [GLOBUS.FTP.CLIENT MODULE](#).

To use the Globus FTP Client API, one must create an [FTP Client handle](#). 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's Attributes](#) 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](#) 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 [FTP Operation Attributes](#) 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](#) 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 [Reading and Writing Data](#) section of the manual.

2 globus ftp client Module Index

2.1 globus ftp client Modules

Here is a list of all modules:

Activation	2
------------	-------------------

Restart Markers	3
Handle Management	7
Handle Attributes	10
FTP Operations	17
FTP Operation Attributes	36
Reading and Writing Data	53
Plugins	61
Debugging Plugin	55
Performance Marker Plugin	57
Restart Marker Plugin	85
Restart Plugin	88
Netlogger Throughput Plugin	90
Throughput Performance Plugin	93

3 globus ftp client Data Structure Index

3.1 globus ftp client Data Structures

Here are the data structures with brief descriptions:

globus.ftp_client_restart_extendedblock_t (Extended block mode restart marker)	97
globus.ftp_client_restart_marker_t (Restart marker)	97
globus.ftp_client_restart_stream_t (Stream mode restart marker)	97

4 globus ftp client Page Index

4.1 globus ftp client Related Pages

Here is a list of all related documentation pages:

Bug List	97
----------	----

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.

De nes

```
#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 De ne Documentation

5.1.2.1 #define GLOBUSFTP_CLIENT_MODULE

Module descriptor.

5.2 Restart Markers

FTP Restart Markers.

Data Structures

```
union globusftp_client_restartmarkert
    Restart marker.
```

Functions

```
globus_result_t globusftp_client_restartmarkerinit (globusftp_client_restartmarkert marker)
globus_result_t globusftp_client_restartmarkercopy (globusftp_client_restartmarkert new_marker, globus-
ftp_client_restartmarkert marker)
globus_result_t globusftp_client_restartmarkerdestroy (globusftp_client_restartmarkert marker)
globus_result_t globusftp_client_restartmarkerinsert_range (globusftp_client_restartmarkert marker,
globus_off_t offset, globus_off_t end_offset)
globus_result_t globusftp_client_restartmarker_set_ascii_offset (globusftp_client_restartmarkert marker,
globus_off_t offset, globus_off_t ascii_offset)
globus_result_t globusftp_client_restartmarker_set_offset (globusftp_client_restartmarkert marker, globus-
off_t offset)
globus_result_t globusftp_client_restartmarker_get_total (globusftp_client_restartmarkert marker, globus-
off_t total_bytes)
globus_result_t globusftp_client_restartmarkerto_string (globusftp_client_restartmarkert marker, char
marker_string)
```

```
globusresult_t globusftp_client_restartmarkerfrom_string (globusftp_client_restartmarker_t marker, const
char markerstring)
```

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 [globusftp_client_get\(\)](#), [globusftp_client_put\(\)](#), or [globusftp_client-third_party_transfer\(\)](#) functions.

5.2.2 Function Documentation

5.2.2.1 `globusresult_t globusftp_client_restart_marker_init (globusftp_client_restart_marker_t marker)`

Initialize a restart marker.

Parameters:

marker New restart marker.

See also:

[globusftp_client_restartmarkert](#), [globusftp_client_restartmarkerdestroy\(\)](#)

5.2.2.2 `globusresult_t globusftp_client_restart_marker_copy (globusftp_client_restart_marker_t new-marker, globusftp_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:

[globusftp_client_restartmarkerinit\(\)](#), [globusftp_client_restartmarkerdestroy\(\)](#)

5.2.2.3 `globusresult_t globusftp_client_restart_marker_destroy (globusftp_client_restart_marker_t marker)`

Destroy a restart marker.

Parameters:

marker Restart marker. This marker must be initialized by either calling [globusftp_client_restartmarkerinit\(\)](#) or [globusftp_client_restartmarker.copy\(\)](#)

See also:

[globusftp_client_restartmarkert](#), [globusftp_client_restartmarkerinit\(\)](#), [globusftp_client_restartmarker.copy\(\)](#)

5.2.2.4 `globusresult_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_restartmarkerinit()` or `globus_ftp_client_restartmarkercopy()`.

A marker can only hold a range list or a stream offset. Calling this function after calling `globus_ftp_client_restartmarker_set_offset()` 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_restartmarker_set_offset()` `globus_ftp_client_operationatt_setmode()`

5.2.2.5 `globusresult_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_restartmarkerinit()` or `globus_ftp_client_restartmarkercopy()`.

A marker can only hold a range list or a stream offset. Calling this function after calling `globus_ftp_client_restartmarkerinsert_range()` 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_restartmarker_set_ascii_offset()` 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_restartmarkerinsert_range()` `globus_ftp_client_restartmarker_set_offset()` `globus_ftp_client_operationatt_setmode()` `globus_ftp_client_operationatt_settype()`

5.2.2.6 `globusresult_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 [globusftp_client_restartmarkerinit\(\)](#) or [globusftp_client_restartmarkercopy\(\)](#).

A marker can only hold a range list or a stream offset. Calling this function after calling [globusftp_client_restartmarkerinsertrange\(\)](#) 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:

[globusftp_client_restartmarkerinsertrange\(\)](#) [globusftp_client_operationattsetmode\(\)](#) [globusftp_client_operationattsettype\(\)](#)

5.2.2.7 `globusresult_t globusftp_client_restart_marker_gettotal (globusftp_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 is 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 `globusresult_t globusftp_client_restart_marker_to_string (globusftp_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 [globusftp_client_restartmarkerfromstring\(\)](#).

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](#)

5.2.2.9 `globusresult_t globusftp_client_restart_marker_from_string (globusftp_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 initialized.

`marker_string` The string containing a textual representation of a restart marker.

See also:

[Restart Markers](#)

5.3 Handle Management

Create/Destroy/Modify an FTP Client Handle.

Initialize

```
globusresult_t globusftp_client_handleinit (globusftp_client_handle_t handle, globusftp_client_handleattr_t attr)
```

Destroy

```
globusresult_t globusftp_client_handledestroy(globusftp_client_handle_t handle)
```

URL Caching

```
globusresult_t globusftp_client_handlecacheurl_state(globusftp_client_handle_t handle, const char url)
globusresult_t globusftp_client_handle_ush_url_state(globusftp_client_handle_t handle, const char url)
```

User Pointer

```
globusresult_t globusftp_client_handlesetuserpointer (globusftp_client_handle_t handle, void userpointer)
globusresult_t globusftp_client_handlegetuserpointer (const globusftp_client_handle_t handle, void userpointer)
```

Plugins

```
globusresult_t globusftp_client_handleaddplugin (globusftp_client_handle_t handle, globusftp_client_plugin_t plugin)
globusresult_t globusftp_client_handlerremoveplugin (globusftp_client_handle_t handle, globusftp_client_plugin_t plugin)
```

Typedefs

```
typedef globusftp_client_handle_t globusftp_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_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 associated with them. All FTP operations take a handle pointer as a parameter.

See also:

[globusftp_client_handleinit\(\)](#), [globusftp_client_handledestroy\(\)](#) [globusftp_client_handleattr](#)

5.3.3 Function Documentation

5.3.3.1 globusresult_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:

[globusftp_client_handledestroy\(\)](#)

5.3.3.2 globusresult_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:

[globusftp_client_handleinit\(\)](#)

5.3.3.3 `globusresult_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_usch_url_state()`

5.3.3.4 `globusresult_t globus_ftp_client_handle_usch_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 `globusresult_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 its value.

5.3.3.6 `globusresult_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:

globusftp_client_removeplugin(), globusftp_client_handleattraddplugin(), globusftp_client_handleattr_removeplugin()

5.3.3.7 globusresult_t globusftp_client_handle_remove_plugin (globusftp_client_handle_t handle, globusftp_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 by calling `globusftp_client_handleinit()` "handle initialization time" or by calling `globusftp_client_handleaddplugin()`.

Parameters:

handle The FTP client handle to set or query.

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

See also:

globusftp_client_addplugin(), globusftp_client_handleattraddplugin(), globusftp_client_handleattr_removeplugin()

5.3.3.8 globusresult_t globusftp_client_handle_get_user_pointer (const globusftp_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 its value.

5.4 Handle Attributes

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

Initialize

globusresult_t globusftp_client_handleattrinit (globusftp_client_handleattr_t attr)

Destroy

```
globusresultt globusftp_client_handleattrdestroy(globusftp_client_handleattr attr)
```

Copy

```
globusresultt globusftp_client_handleattrcopy (globusftp_client_handleattr dest, globusftp_client_handleattr src)
```

Connection Caching

```
globusresultt globusftp_client_handleattrset.cacheall (globusftp_client_handleattr attr, globusbool_t cacheall)
globusresultt globusftp_client_handleattrget.cacheall (const globusftp_client_handleattr attr, globusbool_t cacheall)
```

Non-root relative URLs

```
globusresultt globusftp_client_handleattrset.rfc1738.url (globusftp_client_handleattr attr, globusbool_t rfc1738.url)
globusresultt globusftp_client_handleattrget.rfc1738.url (const globusftp_client_handleattr attr, globusbool_t rfc1738.url)
```

GridFTP2 support

```
globusresultt globusftp_client_handleattrset.gridftp2 (globusftp_client_handleattr attr, globusbool_t gridftp2)
globusresultt globusftp_client_handleattrget.gridftp2 (const globusftp_client_handleattr attr, globusbool_t gridftp2)
```

Command Pipelining

```
globusresultt globusftp_client_handleattrset.pipeline(globusftp_client_handleattr attr, globussize_t outstandingcommands, globusftp_client_pipeline_callbackt pipeline.callback, void pipeline.arg)
globusresultt globusftp_client_handleattrget.pipeline (const globusftp_client_handleattr attr, globussize_t outstandingcommands, globusftp_client_pipeline_callbackt pipeline.callback, void pipeline.arg)
```

URL Caching

```
globusresultt globusftp_client_handleattradd.cachedurl (globusftp_client_handleattr attr, const char url)
globusresultt globusftp_client_handleattrremove.cachedurl (globusftp_client_handleattr attr, const char url)
```

Netlogger management

```
globusresultt globusftp_client_handleattrset.netlogger (globusftp_client_handleattr attr, globusnetlogger_handle_t nl_handle)
globusresultt globusftp_client_handleattrset.netloggerftp_io (globusftp_client_handleattr attr, globusnetlogger_handle_t nl_handle, globusbool_t ftp, globusbool_t io)
```

Plugin Management

```
globusresult_t globusftp_client_handleattraddplugin (globusftp_client_handleattr_t attr, globusftp_client-
plugin_t plugin)
globusresult_t globusftp_client_handleattrremoveplugin (globusftp_client_handleattr_t attr, globusftp-
client_plugin_t plugin)
```

Typedefs

```
typedef globusftp_client_handleattr_t globusftp_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:

[globusftp_client_handlet](#)

5.4.2 Typedef Documentation

5.4.2.1 typedef struct globusftp_client_handleattr_t globusftp_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:

[globusftp_client_handlet](#), [Handle Attributes](#)

5.4.3 Function Documentation

5.4.3.1 globusresult_t globusftp_client_handleattr_init (globusftp_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:

[globusftp_client_handleattrdestroy\(\)](#)

5.4.3.2 globusresult_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_handleattrinit\(\)](#) 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 globusresult_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 globusresult_t globus_ftp_client_handleattr_set_cacheall ([globus_ftp_client_handleattr_t](#) attr, globus_bool_t cacheall)

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 no cached caching is needed, then the user should disable this attribute and explicitly cache specific URLs.

Parameters:

attr Attribute to query or modify.

cacheall Value of the cacheall attribute.

See also:

[globus_ftp_client_handleattradd_cachedurl\(\)](#), [globus_ftp_client_handleattrremove_cachedurl\(\)](#), [globus_ftp_client_handlecacheurl.state\(\)](#), [globus_ftp_client_handle_url.state\(\)](#)

5.4.3.5 globusresult_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 GLOBUSTRUE to enable non-root relative URLs. Default of GLOBUSTRUE specifies root-relative URLs.

5.4.3.6 `globusresult_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.

Parameters:

attr Attribute to modify

gridftp2 Set to GLOBUSTRUE to enable GridFTP2 support. Default of GLOBUSTRUE specifies that GridFTP is disabled.

5.4.3.7 `globusresult_t globus.ftp_client.handleattr_set.pipeline (globus.ftp_client.handleattr_t attr, globus_size_t outstandingcommands, globus.ftp_client.pipeline_callback_t pipeline_callback, void pipeline_arg)`

Enable/Disable command queueing for pipelined transfers.

Parameters:

attr Attribute to modify

outstandingcommands 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.

5.4.3.8 `globusresult_t globus.ftp_client.handleattr_add_cachedurl (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

Parameters:

attr Attribute to modify

url URL string to cache

5.4.3.9 `globusresult_t globus.ftp_client.handleattr_remove_cachedurl (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

Parameters:

attr Attribute to modify

url URL string to cache

5.4.3.10 `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.

5.4.3.11 `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 plugin's '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.

5.4.3.12 `globus_result_t globus_ftp_client_handleattr_get_cacheall (const globus_ftp_client_handleattr_t attr, globus_bool_t cacheall)`

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.

`cacheall` Value of the cacheall attribute.

See also:

`globus_ftp_client_handleattradd_cachedurl()`, `globus_ftp_client_handleattrremove_cachedurl()`, `globus_ftp_client_handlecacheurl_state()`, `globus_ftp_client_handle_url_state()`

5.4.3.13 `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 GLOBUSTRUE to enable non-root relative URLs. Default of GLOBUSTRUE specifies root-relative URLs.

5.4.3.14 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.

Parameters:

attr Attribute to modify

gridftp2 Set to GLOBUSTRUE to enable GridFTP2 support. Default of GLOBUSTRUE specifies that GridFTP is disabled.

5.4.3.15 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.

5.4.3.16 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.

5.4.3.17 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 plugin's '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

```
globusresultt globusftp_client_exists (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Features

```
globusresultt globusftp_client_featuresinit (globusftp_client_featurest u_features)
globusresultt globusftp_client_featuresdestroy(globusftp_client_featurest u_features)
globusresultt globusftp_client_feat (globusftp_client_handle_t u_handle, char url, globusftp_client_operationattr_t attr, globusftp_client_featurest u_features, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresultt globusftp_client_is_featuresupported(const globusftp_client_featurest u_features, globusftp_client_tristatet answer, globusftp_client_probedfeature_t feature)
```

Make Directory

```
globusresultt globusftp_client_mkdir (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Remove Directory

```
globusresultt globusftp_client_rmdir (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Delete

```
globusresultt globusftp_client_delete (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

List

```
globusresultt globusftp_client_list (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresultt globusftp_client_verbose_list (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

STAT

```
globusresult_t globusftp_client_stat(globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusbyte_t statbuffer, globussize_t statbuffer_length, globusftp_client_completercallback_t completercallback, void callbackarg)
```

MLST

```
globusresult_t globusftp_client_mlst(globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusbyte_t mlstbuffer, globussize_t mlstbuffer_length, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Move

```
globusresult_t globusftp_client_move(globusftp_client_handle_t u_handle, const char sourceurl, const char desturl, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

chmod

```
globusresult_t globusftp_client_chmod(globusftp_client_handle_t u_handle, const char url, int mode, globusftp_client_operationattr_t attr, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Get

```
globusresult_t globusftp_client_get(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresult_t globusftp_client_partial_get(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, globusoff_t partialOffset, globusoff_t partialEndOffset, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresult_t globusftp_client_extended_get(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, const char etag.str, globusftp_client_completercallback_t completercallback, void callbackarg)
```

Put

```
globusresult_t globusftp_client_put(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresult_t globusftp_client_partial_put(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, globusoff_t partialOffset, globusoff_t partialEndOffset, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresult_t globusftp_client_extended_put(globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restartmarkert restart, const char etag.str, globusftp_client_completercallback_t completercallback, void callbackarg)
```

3rd Party Transfer

```

globusresultt globusftp_client_third_party_transfer(globusftp_client_handle_t handle, const char sourceurl,
globusftp_client_operationattr_t sourceattr, const char desturl, globusftp_client_operationattr_t destattr,
globusftp_client_restartmarkert restart, globusftp_client_completercallback_t completercallback, void callbackarg)
globusresultt globusftp_client_partial_third_party_transfer(globusftp_client_handle_t handle, const char sourceurl,
globusftp_client_operationattr_t sourceattr, const char desturl, globusftp_client_operationattr_t destattr,
globusftp_client_restartmarkert restart, globusoff_t partialOffset, globusoff_t partialEndOffset,
globusftp_client_completercallback_t completercallback, void callbackarg)
globusresultt globusftp_client_extended_third_party_transfer(globusftp_client_handle_t handle, const char sourceurl,
globusftp_client_operationattr_t sourceattr, const char retalg_str, const char desturl, globusftp_client_operationattr_t destattr,
const char estalg_str, globusftp_client_restartmarkert restart, globusftp_client_completercallback_t completercallback, void callbackarg)

```

Modification Time

```

globusresultt globusftp_client_modification_time(globusftp_client_handle_t u_handle, const char url,
globusftp_client_operationattr_t attr, globusabstime_t modification_time, globusftp_client_completercallback_t completercallback, void callbackarg)

```

Size

```

globusresultt globusftp_client_size(globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr,
globusoff_t size, globusftp_client_completercallback_t completercallback, void callbackarg)

```

Cksum

```

globusresultt globusftp_client_cksum(globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr,
char cksum, globusoff_t offset, globusoff_t length, const char algorithm, globusftp_client_completercallback_t completercallback, void callbackarg)

```

Abort

```

globusresultt globusftp_client_abort(globusftp_client_handle_t u_handle)

```

Typedefs

```

typedef void( globusftp_client_completercallback_t )(void userarg, globusftp_client_handle_t handle,
globusobject_t error)
typedef globusftp_client_feature_t globusftp_client_feature_t

```

Enumerations

```

enum globusftp_client_tristate_t
enum globusftp_client_probed_feature_t

```

Functions

`globusresult_t globusftp_client_machinelist (globusftp_client_handle_t u_handle, const char url, globusftp_client_operation_attr_t attr, globusftp_client_completecallback_t completecallback, void callbackarg)`

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 le operation.

5.5.2 Typedef Documentation

5.5.2.1 `typedef void(globusftp_client_completecallback_t)(void user_arg, globusftp_client_handle_t handle, globusobject_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 `userarg` 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 globusftp_client_features_s globusftp_client_features_t`

Feature Handle.

Handle used to associate state with feature operations.

See also:

[globusftp_client_feat](#), [globusftp_client_featuresinit](#), [globusftp_client_featuresdestroy](#)

5.5.3 Enumeration Type Documentation

5.5.3.1 `enum globusftp_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 globusftp_client_probed_feature_t`

Types of features.

5.5.4 Function Documentation

5.5.4.1 `globusresult_t globus_ftp_client_exists (globus_ftp_client_handle_t u_handle, const char url, globus_ftp_client_operationattr_t attr, globus_ftp_client_completecallback_t completecallback, 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 completecallback 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.
- `completecallback` Callback to be invoked once the existence operation is completed.
- `callback_arg` Argument to be passed to the completecallback.

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
- `completecallback` is GLOBUS_NULL
- handle already has an operation in progress

5.5.4.2 `globusresult_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_features_get()`. Each feature gets initial value GLOBUS_FTP_CLIENT_MAYBE.

Note:

Structure initialized by this function must be destroyed using `globus_ftp_client_features_destroy()`

Returns:

GLOBUS_SUCCESS on success, otherwise error.

5.5.4.3 `globusresult_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()`.

Returns:

GLOBUS_SUCCESS on success, otherwise error.

5.5.4.4 `globusresult_t globusftp_client_feat (globusftp_client_handle_t u_handle, char url, globusftp_client_operationattr_t attr, globusftp_client_features_t u_features, globusftp_client_completecallback_t completecallback, 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 `globusftp_client_featuresinit()`; afterwards, it should be destroyed by `globusftp_client_featuresdestroy()`. After `globusftp_client_feat()` returns, each feature in the list has one of the values: `GLOBUSFTP_CLIENT_TRUE`, `GLOBUSFTP_CLIENT_FALSE`, or `GLOBUSFTP_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 `globusftp_client_features_t` to be filled with the feature set supported by the server.
- `completecallback` Callback to be invoked once the size check is completed.
- `callback.arg` Argument to be passed to the `completecallback`.

Returns:

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

- `u_handle` is `GLOBUSNULL`
- `sourceurl` is `GLOBUSNULL`
- `sourceurl` cannot be parsed
- `sourceurl` is not a ftp or gsiftp url
- `u_features` is `GLOBUSNULL` or badly initialized
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

5.5.4.5 `globusresult_t globusftp_client_is_feature_supported (const globusftp_client_features_t u_features, globusftp_client_tristate_t answer, globusftp_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: `GLOBUSFTP_CLIENT_TRUE`, `GLOBUSFTP_CLIENT_FALSE`, or `GLOBUSFTP_CLIENT_MAYBE`.

Parameters:

- `u_features` list of features, as returned by `globusftp_client_feat()`
- `answer` this variable will contain the answer
- `feature` feature number, $0 \leq \text{feature} < \text{GLOBUSFTP_CLIENT_FEATURE_MAX}$

Returns:

- error when any of the parameters is null or badly initialized

5.5.4.6 `globusresult_t globusftp_client_mkdir (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_completecallback_t completecallback, 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the mkdir is completed.
- `callbackarg` Argument to be passed to the `completecallback`.

Returns:

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

- `u_handle` is `GLOBUSNULL`
- `url` is `GLOBUSNULL`
- `url` cannot be parsed
- `url` is not a ftp or gsiftp url
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

5.5.4.7 `globusresult_t globus_ftp_client_rmdir (globus_ftp_client_handle_t u_handle, const char url, globus_ftp_client_operationattr_t attr, globus_ftp_client_completecallback_t completecallback, void callbackarg)`

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 `completecallback` 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.
- `completecallback` Callback to be invoked once the rmdir is completed.
- `callbackarg` Argument to be passed to the `completecallback`.

Returns:

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

- `u_handle` is `GLOBUSNULL`
- `url` is `GLOBUSNULL`
- `url` cannot be parsed
- `url` is not a ftp or gsiftp url
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

5.5.4.8 `globusresult_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 completecallback, 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the delete is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

Returns:

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

- `u_handle` is `GLOBUSNULL`
- `url` is `GLOBUSNULL`
- `url` cannot be parsed
- `url` is not a ftp or gsiftp url
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

5.5.4.9 `globusresult_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 completecallback, 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_registerread()` 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the "list" is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

Returns:

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

- handle is `GLOBUSNULL`
- `url` is `GLOBUSNULL`
- `url` cannot be parsed
- `url` is not a ftp or gsiftp url
- `completecallback` is `GLOBUSNULL`

handle already has an operation in progress

See also:

[globusftp_client_registerread\(\)](#)

5.5.4.10 `globusresult_t globusftp_client_verbose_list (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_complete_callback_t completecallback, 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 [globusftp_client_registerread\(\)](#) 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the "list" is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

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
- `completecallback` is `GLOBUS_NULL`
- handle already has an operation in progress

See also:

[globusftp_client_registerread\(\)](#)

5.5.4.11 `globusresult_t globusftp_client_stat (globusftp_client_handle_t u_handle, const char url, globusftp_client_operationattr_t attr, globus_byte_t statbuffer, globus_size_t statbuffer_length, globusftp_client_complete_callback_t completecallback, 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 `completecallback` 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 `statbuffer` 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 gsiftp 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`.

`completecallback` Callback to be invoked once the STAT command is completed.

`callback.arg` Argument to be passed to the `completecallback`.

Returns:

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

- handle is `GLOBUSNULL`
- url is `GLOBUSNULL`
- url cannot be parsed
- url is not a ftp or gsiftp url
- `stat.buffer` is `GLOBUSNULL`
- `stat.buffer.length` is `GLOBUSNULL`
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

5.5.4.12 `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_completecallback_t completecallback, 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 `globusftp_client_registerread()` 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the "list" is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

Returns:

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

- handle is `GLOBUSNULL`
- url is `GLOBUSNULL`
- url cannot be parsed
- url is not a ftp or gsiftp url
- `completecallback` is `GLOBUSNULL`
- handle already has an operation in progress

See also:

[globusftp_client_registerread\(\)](#)

5.5.4.13 `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 completecallback, 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 `completecallback` 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 `mlstbuffer` 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 gsiftp 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`.
- `completecallback` Callback to be invoked once the MLST command is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

Returns:

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

- handle is GLOBUSNULL
- url is GLOBUSNULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- mlst_buffer is GLOBUSNULL
- mlst_buffer_length is GLOBUSNULL
- completecallback is GLOBUSNULL
- handle already has an operation in progress

5.5.4.14 `globus_result_t globus_ftp_client_move(globus_ftp_client_handle_t u_handle, const char sourceurl, const char desturl, globus_ftp_client_operationattr_t attr, globus_ftp_client_complete_callback_t completecallback, 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 `completecallback` will be invoked with the result of the move operation.

Parameters:

- `u_handle` An FTP Client handle to use for the move operation.
- `sourceurl` The URL for the file to move.
- `desturl` The URL for the target of the move. The host:port part of this URL is ignored.
- `attr` Attributes for this operation.
- `completecallback` Callback to be invoked once the move is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

Returns:

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

- handle is GLOBUS_NULL
- sourceurl is GLOBUS_NULL
- sourceurl cannot be parsed
- sourceurl is not a ftp or gsiftp url
- completecallback is GLOBUS_NULL
- handle already has an operation in progress

5.5.4.15 `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 completecallback, void callback_arg)`

Change permissions on a file.

This function changes file permissions

When the response to the move request has been received the completecallback 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.
- completecallback Callback to be invoked once the move is completed.
- callback_arg Argument to be passed to the completecallback.

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
- completecallback is GLOBUS_NULL
- handle already has an operation in progress

5.5.4.16 `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 completecallback, 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_registerread()` 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 completecallback 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 le transfer.
 restart Pointer to a restart marker.
 completercallback Callback to be invoked once the "get" is completed.
 callback.arg Argument to be passed to the completercallback.

Returns:

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

- handle is GLOBUSNULL
- url is GLOBUSNULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- completercallback is GLOBUSNULL
- handle already has an operation in progress

See also:

[globusftp_client_registerread\(\)](#)

5.5.4.17 globusresult_t globusftp_client_partial_get (globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restart_marker_t restart, globus_off_t partial_offset, globus_off_t partial_end_offset, globusftp_client_completercallback_t completercallback, void callback.arg)

Get a le from an FTP server.

This function starts a "get" le transfer from an FTP server. If this function returns GLOBUS_SUCCESS, then the user may immediately begin calling [globusftp_client_registerread\(\)](#) 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 completercallback will be invoked with the nal 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 le transfer.
- restart Pointer to a restart marker.
- partial_offset Starting offset for a partial le get.
- partial_end_offset Ending offset for a partial le get. Use -1 for EOF.
- completercallback Callback to be invoked once the "get" is completed.
- callback.arg Argument to be passed to the completercallback.

Returns:

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

- handle is GLOBUSNULL
- url is GLOBUSNULL
- url cannot be parsed
- url is not a ftp or gsiftp url
- completercallback is GLOBUSNULL
- handle already has an operation in progress

5.5.4.18 `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 eretalg_str, globus_ftp_client_completecallback_t completecallback, 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()` 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 completecallback will be invoked with the final status of the get.

This function differs from the `globus_ftp_client_get()` 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.

eretalg_str The ERET algorithm string. This string contains information needed to invoke a server-specific data reduction algorithm on the file being retrieved.

completecallback Callback to be invoked once the "get" is completed.

callback_arg Argument to be passed to the completecallback.

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

completecallback is `GLOBUS_NULL`

handle already has an operation in progress

See also:

[globus_ftp_client_register_read\(\)](#)

5.5.4.19 `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_completecallback_t completecallback, 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()` 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 completecallback 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 le transfer.
 restart Pointer to a restart marker.
 completercallback Callback to be invoked once the "put" is completed.
 callbackarg Argument to be passed to the completercallback.

See also:

[globusftp_client_registerwrite\(\)](#)

5.5.4.20 `globus_result_t globusftp_client_partial_put (globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restart_marker_t restart, globus_off_t partial_offset, globus_off_t partial_end_offset, globusftp_client_completercallback_t completercallback, void callbackarg)`

Store a le on an FTP server.

This function starts a "put" le transfer to an FTP server. If this function returns `GLOBUS_SUCCESS`, then the user may immediately begin calling [globusftp_client_registerwrite\(\)](#) 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 completercallback will be invoked with the nal 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 le transfer.
 restart Pointer to a restart marker.
 partial_offset Starting offset for a partial le put.
 partial_end_offset Ending offset for a partial le put.
 completercallback Callback to be invoked once the "put" is completed.
 callbackarg Argument to be passed to the completercallback.

See also:

[globusftp_client_registerwrite\(\)](#)

5.5.4.21 `globus_result_t globusftp_client_extendedput (globusftp_client_handle_t handle, const char url, globusftp_client_operationattr_t attr, globusftp_client_restart_marker_t restart, const char algo_str, globusftp_client_completercallback_t completercallback, void callbackarg)`

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

This function starts a "put" le transfer to an FTP server. If this function returns `GLOBUS_SUCCESS`, then the user may immediately begin calling [globusftp_client_registerwrite\(\)](#) 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 completercallback will be invoked with the nal status of the put.

This function differs from the [globusftp_client_put\(\)](#) 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 put operation.

url The URL to store the data to. The URL may be an ftp or gsiftp URL.
 attr Attributes for this le transfer.
 restart Pointer to a restart marker.
 estoalg_str
 completecallback Callback to be invoked once the "put" is completed.
 callback.arg Argument to be passed to the completecallback.

See also:

[globus.ftp_client.registerwrite\(\)](#)

5.5.4.22 `globus_result_t globus_ftp_client_third_party_transfer (globus_ftp_client_handle_t handle, const char sourceurl, globus_ftp_client_operationattr_t sourceattr, const char desturl, globus_ftp_client_operationattr_t destattr, globus_ftp_client_restart_marker_t restart, globus_ftp_client_completecallback_t completecallback, void callback.arg)`

Transfer a le between two FTP servers.

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

When the transfer is completed or if the transfer is aborted, the completecallback will be invoked with the nal status of the transfer.

Parameters:

handle An FTP Client handle to use for the get operation.
 sourceurl The URL to transfer. The URL may be an ftp or gsiftp URL.
 sourceattr Attributes for the souce URL.
 desturl The destination URL for the transfer. The URL may be an ftp or gsiftp URL.
 destattr Attributes for the destination URL.
 restart Pointer to a restart marker.
 completecallback Callback to be invoked once the "put" is completed.
 callback.arg Argument to be passed to the completecallback.

Note:

The sourceattr and destattr MUST be compatible. For example, the MODE and TYPE should match for both the source and destination.

5.5.4.23 `globus_result_t globus_ftp_client_partial_third_party_transfer (globus_ftp_client_handle_t handle, const char sourceurl, globus_ftp_client_operationattr_t sourceattr, const char desturl, globus_ftp_client_operationattr_t destattr, globus_ftp_client_restart_marker_t restart, globus_off_t partial_offset, globus_off_t partial_end.offset, globus_ftp_client_completecallback_t completecallback, void callback.arg)`

Transfer a le between two FTP servers.

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

When the transfer is completed or if the transfer is aborted, the completecallback will be invoked with the nal status of the transfer.

Parameters:

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

`sourceurl` The URL to transfer. The URL may be an ftp or gsiftp URL.
`sourceattr` Attributes for the source URL.
`desturl` The destination URL for the transfer. The URL may be an ftp or gsiftp URL.
`destattr` Attributes for the destination URL.
`restart` Pointer to a restart marker.
`partial.offset` Starting offset for a partial le get.
`partial.end.offset` Ending offset for a partial le get. Use -1 for EOF.
`completecallback` Callback to be invoked once the "put" is completed.
`callback.arg` Argument to be passed to the `completecallback`.

Note:

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

5.5.4.24 `globus_result_t globus_ftp_client_extendedthird_party_transfer (globus_ftp_client_handle_t handle, const char sourceurl, globus_ftp_client_operationattr_t sourceattr, const char eretalg_str, const char desturl, globus_ftp_client_operationattr_t destattr, const char estoalg_str, globus_ftp_client_restart_marker_t restart, globus_ftp_client_completecallback_t completecallback, void callback_arg)`

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

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

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

This function differs from the `globus_ftp_client_third_party_transfer()` 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.
`sourceurl` The URL to transfer. The URL may be an ftp or gsiftp URL.
`sourceattr` Attributes for the source URL.
`eretalg_str`
`desturl` The destination URL for the transfer. The URL may be an ftp or gsiftp URL.
`destattr` Attributes for the destination URL.
`estoalg_str`
`restart` Pointer to a restart marker.
`completecallback` Callback to be invoked once the "put" is completed.
`callback.arg` Argument to be passed to the `completecallback`.

Note:

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

5.5.4.25 `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 completecallback, 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 `completecallback` 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.
- `completecallback` Callback to be invoked once the modification time check is completed.
- `callback_arg` Argument to be passed to the `completecallback`.

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`
- completecallback is `GLOBUS_NULL`
- handle already has an operation in progress

5.5.4.26 `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 completecallback, 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 `completecallback` 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.

`completercallback` Callback to be invoked once the size check is completed.
`callback.arg` Argument to be passed to the `completercallback`.

Returns:

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

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

5.5.4.27 `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_completercallback_t completercallback, 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 `completercallback` will be invoked with the final status of the operation. If the callback returns without an error, the checksum will be stored in the `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.
- `completercallback` Callback to be invoked once the checksum is completed.
- `callback.arg` Argument to be passed to the `completercallback`.

Returns:

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

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

5.5.4.28 `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

```
globusresultt globusftp_client_operationattrsetstoragemodule(globusftp_client_operationattr attr, const
char modulename, const charmoduleargs)
globusresultt globusftp_client_operationattrgetstoragemodule(constglobusftp_client_operationattr attr,
char modulename, char moduleargs)
```

Custom Data Channel Driver Stack

```
globusresultt globusftp_client_operationattrsetnet.stack(globusftp_client_operationattr attr, const char
driver.list)
globusresultt globusftp_client_operationattrgetnet.stack(constglobusftp_client_operationattr attr, char
driver.list)
```

Custom Server File Driver Stack

```
globusresultt globusftp_client_operationattrsetdisk.stack(globusftp_client_operationattr attr, const char
driver.list)
globusresultt globusftp_client_operationattrgetdisk.stack(constglobusftp_client_operationattr attr, char
driver.list)
```

Parallelism

```
globusresultt globusftp_client_operationattrsetparallelism (globusftp_client_operationattr attr, const
globusftp_control_parallelism_t parallelism)
globusresultt globusftp_client_operationattrgetparallelism (const globusftp_client_operationattr attr,
globusftp_control_parallelism_t parallelism)
```

allocate

```
globusresultt globusftp_client_operationattrsetallocate (globusftp_client_operationattr attr, const
globusoff_t allocatedsize)
globusresultt globusftp_client_operationattrgetallocate (const globusftp_client_operationattr attr,
globusoff_t allocatedsize)
```

authz.assert

```
globusresultt globusftp_client_operationattrsetauthzassert(globusftp_client_operationattr attr, const
char authzassert, globusboolt cacheauthzassert)
globusresultt globusftp_client_operationattrgetauthzassert(const globusftp_client_operationattr attr,
char authzassert, globusboolt cacheauthzassert)
```

Striped Data Movement

```

globus_result_t globusftp_client_operationattrsetstriped(globusftp_client_operationattr_t attr, globus_bool_t striped)
globus_result_t globusftp_client_operationattrgetstriped(constglobusftp_client_operationattr_t attr, globus_bool_t striped)

```

Striped File Layout

```

globus_result_t globusftp_client_operationattrsetLayout(globusftp_client_operationattr_t attr, constglobusftp_control_layout_t layout)
globus_result_t globusftp_client_operationattrgetlayout(constglobusftp_client_operationattr_t attr, globusftp_control_layout_t layout)

```

TCP Buffer

```

globus_result_t globusftp_client_operationattrsettcpbuffer (globusftp_client_operationattr_t attr, const globusftp_control_tcpbuffer_t tcpbuffer)
globus_result_t globusftp_client_operationattrgettcpbuffer (const globusftp_client_operationattr_t attr, globusftp_control_tcpbuffer_t tcpbuffer)

```

File Type

```

globus_result_t globusftp_client_operationattrsettype (globusftp_client_operationattr_t attr, globusftp_control_type_t type)
globus_result_t globusftp_client_operationattrgettype (constglobusftp_client_operationattr_t attr, globusftp_control_type_t type)

```

Transfer Mode

```

globus_result_t globusftp_client_operationattrsetmode (globusftp_client_operationattr_t attr, globusftp_control_mode_t mode)
globus_result_t globusftp_client_operationattrgetmode(constglobusftp_client_operationattr_t attr, globusftp_control_mode_t mode)

```

[NOHEADER]

```

globus_result_t globusftp_client_operationattrsetlist_usesdatamode (const globusftp_client_operationattr_t attr, globus_bool_t list_usesdatamode)
globus_result_t globusftp_client_operationattrgetlist_usesdatamode (const globusftp_client_operationattr_t attr, globus_bool_t list_usesdatamode)

```

Authorization

```

globus_result_t globusftp_client_operationattrsetauthorization(globusftp_client_operationattr_t attr, gss_cred_id_t credential, const char* user, const char* password, const char* account, const char* subject)
globus_result_t globusftp_client_operationattrgetauthorization(const globusftp_client_operationattr_t attr, gss_cred_id_t credential, char* user, char* password, char* account, char* subject)

```

Data Channel Authentication

```

globus_result_t globusftp_client_operationattrsetdcau(globusftp_client_operationattr attr, constglobus-
ftp_control_dcau_t dcau)
globus_result_t globusftp_client_operationattrgetdcau(constglobusftp_client_operationattr attr, globus-
ftp_control_dcau_t dcau)

```

Data Channel Protection

```

globus_result_t globusftp_client_operationattrsetdataprotection (globusftp_client_operationattr attr,
globusftp_control_protection_t protection)
globus_result_t globusftp_client_operationattrgetdataprotection(constglobusftp_client_operationattr attr,
globusftp_control_protection_t protection)

```

Control Channel Protection

```

globus_result_t globusftp_client_operationattrsetcontrolprotection (globusftp_client_operationattr attr,
globusftp_control_protection_t protection)
globus_result_t globusftp_client_operationattrgetcontrolprotection (const globusftp_client_operationattr-
t attr, globusftp_control_protection_t protection)

```

Append

```

globus_result_t globusftp_client_operationattrsetappend(globusftp_client_operationattr attr, globusbool_t
append)
globus_result_t globusftp_client_operationattrgetappend (const globusftp_client_operationattr attr,
globusbool_t append)

```

IPv6

```

globus_result_t globusftp_client_operationattrsetallow_ipv6 (globusftp_client_operationattr attr, globus-
bool_t allow_ipv6)
globus_result_t globusftp_client_operationattrgetallow_ipv6 (const globusftp_client_operationattr attr,
globusbool_t allow_ipv6)

```

Read into a Single Buffer

```

globus_result_t globusftp_client_operationattrsetreadall (globusftp_client_operationattr attr, globusbool_t
readall, globusftp_client_datacallback_t intermediatecallback, void intermediatecallbackarg)
globus_result_t globusftp_client_operationattrgetreadall (const globusftp_client_operationattr attr,
globusbool_t readall, globusftp_client_datacallback_t intermediatecallback, void intermediate-
callbackarg)

```

Typedefs

```

typedef globusftp_client_operationattr_t globusftp_client_operationattr_t

```

Functions

```

globusresult_t globusftp_client_operationattrinit (globusftp_client_operationattr_t attr)
globusresult_t globusftp_client_operationattrdestroy(globusftp_client_operationattr_t attr)
globusresult_t globusftp_client_operationattrsetdelayedpasv (const globusftp_client_operationattr_t attr,
globus_bool_t delayedpasv)
globusresult_t globusftp_client_operationattrcopy (globusftp_client_operationattr_t dst, const globusftp_
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 globusftp_client_operationattr_t globusftp_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 [attributes](#) section of the library.

See also:

[globusftp_client_operationattrinit\(\)](#), [globusftp_client_operationattrdestroy\(\)](#)

5.6.3 Function Documentation

5.6.3.1 globusresult_t globusftp_client_operationattr_init (globusftp_client_operationattr_t attr)

Initialize an FTP client attribute set.

Parameters:

attr A pointer to the new attribute set.

5.6.3.2 globusresult_t globusftp_client_operationattr_destroy (globusftp_client_operationattr_t attr)

Destroy an FTP client attribute set.

Parameters:

attr A pointer to the attribute to destroy.

5.6.3.3 globusresult_t globusftp_client_operationattr_setstorage.module (globusftp_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

See also:

[globusgsiftp_control_parallelism_t](#), [globusftp_client_operationattr_setlayout\(\)](#), [globusftp_client_operationattr_setmode\(\)](#)

Note:

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

5.6.3.4 `globusresult_t globusftp_client_operationattr_setnet_stack (globusftp_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 `globusresult_t globusftp_client_operationattr_setdisk_stack (globusftp_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 `globusresult_t globusftp_client_operationattr_setparallelism (globusftp_client_operationattr_t attr, const globusftp_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 "xed" 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 "xed" 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:

`globusgsftp_controlparallelism_t`, [globusftp_client_operationattrsetLayout\(\)](#), [globusftp_client_operationattrsetmode\(\)](#)

Note:

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

5.6.3.7 `globusresult_t globusftp_client_operationattr_setallocate (globusftp_client_operationattr_t attr, const globusoff_t allocatedsize)`

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.
- allocatedsize The size to direct server to allocate.

5.6.3.8 `globusresult_t globusftp_client_operationattr_setauthz_assert (globusftp_client_operationattr_t attr, const char authz_assert, globus_bool_t cacheauthz_assert)`

Set/Get the authzassert 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.
- cacheauthz_assert Boolean that specifies whether to cache this assertion for subsequent operations

5.6.3.9 `globusresult_t globusftp_client_operationattr_setstriped (globusftp_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 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:

[globusftp_client_operationattrsetparallelism\(\)](#) [globusftp_client_operationattrsetlayout\(\)](#) [globusftp_client_operationattrsetmode\(\)](#)

Note:

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

5.6.3.10 `globusresult_t globusftp_client_operationattr_setlayout (globusftp_client_operationattr_t attr, const globusftp_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:

[globusftp_control_layout_t](#), [globusftp_client_operationattrsetparallelism\(\)](#) [globusftp_client_operationattrsetmode\(\)](#)

Note:

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

5.6.3.11 `globusresult_t globusftp_client_operationattr_settcp_buffer (globusftp_client_operationattr_t attr, const globusftp_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 `tcpbuffer` attribute.

See also:

`globusgsiftp_control_tcpbuffer_t`

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.

See also:

`globus_ftp_control_type_t`

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_control_mode_t`, `globus_ftp_client_operationattr_set_parallelism()` `globus_ftp_client_operationattr_set_layout()`

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_setlist_usesdata_mode (const globus_ftp_client_operationattr_t attr, globus_bool_t list_usesdata_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_usesdata_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_setdelayedpasv (const globus_ftp_client_operationattr_t attr, globus_bool_t delayedpasv)`

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 lename to be transferred.

Parameters:

attr The attribute set to query or modify.
delayedpasv `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_setauthorization (globus_ftp_client_operationattr_t attr, gsscred_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_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_dcau (globus_ftp_client_operationattr_t attr, const globusftp_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.

5.6.3.18 `globus_result_t globus_ftp_client_operationattr_set_data_protection (globus_ftp_client_operationattr_t attr, globusftp_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_control_protection (globus_ftp_client_operationattr_t attr, globusftp_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.

5.6.3.20 `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()` or `globusftp_client.transfer()`.

This attribute is ignored on the retrieving side of a transfer, `globus_ftp_client_get()`.

Parameters:

`attr` The attribute set to query or modify.

`append` The value of append attribute.

5.6.3.21 `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)

5.6.3.22 `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 intermediatecallback, void intermediatecallback_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()` 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()` or `globus_ftp_client_third_party_transfer()` operations.

Parameters:

attr The attribute set to query or modify.

read_all The value of read_all attribute.

intermediatecallback 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.

intermediatecallback_arg User data to be passed to the intermediate callback function.

5.6.3.23 `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.

5.6.3.24 `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

See also:

[globusgsiftp.control.parallelism](#), [globusftp_client_operationattr_setlayout\(\)](#), [globusftp_client_operationattr_setmode\(\)](#)

Note:

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

5.6.3.25 `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.

5.6.3.26 `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.

5.6.3.27 `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 "xed" 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 "xed" 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:

`globusgsiftp_control_parallelism_t`, [globus_ftp_client_operationattr_set_layout\(\)](#), [globus_ftp_client_operationattr_set_mode\(\)](#)

Note:

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

5.6.3.28 `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.

5.6.3.29 `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

5.6.3.30 `globus_result_t globus_ftp_client_operationattr_get_stripped (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 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:

[globusftp_client_operationattrsetparallelism\(\)](#) [globusftp_client_operationattrsetLayout\(\)](#) [globusftp_client_operationattrsetmode\(\)](#)

Note:

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

5.6.3.31 `globusresult_t globusftp_client_operationattr_getlayout (const globusftp_client_operationattr_t attr, globusftp_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 transfered 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:

[globusftp_control_layout_t](#), [globusftp_client_operationattrsetparallelism\(\)](#) [globusftp_client_operationattrsetmode\(\)](#)

Note:

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

5.6.3.32 `globusresult_t globusftp_client_operationattr_gettcpbuffer (const globusftp_client_operationattr_t attr, globusftp_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 `tcpbuffer` attribute.

See also:

`globusgsiftp_control_tcpbuffer_t`

5.6.3.33 `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.

See also:

`globus_ftp_control_type_t`

5.6.3.34 `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_control_mode_t, [globus.ftp_client.operationattr.setparallelism\(\)](#) [globus.ftp_client.operationattr.setlayout\(\)](#)

Note:

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

5.6.3.35 `globus_result_t globus_ftp_client_operationattr_getlist_usesdata_mode (const globus_ftp_client_operationattr_t attr, globus_bool_t list_usesdata_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_usesdata_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.

5.6.3.36 `globus_result_t globus_ftp_client_operationattr_getauthorization (const globus_ftp_client_operationattr_t attr, gsscred_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_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).

5.6.3.37 `globusresult_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.

5.6.3.38 `globusresult_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.

5.6.3.39 `globusresult_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.

5.6.3.40 `globusresult_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\(\)](#) or [globusftp_client.transfer\(\)](#).

This attribute is ignored on the retrieving side of a transfer, [globus_ftp_client_get\(\)](#).

Parameters:

attr The attribute set to query or modify.

append The value of append attribute.

5.6.3.41 `globusresult_t globusftp_client_operationattr_get_allow_ipv6 (const globusftp_client_operationattr_t attr, globusbool_t allow_ipv6)` (const [globusftp_client_operationattr_t](#) attr, `globusbool_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)

5.6.3.42 `globusresult_t globusftp_client_operationattr_get_read_all (const globusftp_client_operationattr_t attr, globusbool_t read_all, globusftp_client_data_callback_t intermediatecallback, void intermediatecallbackarg)` (const [globusftp_client_operationattr_t](#) attr, `globusbool_t` read_all, [globusftp_client_data_callback_t](#) intermediatecallback, void intermediatecallbackarg)

Set/Get the readall 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 [globusftp_client_registerread\(\)](#) 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 [globusftp_client_put\(\)](#) or [globusftp_client_third_party_transfer\(\)](#) operations.

Parameters:

attr The attribute set to query or modify.

read_all The value of readall attribute.

intermediatecallback 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.

intermediatecallbackarg 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( globusftp\_client\_datacallback\_t )(void userarg, globusftp\_client\_handle\_t handle, globusobject_t error, globusbyte_t buffer, globussize_t length, globusoff_t offset, globusbool_t eof)
```

Functions

```
globusresult_t globusftp\_client\_registerread (globusftp\_client\_handle\_t handle, globusbyte_t buffer, globussize_t buffer_length, globusftp\_client\_datacallback\_t callback, void callbackarg)
globusresult_t globusftp\_client\_registerwrite (globusftp\_client\_handle\_t handle, globusbyte_t buffer, globussize_t buffer_length, globusoff_t offset, globusbool_t eof, globusftp\_client\_datacallback\_t callback, void callbackarg)
```

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 `globusftp_client_get()`, `globusftp_client_partial_get()`, `globusftp_client_put()`, `globusftp_client_partial_put()`, `globusftp_client_list()`, `globusftp_client_machinelist()`, and `globusftp_client_verboselist()`.

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(globusftp_client_data_callback_t)(void user_arg, globusftp_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 `userarg` 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 `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 `le` 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 `globusresult_t globusftp_client_register_read (globusftp_client_handle_t handle, globus_byte_t buffer, globus_size_t buffer_length, globusftp_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:

`globusftp_client_operationattissetreadall()`

5.7.3.2 `globusresult_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

Defines

```
#define GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_MODULE (&globus_ftp_client_debugplugin_module)
```

Functions

```
globusresult_t globus_ftp_client_debugplugin_init (globus_ftp_client_plugin_t plugin, FILE *stream, const char *text)
globusresult_t globus_ftp_client_debugplugin_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 De ne Documentation

5.8.2.1 #de ne GLOBUSFTP_CLIENT_DEBUG_PLUGIN_MODULE (&globus_i_ftp_client_debug.plugin_-module)

Module descriptor.

5.8.3 Function Documentation

5.8.3.1 globusresult_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:

[globusftp_client_debugplugin_destroy\(\)](#), [globusftp_client_handleattraddplugin\(\)](#), [globusftp_client_handleattrremoveplugin\(\)](#), [globusftp_client_handleinit\(\)](#)

5.8.3.2 globusresult_t globusftp_client_debugplugin_destroy ([globusftp_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 [globusftp_client_debugplugin_init\(\)](#)

Returns:

This function returns an error if

- plugin is null
- plugin is not a debugging plugin

See also:

[globusftp_client_debugplugin_init\(\)](#), [globusftp_client_handleattraddplugin\(\)](#), [globusftp_client_handleattrremoveplugin\(\)](#), [globusftp_client_handleinit\(\)](#)

5.9 Performance Marker Plugin

Defines

```
#define GLOBUSFTP_CLIENT_PERFPLUGIN_MODULE (&globusftp_client_perf_plugin_module)
```

Typedefs

```
typedef void( globusftp_client_perf_plugin_begin_cb_t )(void userspec_t, globusftp_client_handle_t handle, const char sourceurl, const char desturl, globusbool_t restart)
typedef void( globusftp_client_perf_plugin_marker_cb_t )(void userspec_t, globusftp_client_handle_t handle, long timestamp, char timestamplength, int stripindex, int numstripes, globusoff_t nbytes)
typedef void( globusftp_client_perf_plugin_complete_cb_t )(void userspec_t, globusftp_client_handle_t handle, globusbool_t success)
typedef void ( globusftp_client_perf_plugin_usercopy_cb_t )(void userspec_t)
typedef void( globusftp_client_perf_plugin_userdestroy_cb_t )(void userspec_t)
```

Functions

```
globusresult_t globusftp_client_perf_plugin_init (globusftp_client_plugin_t plugin, globusftp_client_perf_plugin_begin_cb_t begin_cb, globusftp_client_perf_plugin_marker_cb_t marker_cb, globusftp_client_perf_plugin_complete_cb_t complete_cb, void userspec_t)
globusresult_t globusftp_client_perf_plugin_setcopy_destroy(globusftp_client_plugin_t plugin, globusftp_client_perf_plugin_usercopy_cb_t copy_cb, globusftp_client_perf_plugin_userdestroy_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_spec(c) (globus_ftp_client_plugin_t plugin, void
user_spec(c))

```

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 GLOBUSFTP_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_spec(c), 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_spec(c) this is user_spec(c) 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_spec(c), globus_ftp_client_handle_t handle, long time_stamp, int time_stamp_length, 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 won't 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_speci c this is user speci c data either created by the copy method, or, if a copy method was not speci ed, 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_speci c, 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_speci c this is user speci c data either created by the copy method, or, if a copy method was not speci ed, 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_speci c)`

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 speci c data

Parameters:

user_speci c this is user speci c data either created by this copy method, or the value passed to init

Returns:

a pointer to a user speci c 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_speci c)`

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 speci c data

Parameters:

user_speci c this is user speci c data created by the copy method

Returns:

n/a

5.9.4 Function Documentation

5.9.4.1 `globusresult_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 completecb, void user_spec)`

Initialize a perf plugin.

This function initializes a performance marker plugin. Any params except for the plugin may be `GLOBUS`

Parameters:

- `plugin` a pointer to a plugin type to be initialized
- `user_spec` a pointer to some user spec 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
- `completecb` 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 `globusresult_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
- `destroycb` func to be called when a copy is to be destroyed

Returns:

- Error on NULL arguments
- `GLOBUS.SUCCESS`

5.9.4.3 `globusresult_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 `globusresult_t globusftp_client_perf_plugin_get_user_spec(c)(globusftp_client_plugin_t plugin, void user_spec(c))`

Retrieve user spec(c) pointer.

Parameters:

plugin plugin previously initialized with `init` (above)
 user_spec(c) pointer to storage for user_spec(c) pointer

Returns:

`GLOBUS_SUCCESS`
 Error on NULL plugin
 Error on NULL user_spec(c)

5.10 Plugins

Plugin API.

Modules

[Debugging Plugin](#)
[Performance Marker Plugin](#)
[Restart Marker Plugin](#)
[Restart Plugin](#)
[Netlogger Throughput Plugin](#)
[Throughput Performance Plugin](#)

Typedefs

```
typedef globusftp\_client\_plugin\_t globusftp\_client\_plugin\_t
typedef globusftp\_client\_plugin\_t ( globusftp\_client\_plugin\_copy\_t )(globusftp_client_plugin_t plugin,
template, void plugin_spec(c))
typedef void( globusftp\_client\_plugin\_destroy\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c))
typedef void( globusftp\_client\_plugin\_connect\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const charurl)
typedef void( globusftp\_client\_plugin\_authenticate\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const charurl, constglobusftp_control_auth_info_t authinfo)
typedef void( globusftp\_client\_plugin\_chmod\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const charurl, int mode, constglobusftp\_client\_operationattr\_t attr,
globusbool_t restart)
typedef void( globusftp\_client\_plugin\_cksm\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const char url, globusoff_t offset, globusoff_t length, const char
algorithm, constglobusftp\_client\_operationattr\_t attr, globusbool_t restart)
typedef void( globusftp\_client\_plugin\_delete\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const charurl, constglobusftp\_client\_operationattr\_t attr, globusbool_t
restart)
typedef void( globusftp\_client\_plugin\_feat\_t )(globusftp_client_plugin_t plugin, void plugin_spec(c),
globusftp\_client\_handle\_t handle, const charurl, constglobusftp\_client\_operationattr\_t attr, globusbool_t
restart)
```

```

typedef void( globusftp_client_plugin_mkdir_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_rmdir_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_list_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_verboselist_t )(globusftp_client_plugin_t plugin, void plugin_
speci c, globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr,
globusbool_t restart)
typedef void( globusftp_client_plugin_machinelist_t )(globusftp_client_plugin_t plugin, void plugin_
speci c, globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr,
globusbool_t restart)
typedef void( globusftp_client_plugin_mlst_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_pluginstatt_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_move_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char sourceurl, const char desturl, const globusftp_client_
operationattr_t attr, globusbool_t restart)
typedef void( globusftp_client_plugin_get_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_put_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_third_party_transfer_t )(globusftp_client_plugin_t plugin, void
plugin_speci c, globusftp_client_handle_t handle, const char sourceurl, const globusftp_client_
operationattr_t sourceattr, const char desturl, const globusftp_client_operationattr_t destattr, globusbool_t
restart)
typedef void( globusftp_client_plugin_modification_time_t )(globusftp_client_plugin_t plugin, void plugin_
speci c, globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr,
globusbool_t restart)
typedef void( globusftp_client_plugin_size_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, globusbool_t
restart)
typedef void( globusftp_client_plugin_abort_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle)
typedef void( globusftp_client_plugin_read_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const globusbyte_t buffer, globussize_t buffer_length)
typedef void( globusftp_client_plugin_write_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const globusbyte_t buffer, globussize_t buffer_length, globusoff_t offset,
globusbool_t eof)
typedef void( globusftp_client_plugin_data_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, globusobject_t error, const globusbyte_t buffer, globussize_t length,
globusoff_t offset, globusbool_t eof)
typedef void( globusftp_client_plugin_command_t )(globusftp_client_plugin_t plugin, void plugin_speci c,
globusftp_client_handle_t handle, const char url, const char command)

```

```

typedef void( globusftp_client_plugin_response_t )(globusftp_client_plugin_t plugin, void plugin_spec_t c,
globusftp_client_handle_t handle, const char url, globus_object_t error, const globusftp_control_response_t
ftp_response)
typedef void( globusftp_client_plugin_fault_t )(globusftp_client_plugin_t plugin, void plugin_spec_t c,
globusftp_client_handle_t handle, const char url, globus_object_t error)
typedef void( globusftp_client_plugin_complete_t )(globusftp_client_plugin_t plugin, void plugin_spec_t c,
globusftp_client_handle_t handle)

```

Enumerations

```

enum globusftp_client_plugin_commandmask_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 globusftp_client_plugin_restartlist (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartverbose_list (globusftp_client_handle_t handle, const char url,
const globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartmachinelist (globusftp_client_handle_t handle, const char
url, const globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartmlst (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartstat (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartchmod (globusftp_client_handle_t handle, const char url, int
mode, const globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartcksm (globusftp_client_handle_t handle, const char url,
globus_off_t offset, globus_off_t length, const char algorithm, const globusftp_client_operation_attr_t attr, const
globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartdelete (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartfeat (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartmkdir (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartrmdir (globusftp_client_handle_t handle, const char url, const
globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartmove (globusftp_client_handle_t handle, const char source-
url, const char desturl, const globusftp_client_operation_attr_t attr, const globus_abstime_t when)
globus_result_t globusftp_client_plugin_restartget (globusftp_client_handle_t handle, const char url,
const globusftp_client_operation_attr_t attr, globusftp_client_restartmarkert restartmarker, const globus
abstime_t when)
globus_result_t globusftp_client_plugin_restartput (globusftp_client_handle_t handle, const char url,
const globusftp_client_operation_attr_t attr, globusftp_client_restartmarkert restartmarker, const globus
abstime_t when)

```



```

globus_result_t globusftp_client_plugin_restartthird_party_transfer(globusftp_client_handle_t handle, const
char sourceurl, const globusftp_client_operationattr_t sourceattr, const char desturl, const globus-
ftp_client_operationattr_t destattr, globusftp_client_restartmarker_t restartmarker, const globusabstime_t
when)
globus_result_t globusftp_client_plugin_restartsize(globusftp_client_handle_t handle, const charurl, const
globusftp_client_operationattr_t attr, const globusabstime_t when)
globus_result_t globusftp_client_plugin_restartmodification_time(globusftp_client_handle_t handle, const
char url, const globusftp_client_operationattr_t attr, const globusabstime_t when)
globus_result_t globusftp_client_plugin_restartgetmarker(globusftp_client_handle_t handle, globusftp_-
client_restartmarker_t marker)
globus_result_t globusftp_client_plugin_abort(globusftp_client_handle_t handle)
globus_result_t globusftp_client_plugin_adddatachannels(globusftp_client_handle_t handle, unsigned int
num_channels, unsigned int stripe)
globus_result_t globusftp_client_plugin_removedatachannels(globusftp_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 `globusftp_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 `globusftp_client_plugin_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` and `destroy` 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 globusftp_client_plugin_t globusftp_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 `globusftp_client_handle_t`.

See also:

[globusftp_client_handleinit\(\)](#), [globusftp_client_handledestroy\(\)](#) [globusftp_client_handleattr](#), [Debugging Plugin](#)

5.10.2.2 typedef globusftp_client_plugin_t (globusftp_client_plugin_copy_t)(globusftp_client_plugin_t plugin_template, void plugin_specic)

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-speci c initialization function. by the user.
plugin_speci c Plugin-speci c data.

Returns:

A pointer to a plugin. This plugin copy must remain valid until the [copy_destroy](#) function is called on the copy.

See also:

[globusftp_client_plugin_destroy](#)

5.10.2.3 typedef void(globusftp_client_plugin_destroy_t)([globusftp_client_plugin_t](#) plugin, void plugin_speci c)

Plugin destroy function.

This function is used to free or unreference a copy of a plugin which was allocated by calling the instantiate function from the plugin.

Parameters:

plugin The plugin, created by the create function, which is to be destroyed.
plugin_speci c Plugin-speci c data.

5.10.2.4 typedef void(globusftp_client_plugin_connect_t)([globusftp_client_plugin_t](#) plugin, void plugin_speci c, [globusftp_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_speci c Plugin-speci c 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(globusftp_client_plugin_authenticate_t)([globusftp_client_plugin_t](#) plugin, void plugin_speci c, [globusftp_client_handle_t](#) handle, const char url, const globusftp_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_spec The Plugin-spec 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_spec, 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_spec The Plugin-spec 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_cksm_t)( globus_ftp_client_plugin_t plugin, void plugin_spec, 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 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_spec The Plugin-spec data.

handle The handle associated with the delete operation.

url The url to chmod.

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`.

```
5.10.2.8 typedef void( globusftp_client_plugin_delete_t)( globusftp_client_plugin_t plugin, void plugin_
speci c, globusftp_client_handle_t handle, const char url, const globusftp_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_speci c 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`.

```
5.10.2.9 typedef void( globusftp_client_plugin_feat_t)( globusftp_client_plugin_t plugin, void plugin_
speci c, globusftp_client_handle_t handle, const char url, const globusftp_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_speci c 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`.

5.10.2.10 `typedef void(globus.ftp_client_plugin_mkdir_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec Plugin-spec 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`.

5.10.2.11 `typedef void(globus.ftp_client_plugin_rmdir_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec Plugin-spec 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`.

5.10.2.12 `typedef void(globus.ftp_client_plugin_list_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec c Plugin-spec c 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`.

```
5.10.2.13 typedef void( globus_ftp_client_plugin_verbose_list_t)( globus_ftp_client_plugin_t plugin, void
plugin_spec c, 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_spec c Plugin-spec c 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`.

```
5.10.2.14 typedef void( globus_ftp_client_plugin_machine_list_t)( globus_ftp_client_plugin_t plugin, void
plugin_spec c, 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_spec c Plugin-spec c 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`.

5.10.2.15 `typedef void(globus.ftp_client_plugin_mlst_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec` 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_stat_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec` 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_move_t)(globus.ftp_client_plugin_t plugin, void plugin_spec, 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_spec Plugin-spec data.

handle The handle associated with the move operation.

sourceurl The source url of the move operation.

desturl The destination url of the move operation.

attr The attributes to be used during this move.

restart This value is set to `GLOBAL_TRUE` when this callback is caused by a plugin restarting the current move transfer; otherwise, this is set to `GLOBAL_FALSE`.

```
5.10.2.18 typedef void( globusftp_client_plugin_get_t)( globusftp_client_plugin_t plugin, void plugin_spec, globusftp_client_handle_t handle, const char url, const globusftp_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_spec Plugin-spec 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 `GLOBAL_TRUE` when this callback is caused by a plugin restarting the current get transfer; otherwise, this is set to `GLOBAL_FALSE`.

```
5.10.2.19 typedef void( globusftp_client_plugin_put_t)( globusftp_client_plugin_t plugin, void plugin_spec, globusftp_client_handle_t handle, const char url, const globusftp_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_spec Plugin-spec 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 `GLOBAL_TRUE` when this callback is caused by a plugin restarting the current put transfer; otherwise, this is set to `GLOBAL_FALSE`.


```
5.10.2.20 typedef void( globus_ftp_client_plugin_third_party_transfer_t)( globus_ftp_client_plugin_t plugin,
void plugin_spec c, 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_spec Plugin-spec 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.

```
5.10.2.21 typedef void( globus_ftp_client_plugin_modification_time_t)( globus_ftp_client_plugin_t plugin,
void plugin_spec c, 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_spec Plugin-spec 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.

```
5.10.2.22 typedef void( globus_ftp_client_plugin_size_t)( globus_ftp_client_plugin_t plugin, void plugin_
spec c, 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_spec The 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`.

```
5.10.2.23 typedef void( globus_ftp_client_plugin_abort_t)( globus_ftp_client_plugin_t plugin, void plugin_spec, 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_spec The plugin-specific data.

handle The handle associated with the request.

```
5.10.2.24 typedef void( globus_ftp_client_plugin_read_t)( globus_ftp_client_plugin_t plugin, void plugin_spec, 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_spec The 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.

```
5.10.2.25 typedef void( globus_ftp_client_plugin_write_t)( globus_ftp_client_plugin_t plugin, void plugin_spec, 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_spec The 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.

5.10.2.26 `typedef void(globusftp_client_plugin_data_t)(globusftp_client_plugin_t plugin, void plugin_spec, globusftp_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_spec The 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.

5.10.2.27 `typedef void(globusftp_client_plugin_command_t)(globusftp_client_plugin_t plugin, void plugin_spec, globusftp_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_spec The 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).

5.10.2.28 `typedef void(globus_ftp_client_plugin_responset)(globus_ftp_client_plugin_t plugin, void plugin_speci_c, globus_ftp_client_handle_t handle, const char url, globus_object_t error, const globus_ftp_control_responset 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_speci_c` 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.

5.10.2.29 `typedef void(globus_ftp_client_plugin_fault_t)(globus_ftp_client_plugin_t plugin, void plugin_speci_c, 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_speci_c` 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.

5.10.2.30 `typedef void(globus_ftp_client_plugin_completest)(globus_ftp_client_plugin_t plugin, void plugin_speci_c, 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_speci_c` Plugin-specific data.
- `handle` The handle associated with the operation.

5.10.3 Enumeration Type Documentation

5.10.3.1 enum globusftp_client_plugin_command_mask_t

Command Mask.

This enumeration includes the types of commands which the plugin is interested in.

Enumeration values:

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, 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 globusresult_t globusftp_client_plugin_restart_list (globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, const globusabstime_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 globusresult_t globusftp_client_plugin_restart_verbose_list (globusftp_client_handle_t handle, const char url, const globusftp_client_operationattr_t attr, const globusabstime_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 `GLOBAL_RESTART`.

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_operation_attr_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 `GLOBAL_RESTART`.

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_mlst (globus_ftp_client_handle_t handle, const char *url, const globus_ftp_client_operation_attr_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 `GLOBAL_RESTART`.

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.

5.10.4.5 `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 `FALSE`.

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_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 `FALSE`.

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.

5.10.4.7 `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 cksm events will receive a cksm callback with the restart boolean set to `TRUE`.

Parameters:

- handle The handle which is associated with the cksm.
- url The destination URL of the transfer. This may be different than the original cksm'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 cksm's attribute set.
- when Absolute time for when to restart the cksm. 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_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 `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.

5.10.4.9 `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 `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.

5.10.4.10 `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 `GLOBAL_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.

5.10.4.11 `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 `GLOBAL_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.

5.10.4.12 `globus_result_t globus_ftp_client_plugin_restart_move (globus_ftp_client_handle_t handle, const char sourceurl, const char desturl, 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.

`sourceurl` 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.

`desturl` 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.

5.10.4.13 `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 restartmarker, 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.

`restartmarker` 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.

5.10.4.14 `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 restartmarker, 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 `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 its 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.

`restartmarker` 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.

5.10.4.15 `globus_result_t globus_ftp_client_plugin_restart_third_party_transfer (globus_ftp_client_handle_t handle, const char sourceurl, const globus_ftp_client_operationattr_t sourceattr, const char desturl, const globus_ftp_client_operationattr_t destattr, globus_ftp_client_restart_marker_t restartmarker, 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 `TRUE`.

Parameters:

`handle` The handle which is associated with the transfer.

`sourceurl` 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.

`sourceattr` 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.

`desturl` 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.

- `destattr` 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.

5.10.4.16 `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.

5.10.4.17 `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.

5.10.4.18 `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_SUCCESS`.

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)

5.10.4.19 `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.

5.10.4.20 `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?

5.10.4.21 `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

Defines

```
#define GLOBUSFTP_CLIENT_RESTART_MARKER_PLUGIN_MODULE (&globus_i_ftp_client_restart-
markerplugin_module)
```

Typedefs

```
typedef globus_bool_t (globusftp_client_restartmarkerplugin_begin_cb_t)(void userarg, globusftp_client-
handle_t handle, const char sourceurl, const char desturl, globusftp_client_restartmarkert_t usersaved-
marker)
typedef void (globusftp_client_restartmarkerplugin_marker_cb_t)(void userarg, globusftp_client_handle_t
handle, globusftp_client_restartmarkert_t marker)
typedef void (globusftp_client_restartmarkerplugin_completecb_t)(void userarg, globusftp_client-
handle_t handle, globus_object_t error, const char error_url)
```

Functions

```
globus_result_t globusftp_client_restartmarkerplugin_init (globusftp_client_plugin_t plugin, globusftp-
client_restartmarkerplugin_begin_cb_t begin_cb, globusftp_client_restartmarkerplugin_marker_cb_t marker-
cb, globusftp_client_restartmarkerplugin_completecb_t completecb, void userarg)
globus_result_t globusftp_client_restartmarkerplugin_destroy(globusftp_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 GLOBUSFTP_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_savedmarker)`

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_savedmarker' 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
- sourceurl source of the transfer (`GLOBUS_NULL` if 'put')
- desturl dest of the transfer (`GLOBUS_NULL` if 'get')
- user_savedmarker pointer to an uninitialized restart marker

Returns:

- `GLOBUS_TRUE` to indicate that the plugin should use 'user_savedmarker' to restart the transfer (and subsequently, destroy the marker)
- `GLOBUS_FALSE` to indicate that 'user_savedmarker' 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_restartmarker_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, globusobject_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_SUCCESS on success)

error_url the url which is the source of the above error (GLOBUS_SUCCESS on success)

Returns:

n/a

5.11.4 Function Documentation

5.11.4.1 globusresult_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 completecb, void user_arg)

Initialize a restart marker plugin.

This function initializes a restart marker plugin. Any params except for the plugin may be GLOBUS_SUCCESS

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

completecb 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 globusresult_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

De nes

```
#define GLOBUS_FTP_CLIENT_RESTART_PLUGIN_MODULE (&globus_i_ftp_client_restartplugin_
module)
```

Functions

```
globus_result_t globus_ftp_client_restartplugin_init (globus_ftp_client_plugin_t plugin, int maxretries, globus_
retime_t interval, globus_abstime_t deadline)
globus_result_t globus_ftp_client_restartplugin_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_restartplugin_init()`.

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_restartplugin_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 De ne Documentation

5.12.2.1 #de ne GLOBUSFTP_CLIENT_RESTART_PLUGIN_MODULE (&globus_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_retime_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_restartplugin_destroy\(\)](#) [globus_ftp_client_handleattraddplugin\(\)](#), [globus_ftp_client_handleattrremoveplugin\(\)](#), [globus_ftp_client_handleinit\(\)](#)

5.12.3.2 globusresult_t globusftp_client_restart_plugin_destroy (globusftp_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 `globusftp_client_restart_plugin_init()`

Returns:

This function returns an error if

- plugin is null
- plugin is not a restart plugin

See also:

`globusftp_client_restart_plugin_init()`, `globusftp_client_handleattraddplugin()`, `globusftp_client_handleattrremoveplugin()`, `globusftp_client_handleinit()`

5.13 Netlogger Throughput Plugin

Defines

```
#define GLOBUSFTP_CLIENT_THROUGHPUTNL_PLUGIN_MODULE (&globus_i_ftp_client_throughputnl_plugin_module)
```

Functions

```
globusresult_t globusftp_client_throughputnl_plugin_init (globusftp_client_plugin_t plugin, const char nl_url, const char prog_name, const char paquestring)
globusresult_t globusftp_client_throughputnl_plugin_init_with_handle (globusftp_client_plugin_t plugin, NLhandle nl_handle, const char paquestring)
globusresult_t globusftp_client_throughputnl_plugin_destroy (globusftp_client_plugin_t plugin)
globusresult_t globusftp_client_throughputnl_plugin_set_callbacks (globusftp_client_plugin_t plugin, globusftp_client_throughputplugin_begin_cb_t begin_cb, globusftp_client_throughputplugin_stripe_cb_t per_stripe_cb, globusftp_client_throughputplugin_total_cb_t total_cb, globusftp_client_throughputplugin_completecb_t completecb, void user_specic)
```

5.13.1 Detailed Description

This plugin allows a user to easily use the throughput plugin to log performance data via Netlogger.

The plugin will log the following Event Types with its corresponding 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 transferred thus far
```

BW.CURRENT< oat> Current (instantaneous) bandwidth
 BW.AVG < oat> 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< oat> Current (instantaneous) bandwidth on this stripe
 BW.AVG < oat> 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 De ne Documentation

5.13.2.1 #de ne GLOBUSFTP_CLIENT_THROUGHPUT_NL_PLUGIN_MODULE (&globus_i_ftp_client_-throughput_nl_plugin_module)

Module descriptor.

5.13.3 Function Documentation

5.13.3.1 globusresult_t globus.ftp_client.throughput_nl_plugin_init (globus.ftp_client.plugin_t plugin, const char nl_url, const char prog_name, const char opaquestring)

Initialize netlogger wrapped throughput plugin.

This will initialize a netlogger wrapped throughput plugin. Note that ~~the url~~ may be NULL. Regardless of what nl_host is set to, if the env variable NDEST.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

opaquestring 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 opaquestring)`

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

opaquestring 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 completecb, void user_specic)`

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

completecb the callback to be called to indicate transfer completion

user_specic 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](#)

5.14 Throughput Performance Plugin

De nes

```
#define GLOBUS_FTP_CLIENT_THROUGHPUT_PLUGIN_MODULE (&globus_i_ftp_client_throughput_plugin_module)
```

Typedefs

```
typedef void( globusftp_client_throughputplugin_begin_cb_t )(void userspeci c, globusftp_client_handle_t handle, const char sourceurl, const char desturl)
typedef void( globusftp_client_throughputplugin_stripe_cb_t )(void userspeci c, globusftp_client_handle_t handle, int stripeidx, globusoff_t bytes, oat instantaneousthroughput, oat avgthroughput)
typedef void( globusftp_client_throughputplugin_total_cb_t )(void userspeci c, globusftp_client_handle_t handle, globusoff_t bytes, oat instantaneousthroughput, oat avgthroughput)
typedef void( globusftp_client_throughputplugin_completecb_t )(void userspeci c, globusftp_client_handle_t handle, globusbool_t success)
typedef void ( globusftp_client_throughputplugin_usercopy_cb_t )(void userspeci c)
typedef void( globusftp_client_throughputplugin_userdestroycb_t )(void userspeci c)
```

Functions

```
globusresult_t globusftp_client_throughputplugin_init (globusftp_client_plugin_t plugin, globusftp_client_throughputplugin_begin_cb_t begin_cb, globusftp_client_throughputplugin_stripe_cb_t per_stripe_cb, globusftp_client_throughputplugin_total_cb_t total_cb, globusftp_client_throughputplugin_completecb_t completecb, void userspeci c)
globusresult_t globusftp_client_throughputplugin_setcopy_destroy (globusftp_client_plugin_t plugin, globusftp_client_throughputplugin_usercopy_cb_t copy_cb, globusftp_client_throughputplugin_userdestroycb_t destroycb)
globusresult_t globusftp_client_throughputplugin_destroy(globusftp_client_plugin_t plugin)
globusresult_t globusftp_client_throughputplugin_getuserspeci c (globusftp_client_plugin_t plugin, void userspeci c)
```

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 De ne Documentation

5.14.2.1 #define GLOBUSFTP_CLIENT_THROUGHPUT_PLUGIN_MODULE (&globus_i_ftp_client_throughput_plugin_module)

Module descriptor.

5.14.3 Typedef Documentation

5.14.3.1 `typedef void(globusftp_client_throughput_plugin_begin_cb_t)(void user_spec, globusftp_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_spec` User argument passed to `globusftp_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(globusftp_client_throughput_plugin_stripe_cb_t)(void user_spec, globusftp_client_handle_t handle, int stripe_ndx, globus_off_t bytes, oat instantaneous_throughput, oat 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 will have an instantaneous throughput based from the time the command was sent.

Parameters:

`handle` The client handle associated with this transfer
`user_spec` User argument passed to `globusftp_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(globusftp_client_throughput_plugin_total_cb_t)(void user_spec, globusftp_client_handle_t handle, globus_off_t bytes, oat instantaneous_throughput, oat 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 `perstripe_cb`

Parameters:

`handle` The client handle associated with this transfer
`user_spec` User argument passed to `globusftp_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(globusftp_client_throughput_plugin_complete_cb_t)(void user_speci c, globusftp_client_handle_t handle, globusbool_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_speci c User argument passed to `globusftp_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(globusftp_client_throughput_plugin_user_copy_cb_t)(void user_speci c)`

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 speci c data

Parameters:

user_speci c this is user speci c data either created by this copy method, or the value passed to init

Returns:

a pointer to a user speci c piece of data
GLOBUS.NULL (does not indicate error)

5.14.3.6 `typedef void(globusftp_client_throughput_plugin_user_destroy_cb_t)(void user_speci c)`

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 speci c data

Parameters:

user_speci c this is user speci c data created by the copy method

Returns:

n/a

5.14.4 Function Documentation

5.14.4.1 `globusresult_t globusftp_client_throughput_plugin_init (globusftp_client_plugin_t plugin, globusftp_client_throughput_plugin_begin_cb_t begin_cb, globusftp_client_throughput_plugin_stripe_cb_t per_stripe_cb, globusftp_client_throughput_plugin_total_cb_t total_cb, globusftp_client_throughput_plugin_complete_cb_t completecb, void user_speci c)`

Throughput plugin init.

Use this function to initialize a throughput plugin. The throughput plugin sits on top of the 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
 completecb the callback to be called to indicate transfer completion
 user_spec c a pointer to some user spec c 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 destroycb)`

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
 destroycb 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_spec c (globus_ftp_client_plugin_t plugin, void user_spec c)`

Retrieve user spec c pointer.

Parameters:

plugin plugin previously initialized with init (above)

user_speci c pointer to storage for user_speci c pointer

Returns:

GLOBUS_SUCCESS
Error on NULL plugin
Error on NULL user_speci c

6 globus ftp client Data Structure Documentation

6.1 globusftp_client_restart_extendedblock_t Struct Reference

Extended block mode restart marker.

6.1.1 Detailed Description

Extended block mode restart marker.

6.2 globusftp_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:

[globusftp_client_restartmarkerinit\(\)](#), [globusftp_client_restartmarkerdestroy\(\)](#) [globusftp_client_restartmarkercopy\(\)](#), [globusftp_client_restartmarkerinsertrange\(\)](#) [globusftp_client_restartmarkeroffset\(\)](#)

6.3 globusftp_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 [globusftp_client_operationattr_set_data_protection](#)(globusftp_client_operationattr_t attr, globusftp_control_protect
Only safe and private protection levels are supported by gsiftp.

Global [globus.ftp_client_operationattr_setcontrol_protection](#)(globus.ftp_client_operationattr_t attr, globus_ftp_control_prot
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 [55](#)

FTP Operation Attributes [36](#)

FTP Operations [17](#)

globusftp_client.abort
 globusftp_client.operations [35](#)
globusftp_client.activation
 GLOBUS_FTP_CLIENT_MODULE, [3](#)
globusftp_client.chmod
 globusftp_client.operations [28](#)
globusftp_client.cksm
 globusftp_client.operations [35](#)
GLOBUS_FTP_CLIENT_CMD_MASK_ALL
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK_BUFFER
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK_CONTROL-
 ESTABLISHMENT
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK_DATA -
 ESTABLISHMENT
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK_FILE -
 ACTIONS
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK -
 INFORMATION
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK_MISC
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK -
 TRANSFERMODIFIERS
 globusftp_client.plugins, [76](#)
GLOBUS_FTP_CLIENT_CMD_MASK -
 TRANSFERPARAMETERS
 globusftp_client.plugins, [76](#)
globusftp_client.completecallbackt
 globusftp_client.operations [20](#)
globusftp_client.data
 globusftp_client.datacallbackt, [54](#)
 globusftp_client.registerread, [54](#)
 globusftp_client.registerwrite, [54](#)
globusftp_client.datacallbackt
 globusftp_client.data, [54](#)
globusftp_client.debugplugin
 globusftp_client.debugplugin.destroy, [57](#)
 globusftp_client.debugplugin.init, [56](#)

 GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_-
 MODULE, [56](#)
globusftp_client.debugplugin.destroy
 globusftp_client.debugplugin, [57](#)
globusftp_client.debugplugin.init
 globusftp_client.debugplugin, [56](#)
GLOBUS_FTP_CLIENT_DEBUG_PLUGIN_-
 MODULE
 globusftp_client.debugplugin, [56](#)
globusftp_client.delete
 globusftp_client.operations [23](#)
globusftp_client.exists
 globusftp_client.operations [21](#)
globusftp_client.extendedget
 globusftp_client.operations [29](#)
globusftp_client.extendedput
 globusftp_client.operations [31](#)
globusftp_client.extendedthird_party.transfer
 globusftp_client.operations [33](#)
globusftp_client.feats
 globusftp_client.operations [21](#)
globusftp_client.featuresdestroy
 globusftp_client.operations [21](#)
globusftp_client.featuresinit
 globusftp_client.operations [21](#)
globusftp_client.featurest
 globusftp_client.operations [20](#)
globusftp_client.get
 globusftp_client.operations [28](#)
globusftp_client.handle
 globusftp_client.handleadd.plugin, [9](#)
 globusftp_client.handlecacheurl.state, [8](#)
 globusftp_client.handedestroy, [8](#)
 globusftp_client.handle_ush_url.state, [9](#)
 globusftp_client.handleget.userpointer, [10](#)
 globusftp_client.handleinit, [8](#)
 globusftp_client.handlerremoveplugin, [10](#)
 globusftp_client.handleset.userpointer, [9](#)
 globusftp_client.handlelet, [8](#)
globusftp_client.handleadd.plugin
 globusftp_client.handle, [9](#)
globusftp_client.handlecacheurl.state
 globusftp_client.handle, [8](#)
globusftp_client.handedestroy
 globusftp_client.handle, [8](#)
globusftp_client.handle_ush_url.state
 globusftp_client.handle, [9](#)
globusftp_client.handleget.userpointer
 globusftp_client.handle, [10](#)
globusftp_client.handleinit

- globusftp_client.handle, [8](#)
- globusftp_client.handlerremoveplugin
 - globusftp_client.handle, [10](#)
- globusftp_client.handlesetuserpointer
 - globusftp_client.handle, [9](#)
- globusftp_client.handlet
 - globusftp_client.handle, [8](#)
- globusftp_client.handleattr
 - globusftp_client.handleattradd.cachedurl, [14](#)
 - globusftp_client.handleattradd.plugin, [15](#)
 - globusftp_client.handleattrcopy, [13](#)
 - globusftp_client.handleattrdestroy, [12](#)
 - globusftp_client.handleattrget.cacheall, [15](#)
 - globusftp_client.handleattrget.gridftp2, [16](#)
 - globusftp_client.handleattrget.pipeline, [16](#)
 - globusftp_client.handleattrget.rfc1738.url, [15](#)
 - globusftp_client.handleattrinit, [12](#)
 - globusftp_client.handleattrremovecachedurl, [14](#)
 - globusftp_client.handleattrremoveplugin, [16](#)
 - globusftp_client.handleattrset.cacheall, [13](#)
 - globusftp_client.handleattrset.gridftp2, [13](#)
 - globusftp_client.handleattrset.netlogger, [14](#)
 - globusftp_client.handleattrset.netloggerftp.io, [16](#)
 - globusftp_client.handleattrset.pipeline, [14](#)
 - globusftp_client.handleattrset.rfc1738.url, [13](#)
 - globusftp_client.handleattrt, [12](#)
- globusftp_client.handleattradd.cachedurl
 - globusftp_client.handleattr, [14](#)
- globusftp_client.handleattradd.plugin
 - globusftp_client.handleattr, [15](#)
- globusftp_client.handleattrcopy
 - globusftp_client.handleattr, [13](#)
- globusftp_client.handleattrdestroy
 - globusftp_client.handleattr, [12](#)
- globusftp_client.handleattrget.cacheall
 - globusftp_client.handleattr, [15](#)
- globusftp_client.handleattrget.gridftp2
 - globusftp_client.handleattr, [16](#)
- globusftp_client.handleattrget.pipeline
 - globusftp_client.handleattr, [16](#)
- globusftp_client.handleattrget.rfc1738.url
 - globusftp_client.handleattr, [15](#)
- globusftp_client.handleattrinit
 - globusftp_client.handleattr, [12](#)
- globusftp_client.handleattrremovecachedurl
 - globusftp_client.handleattr, [14](#)
- globusftp_client.handleattrremoveplugin
 - globusftp_client.handleattr, [16](#)
- globusftp_client.handleattrset.cacheall
 - globusftp_client.handleattr, [13](#)
- globusftp_client.handleattrset.gridftp2
 - globusftp_client.handleattr, [13](#)
- globusftp_client.handleattrset.netlogger
 - globusftp_client.handleattr, [14](#)
- globusftp_client.handleattrset.netloggerftp.io
 - globusftp_client.handleattr, [16](#)
- globusftp_client.handleattrset.pipeline
 - globusftp_client.handleattr, [14](#)
- globusftp_client.handleattrset.rfc1738.url
 - globusftp_client.handleattr, [13](#)
- globusftp_client.handleattrt
 - globusftp_client.handleattr, [12](#)
- globusftp_client.is_featuresupported
 - globusftp_client.operations, [22](#)
- globusftp_client.list
 - globusftp_client.operations, [24](#)
- globusftp_client.machinelist
 - globusftp_client.operations, [26](#)
- globusftp_client.mkdir
 - globusftp_client.operations, [22](#)
- globusftp_client.mlst
 - globusftp_client.operations, [26](#)
- globusftp_client.modification_time
 - globusftp_client.operations, [33](#)
- GLOBUS_FTP_CLIENT_MODULE
 - globusftp_client.activation, [3](#)
- globusftp_client.move
 - globusftp_client.operations, [27](#)
- globusftp_client.operationattr
 - globusftp_client.operationattrcopy, [46](#)
 - globusftp_client.operationattrdestroy, [39](#)
 - globusftp_client.operationattrget.allocate, [48](#)
 - globusftp_client.operationattrget.allow_ipv6, [52](#)
 - globusftp_client.operationattrget.append, [52](#)
 - globusftp_client.operationattrget.authorization, [51](#)
 - globusftp_client.operationattrget.authzassert, [48](#)
 - globusftp_client.operationattrget.control-protection, [52](#)
 - globusftp_client.operationattrget.data-protection, [52](#)
 - globusftp_client.operationattrget.dcau, [51](#)
 - globusftp_client.operationattrget.disk_stack, [47](#)
 - globusftp_client.operationattrget.layout, [49](#)
 - globusftp_client.operationattrget.list_uses-datamode, [51](#)
 - globusftp_client.operationattrget.mode, [50](#)
 - globusftp_client.operationattrget.net_stack, [47](#)
 - globusftp_client.operationattrget.parallelism, [47](#)
 - globusftp_client.operationattrget.readall, [53](#)

globusftp_client.operationattrget.storage-
 module,46
 globusftp_client.operationattrget.striped,48
 globusftp_client.operationattrget.tcp_buffer, 49
 globusftp_client.operationattrget.type,50
 globusftp_client.operationattrinit, 39
 globusftp_client.operationattrset.allocate,41
 globusftp_client.operationattrset.allow_ipv6,
 45
 globusftp_client.operationattrset.append,45
 globusftp_client.operationattrset.authorization,
 44
 globusftp_client.operationattrset.authzassert,
 41
 globusftp_client.operationattrset.control-
 protection,45
 globusftp_client.operationattrset.data-
 protection,45
 globusftp_client.operationattrset.dcau,44
 globusftp_client.operationattrset.delayedpasv,
 44
 globusftp_client.operationattrset.disk_stack,40
 globusftp_client.operationattrset.layout,42
 globusftp_client.operationattrset.list_uses-
 datamode,43
 globusftp_client.operationattrset.mode,43
 globusftp_client.operationattrset.net_stack,40
 globusftp_client.operationattrset.parallelism,
 40
 globusftp_client.operationattrset.readall, 46
 globusftp_client.operationattrset.storage-
 module,39
 globusftp_client.operationattrset.striped,41
 globusftp_client.operationattrset.tcp_buffer, 42
 globusftp_client.operationattrset.type,43
 globusftp_client.operationattr, 39
 globusftp_client.operationattrcopy
 globusftp_client.operationattr,46
 globusftp_client.operationattrdestroy
 globusftp_client.operationattr,39
 globusftp_client.operationattrget.allocate
 globusftp_client.operationattr,48
 globusftp_client.operationattrget.allow_ipv6
 globusftp_client.operationattr,52
 globusftp_client.operationattrget.append
 globusftp_client.operationattr,52
 globusftp_client.operationattrget.authorization
 globusftp_client.operationattr,51
 globusftp_client.operationattrget.authzassert
 globusftp_client.operationattr,48
 globusftp_client.operationattrget.controlprotection
 globusftp_client.operationattr,52
 globusftp_client.operationattrget.dataprotection
 globusftp_client.operationattr,52
 globusftp_client.operationattrget.dcau
 globusftp_client.operationattr,51
 globusftp_client.operationattrget.disk_stack
 globusftp_client.operationattr,47
 globusftp_client.operationattrget.layout
 globusftp_client.operationattr,49
 globusftp_client.operationattrget.list_usesdata-
 mode
 globusftp_client.operationattr,51
 globusftp_client.operationattrget.mode
 globusftp_client.operationattr,50
 globusftp_client.operationattrget.net_stack
 globusftp_client.operationattr,47
 globusftp_client.operationattrget.parallelism
 globusftp_client.operationattr,47
 globusftp_client.operationattrget.readall
 globusftp_client.operationattr,53
 globusftp_client.operationattrget.storagemodule
 globusftp_client.operationattr,46
 globusftp_client.operationattrget.striped
 globusftp_client.operationattr,48
 globusftp_client.operationattrget.tcp_buffer
 globusftp_client.operationattr,49
 globusftp_client.operationattrget.type
 globusftp_client.operationattr,50
 globusftp_client.operationattrinit
 globusftp_client.operationattr,39
 globusftp_client.operationattrset.allocate
 globusftp_client.operationattr,41
 globusftp_client.operationattrset.allow_ipv6
 globusftp_client.operationattr,45
 globusftp_client.operationattrset.append
 globusftp_client.operationattr,45
 globusftp_client.operationattrset.authorization
 globusftp_client.operationattr,44
 globusftp_client.operationattrset.authzassert
 globusftp_client.operationattr,41
 globusftp_client.operationattrset.controlprotection
 globusftp_client.operationattr,45
 globusftp_client.operationattrset.dataprotection
 globusftp_client.operationattr,45
 globusftp_client.operationattrset.dcau
 globusftp_client.operationattr,44
 globusftp_client.operationattrset.delayedpasv
 globusftp_client.operationattr,44
 globusftp_client.operationattrset.disk_stack
 globusftp_client.operationattr,40
 globusftp_client.operationattrset.layout
 globusftp_client.operationattr,42
 globusftp_client.operationattrset.list_usesdata-
 mode
 globusftp_client.operationattr,43

- globusftp_client.operationattrsetmode
 - globusftp_client.operationattr, 43
- globusftp_client.operationattrsetnetstack
 - globusftp_client.operationattr, 40
- globusftp_client.operationattrsetparallelism
 - globusftp_client.operationattr, 40
- globusftp_client.operationattrsetreadall
 - globusftp_client.operationattr, 46
- globusftp_client.operationattrsetstoragemodule
 - globusftp_client.operationattr, 39
- globusftp_client.operationattrsetstriped
 - globusftp_client.operationattr, 41
- globusftp_client.operationattrsettcpbuffer
 - globusftp_client.operationattr, 42
- globusftp_client.operationattrsettype
 - globusftp_client.operationattr, 43
- globusftp_client.operationattr
 - globusftp_client.operationattr, 39
- globusftp_client.operations
 - globusftp_client.abort, 35
 - globusftp_client.chmod, 28
 - globusftp_client.cksm, 35
 - globusftp_client.completecallbackt, 20
 - globusftp_client.delete, 23
 - globusftp_client.exists, 21
 - globusftp_client.extendedget, 29
 - globusftp_client.extendedput, 31
 - globusftp_client.extendedthird_party_transfer, 33
 - globusftp_client.feats, 21
 - globusftp_client.featuresdestroy, 21
 - globusftp_client.featuresinit, 21
 - globusftp_client.featurest, 20
 - globusftp_client.get, 28
 - globusftp_client.is_featuresupported, 22
 - globusftp_client.list, 24
 - globusftp_client.machinelist, 26
 - globusftp_client.mkdir, 22
 - globusftp_client.mlst, 26
 - globusftp_client.modification_time, 33
 - globusftp_client.move, 27
 - globusftp_client.partialget, 29
 - globusftp_client.partialput, 31
 - globusftp_client.partialthird_party_transfer, 32
 - globusftp_client.probedfeaturet, 20
 - globusftp_client.put, 30
 - globusftp_client.rmdir, 23
 - globusftp_client.size, 34
 - globusftp_client.stat, 25
 - globusftp_client.third_party_transfer, 32
 - globusftp_client.tristatet, 20
 - globusftp_client.verbooselist, 25
- globusftp_client.partialget
 - globusftp_client.operations, 29
- globusftp_client.partialput
 - globusftp_client.operations, 31
- globusftp_client.partialthird_party_transfer
 - globusftp_client.operations, 32
- globusftp_client.perf.plugin
 - globusftp_client.perf.plugin.begin.cb.t, 58
 - globusftp_client.perf.plugin.completecb.t, 59
 - globusftp_client.perf.plugin.destroy, 60
 - globusftp_client.perf.plugin.getuserspeci c, 60
 - globusftp_client.perf.plugin.init, 60
 - globusftp_client.perf.plugin.markercb.t, 58
 - GLOBUS_FTP_CLIENT_PERF_PLUGIN_MODULE, 58
 - globusftp_client.perf.plugin.setcopy_destroy, 60
 - globusftp_client.perf.plugin.usercopy.cb.t, 59
 - globusftp_client.perf.plugin.userdestroycb.t, 59
- globusftp_client.perf.plugin.begin.cb.t
 - globusftp_client.perf.plugin, 58
- globusftp_client.perf.plugin.completecb.t
 - globusftp_client.perf.plugin, 59
- globusftp_client.perf.plugin.destroy
 - globusftp_client.perf.plugin, 60
- globusftp_client.perf.plugin.getuserspeci c
 - globusftp_client.perf.plugin, 60
- globusftp_client.perf.plugin.init
 - globusftp_client.perf.plugin, 60
- globusftp_client.perf.plugin.markercb.t
 - globusftp_client.perf.plugin, 58
- GLOBUS_FTP_CLIENT_PERF_PLUGIN_MODULE
 - globusftp_client.perf.plugin, 58
- globusftp_client.perf.plugin.setcopy_destroy
 - globusftp_client.perf.plugin, 60
- globusftp_client.perf.plugin.usercopy.cb.t
 - globusftp_client.perf.plugin, 59
- globusftp_client.perf.plugin.userdestroycb.t
 - globusftp_client.perf.plugin, 59
- globusftp_client.plugin_abort
 - globusftp_client.plugins, 84
- globusftp_client.plugin_abortt
 - globusftp_client.plugins, 73
- globusftp_client.plugin_add_datachannels
 - globusftp_client.plugins, 84
- globusftp_client.plugin_authentication
 - globusftp_client.plugins, 65
- globusftp_client.plugin_chmodt
 - globusftp_client.plugins, 66
- globusftp_client.plugin_cksm
 - globusftp_client.plugins, 66
- globusftp_client.plugin_commandmaskt

- globusftp_client.plugins,76
- globusftp_client.plugin_commandt
 - globusftp_client.plugins,74
- globusftp_client.plugin_completet
 - globusftp_client.plugins,75
- globusftp_client.plugin_connectt
 - globusftp_client.plugins,65
- globusftp_client.plugin_copy.t
 - globusftp_client.plugins,64
- globusftp_client.plugin_datat
 - globusftp_client.plugins,74
- globusftp_client.plugin_deletet
 - globusftp_client.plugins,67
- globusftp_client.plugin_destroyt
 - globusftp_client.plugins,65
- globusftp_client.plugin_fault.t
 - globusftp_client.plugins,75
- globusftp_client.plugin_feat.t
 - globusftp_client.plugins,67
- globusftp_client.plugin_get.t
 - globusftp_client.plugins,71
- globusftp_client.plugin_list.t
 - globusftp_client.plugins,68
- globusftp_client.plugin_machinelist.t
 - globusftp_client.plugins,69
- globusftp_client.plugin_mkdir.t
 - globusftp_client.plugins,67
- globusftp_client.plugin_mlst.t
 - globusftp_client.plugins,69
- globusftp_client.plugin_modification.time.t
 - globusftp_client.plugins,72
- globusftp_client.plugin_move.t
 - globusftp_client.plugins,70
- globusftp_client.plugin_put.t
 - globusftp_client.plugins,71
- globusftp_client.plugin_readt
 - globusftp_client.plugins,73
- globusftp_client.plugin_removedatachannels
 - globusftp_client.plugins,84
- globusftp_client.plugin_response
 - globusftp_client.plugins,74
- globusftp_client.plugin_restartchmod
 - globusftp_client.plugins,78
- globusftp_client.plugin_restartcksm
 - globusftp_client.plugins,78
- globusftp_client.plugin_restartdelete
 - globusftp_client.plugins,79
- globusftp_client.plugin_restartfeat
 - globusftp_client.plugins,79
- globusftp_client.plugin_restartget
 - globusftp_client.plugins,81
- globusftp_client.plugin_restartget.marker
 - globusftp_client.plugins,83
- globusftp_client.plugin_restartlist
 - globusftp_client.plugins,76
- globusftp_client.plugin_restartmachinelist
 - globusftp_client.plugins,77
- globusftp_client.plugin_restartmkdir
 - globusftp_client.plugins,80
- globusftp_client.plugin_restartmlst
 - globusftp_client.plugins,77
- globusftp_client.plugin_restartmodification.time
 - globusftp_client.plugins,83
- globusftp_client.plugin_restartmove
 - globusftp_client.plugins,80
- globusftp_client.plugin_restartput
 - globusftp_client.plugins,81
- globusftp_client.plugin_restartrmdir
 - globusftp_client.plugins,80
- globusftp_client.plugin_restartsize
 - globusftp_client.plugins,83
- globusftp_client.plugin_restartstat
 - globusftp_client.plugins,77
- globusftp_client.plugin_restartthird.party.transfer
 - globusftp_client.plugins,82
- globusftp_client.plugin_restartverboselist
 - globusftp_client.plugins,76
- globusftp_client.plugin_rmdir.t
 - globusftp_client.plugins,68
- globusftp_client.plugin_size.t
 - globusftp_client.plugins,72
- globusftp_client.plugin_stat.t
 - globusftp_client.plugins,70
- globusftp_client.plugin.t
 - globusftp_client.plugins,64
- globusftp_client.plugin_third.party.transfert
 - globusftp_client.plugins,71
- globusftp_client.plugin_verboselist.t
 - globusftp_client.plugins,69
- globusftp_client.plugin_write.t
 - globusftp_client.plugins,73
- globusftp_client.plugins
 - GLOBUS.FTP.CLIENT.CMD.MASK.ALL, 76
 - GLOBUS.FTP.CLIENT.CMD.MASK.-
 - BUFFER,76
 - GLOBUS.FTP.CLIENT.CMD.MASK.-
 - CONTROLESTABLISHMENT,76
 - GLOBUS.FTP.CLIENT.CMD.MASK.DATA.-
 - ESTABLISHMENT,76
 - GLOBUS.FTP.CLIENT.CMD.MASK.FILE.-
 - ACTIONS,76
 - GLOBUS.FTP.CLIENT.CMD.MASK.-
 - INFORMATION,76
 - GLOBUS.FTP.CLIENT.CMD.MASK.MISC,
 - 76

- GLOBUS.FTP.CLIENT_CMD_MASK_-TRANSFERMODIFIERS, 76
 - GLOBUS.FTP.CLIENT_CMD_MASK_-TRANSFERPARAMETERS, 76
- globusftp_client_plugins
 - globusftp_client.plugin.abort, 84
 - globusftp_client.plugin.abortt, 73
 - globusftp_client.plugin.add_datachannels, 84
 - globusftp_client.plugin.authenticate, 65
 - globusftp_client.plugin.chmodt, 66
 - globusftp_client.plugin.cksm_t, 66
 - globusftp_client.plugin.commandmaskt, 76
 - globusftp_client.plugin.commandt, 74
 - globusftp_client.plugin.completet, 75
 - globusftp_client.plugin.connectt, 65
 - globusftp_client.plugin.copy_t, 64
 - globusftp_client.plugin.datat, 74
 - globusftp_client.plugin.deletet, 67
 - globusftp_client.plugin.destroyt, 65
 - globusftp_client.plugin.fault_t, 75
 - globusftp_client.plugin.featt, 67
 - globusftp_client.plugin.get_t, 71
 - globusftp_client.plugin.list_t, 68
 - globusftp_client.plugin.machinelist_t, 69
 - globusftp_client.plugin.mkdir_t, 67
 - globusftp_client.plugin.mlst_t, 69
 - globusftp_client.plugin.moderation_time_t, 72
 - globusftp_client.plugin.movet, 70
 - globusftp_client.plugin.put_t, 71
 - globusftp_client.plugin.readt, 73
 - globusftp_client.plugin.removedatachannels, 84
 - globusftp_client.plugin.respondt, 74
 - globusftp_client.plugin.restartchmod, 78
 - globusftp_client.plugin.restartcksm, 78
 - globusftp_client.plugin.restartdelete, 79
 - globusftp_client.plugin.restartfeat, 79
 - globusftp_client.plugin.restartget, 81
 - globusftp_client.plugin.restartget.marker, 83
 - globusftp_client.plugin.restartlist, 76
 - globusftp_client.plugin.restartmachinelist, 77
 - globusftp_client.plugin.restartmkdir, 80
 - globusftp_client.plugin.restartmlst, 77
 - globusftp_client.plugin.restartmoderation_time, 83
 - globusftp_client.plugin.restartmove, 80
 - globusftp_client.plugin.restartput, 81
 - globusftp_client.plugin.restartmkdir, 80
 - globusftp_client.plugin.restartsize, 83
 - globusftp_client.plugin.restartstat, 77
 - globusftp_client.plugin.restartthird_party_transfer, 82
 - globusftp_client.plugin.restartverboselect, 76
 - globusftp_client.plugin.rmdir_t, 68
 - globusftp_client.plugin.size_t, 72
 - globusftp_client.plugin.stat_t, 70
 - globusftp_client.plugin.t, 64
 - globusftp_client.plugin.third_party.transfert, 71
 - globusftp_client.plugin.verboselect_t, 69
 - globusftp_client.plugin.write_t, 73
- globusftp_client.probedfeaturet
 - globusftp_client.operations, 20
- globusftp_client.put
 - globusftp_client.operations, 30
- globusftp_client.registerread
 - globusftp_client.data, 54
- globusftp_client.registerwrite
 - globusftp_client.data, 54
- globusftp_client.restartextendedblock_t, 97
- globusftp_client.restartmarker
 - globusftp_client.restartmarker.copy, 4
 - globusftp_client.restartmarker.destroy, 4
 - globusftp_client.restartmarker.from_string, 6
 - globusftp_client.restartmarker.gettotal, 6
 - globusftp_client.restartmarker.init, 4
 - globusftp_client.restartmarker.insertrange, 4
 - globusftp_client.restartmarker.setascii_offset, 5
 - globusftp_client.restartmarker.set_offset, 5
 - globusftp_client.restartmarker.to_string, 6
- globusftp_client.restartmarker.copy
 - globusftp_client.restartmarker, 4
- globusftp_client.restartmarker.destroy
 - globusftp_client.restartmarker, 4
- globusftp_client.restartmarker.from_string
 - globusftp_client.restartmarker, 6
- globusftp_client.restartmarker.gettotal
 - globusftp_client.restartmarker, 6
- globusftp_client.restartmarker.init
 - globusftp_client.restartmarker, 4
- globusftp_client.restartmarker.insertrange
 - globusftp_client.restartmarker, 4
- globusftp_client.restartmarker.plugin
 - globusftp_client.restartmarker.plugin.begin_cb_t, 86
 - globusftp_client.restartmarker.plugin.completecb_t, 86
 - globusftp_client.restartmarker.plugin.destroy, 87
 - globusftp_client.restartmarker.plugin.init, 87
 - globusftp_client.restartmarker.plugin.marker_cb_t, 86
- GLOBUS.FTP.CLIENT.RESTART-MARKER_PLUGIN.MODULE, 85
- globusftp_client.restartmarker.plugin.begin.cb.t

- globusftp_client.restartmarkerplugin, [86](#)
- globusftp_client.restartmarkerplugin.completecb.t
 - globusftp_client.restartmarkerplugin, [86](#)
- globusftp_client.restartmarkerplugin.destroy
 - globusftp_client.restartmarkerplugin, [87](#)
- globusftp_client.restartmarkerplugin.init
 - globusftp_client.restartmarkerplugin, [87](#)
- globusftp_client.restartmarkerplugin.markercb.t
 - globusftp_client.restartmarkerplugin, [86](#)
- GLOBUS.FTP.CLIENT.RESTART.MARKER.-
PLUGIN.MODULE
 - globusftp_client.restartmarkerplugin, [85](#)
- globusftp_client.restartmarkersetascii.offset
 - globusftp_client.restartmarker, [5](#)
- globusftp_client.restartmarkersetoffset
 - globusftp_client.restartmarker, [5](#)
- globusftp_client.restartmarkert, [97](#)
- globusftp_client.restartmarkerto.string
 - globusftp_client.restartmarker, [6](#)
- globusftp_client.restartplugin
 - globusftp_client.restartplugin.destroy, [89](#)
 - globusftp_client.restartplugin.init, [89](#)
- GLOBUS.FTP.CLIENT.RESTART.PLUGIN.-
MODULE, [89](#)
- globusftp_client.restartplugin.destroy
 - globusftp_client.restartplugin, [89](#)
- globusftp_client.restartplugin.init
 - globusftp_client.restartplugin, [89](#)
- GLOBUS.FTP.CLIENT.RESTART.PLUGIN.-
MODULE
 - globusftp_client.restartplugin, [89](#)
- globusftp_client.restartstreamt, [97](#)
- globusftp_client.rmdir
 - globusftp_client.operations, [23](#)
- globusftp_client.size
 - globusftp_client.operations, [34](#)
- globusftp_client.stat
 - globusftp_client.operations, [25](#)
- globusftp_client.third_party.transfer
 - globusftp_client.operations, [32](#)
- globusftp_client.throughputnl_plugin
 - globusftp_client.throughputnl_plugin.destroy, [92](#)
 - globusftp_client.throughputnl_plugin.init, [91](#)
 - globusftp_client.throughputnl_plugin.init.-
with_handle, [91](#)
- GLOBUS.FTP.CLIENT.THROUGHPUTNL.-
PLUGIN.MODULE, [91](#)
- globusftp_client.throughputnl_plugin.set-
callbacks, [92](#)
- globusftp_client.throughputnl_plugin.destroy
 - globusftp_client.throughputnl_plugin, [92](#)
- globusftp_client.throughputnl_plugin.init
 - globusftp_client.throughputnl_plugin, [91](#)
- globusftp_client.throughputnl_plugin.init.-
with_handle
 - globusftp_client.throughputnl_plugin, [91](#)
- GLOBUS.FTP.CLIENT.THROUGHPUTNL.-
PLUGIN.MODULE
 - globusftp_client.throughputnl_plugin, [91](#)
- globusftp_client.throughputnl_plugin.setcallbacks
 - globusftp_client.throughputnl_plugin, [92](#)
- globusftp_client.throughputplugin
 - globusftp_client.throughputplugin.begin.cb.t, [94](#)
 - globusftp_client.throughputplugin.complete-
cb.t, [94](#)
 - globusftp_client.throughputplugin.destroy, [96](#)
 - globusftp_client.throughputplugin.getuser-
speci c, [96](#)
 - globusftp_client.throughputplugin.init, [95](#)
- GLOBUS.FTP.CLIENT.THROUGHPUT-
PLUGIN.MODULE, [93](#)
- globusftp_client.throughputplugin.setcopy.-
destroy, [96](#)
- globusftp_client.throughputplugin.stripe.cb.t, [94](#)
- globusftp_client.throughputplugin.total.cb.t, [94](#)
- globusftp_client.throughputplugin.usercopy.-
cb.t, [95](#)
- globusftp_client.throughputplugin.user-
destroycb.t, [95](#)
- globusftp_client.throughputplugin.begin.cb.t
 - globusftp_client.throughputplugin, [94](#)
- globusftp_client.throughputplugin.completecb.t
 - globusftp_client.throughputplugin, [94](#)
- globusftp_client.throughputplugin.destroy
 - globusftp_client.throughputplugin, [96](#)
- globusftp_client.throughputplugin.getuserspeci c
 - globusftp_client.throughputplugin, [96](#)
- globusftp_client.throughputplugin.init
 - globusftp_client.throughputplugin, [95](#)
- GLOBUS.FTP.CLIENT.THROUGHPUT-
PLUGIN.MODULE
 - globusftp_client.throughputplugin, [93](#)
- globusftp_client.throughputplugin.setcopy.destroy
 - globusftp_client.throughputplugin, [96](#)
- globusftp_client.throughputplugin.stripe.cb.t
 - globusftp_client.throughputplugin, [94](#)
- globusftp_client.throughputplugin.total.cb.t
 - globusftp_client.throughputplugin, [94](#)
- globusftp_client.throughputplugin.usercopy.cb.t
 - globusftp_client.throughputplugin, [95](#)
- globusftp_client.throughputplugin.userdestroycb.t
 - globusftp_client.throughputplugin, [95](#)

globus.ftp_client.tristate
 globus.ftp_client.operations [20](#)
globus.ftp_client.verbose
 globus.ftp_client.operations [25](#)

Handle Attributes [10](#)
Handle Management [7](#),

Netlogger Throughput Plugin [90](#)

Performance Marker Plugin [57](#)
Plugins [61](#)

Reading and Writing Data [53](#)
Restart Marker Plugin [85](#)
Restart Markers [3](#)
Restart Plugin [88](#)

Throughput Performance Plugin [93](#)