

GT 4.2.1 GridFTP : Developer's Guide

GT 4.2.1 GridFTP : Developer's Guide

Introduction

This guide contains information of interest to developers working with GridFTP. It provides reference information for application developers, including APIs, architecture, procedures for using the APIs and code samples.

Table of Contents

1. Before you begin	1
1. Feature Summary	1
2. Tested platforms	2
3. Backward compatibility summary	2
4. Technology dependencies	3
2. Usage Scenarios	4
3. Tutorials	5
4. Architecture and design overview	6
1. GridFTP Listening	6
2. GridFTP Transfer	6
3. GridFTP Server and DSIs	6
5. API Summary	9
1. Programming Model Overview	9
2. Component API	9
I. GridFTP Commands	11
globus-url-copy	12
globus-gridftp-server	22
6. Graphical User Interface	32
7. Configuring GridFTP	33
1. GridFTP server configuration overview	33
2. Types of configurations	33
3. globus-gridftp-server quickstart	34
4. Running in daemon mode	35
5. Running under inetd or xinetd	35
8. Environment variable interface	37
1. Environment variables for GridFTP	37
9. Debugging	38
10. Troubleshooting	39
1. Error Codes in GridFTP	40
2. Establish control channel connection	40
3. Try running globus-url-copy	41
4. If your server starts... ..	42
5. High latency for GridFTP server connections	42
11. Related Documentation	43
A. Developing DSIs for GridFTP	44
1. DSI Interface	44
2. DSI utility API	44
3. Implementation	44
4. DSI Bones	46
B. GridFTP Multicast	47
1. Architecture	47
2. Globus XIO	47
3. Network Overlay	48
4. Results	48
5. Protocol Details	49
6. Usage	49
Glossary	51
Index	54

List of Figures

1. Effect of Parallel Streams in GridFTP	20
B.1. GridFTP Spanning Tree	47
B.2. From client to server through Internet	48
B.3. Routing data through network via multicast	48
B.4.	49
B.5.	49
B.6.	49

List of Tables

1. URL formats	14
10.1. GridFTP Errors	40

Chapter 1. Before you begin

1. Feature Summary

Features new in GT 4.2.1

- [GridFTP over UDT](#)
- [SSH security for GridFTP control channel](#)
- [Running the GridFTP server with GFork GridFTP](#)
- [Multicasting / Network overlays](#)
- [Netlogger's bottleneck detection for GridFTP transfers](#)¹

Features that continue to be supported from previous versions

- GSI security: This is the PKI based, de facto standard security system used in Grid applications. Kerberos is also possible but is not supported and can be difficult to use due to divergence in the capabilities of GSI and Kerberos.
- Third-party transfers: Very common in Grid applications, this is where a *client* mediates a transfer between two servers (both likely at remote sites) rather than between the server and itself (called a *client/server transfer*).
- Cluster-to-cluster data movement or Striping: GridFTP can do coordinated data transfer by using multiple computer nodes at the source and destination.
- Partial file access: Regions of a file may be accessed by specifying an offset into the file and the length of the block desired.
- Reliability/restart: The receiving server periodically (the default is 5 seconds, but this can be changed) sends “restart markers” to the client. This marker is a messages specifying what bytes have been successfully written to the disk. If the transfer fails, the client may restart the transfer and provide these markers (or an aggregated equivalent marker), and the transfer will pick up where it left off. This can include “holes” in the file.
- Large file support: All file sizes, lengths, and offsets are 64 bits in length.
- Data channel reuse: Data channel can be held open and reused if the next transfer has the same source, destination, and credentials. This saves the time of connection establishment, authentication, and delegation. This can be a huge performance difference when moving lots of small files.
- Integrated instrumentation (Performance Markers).
- Logging/audit trail (Extensive Logging in the server).
- Parallel transfers (Multiple TCP streams between a pair of hosts).
- TCP Buffer size control (Protocol supports Manual and Automatic; Only Manual Implemented).
- Server-side computation (Extended Retrieve (ERET) / Extended Store (ESTO) commands).
- Based on Standards: RFC 959, RFC 2228, RFC 2389, IETF Draft MLST-16 , GGF GFD.020.

¹ <http://www.cedps.net/index.php/Gridftp-netlogger>

Other Supported Features

- On the client side we provide a scriptable tool called globus-url-copy. This tool can take advantage of all the GridFTP protocol features and can also do protocol translation between FTP, HTTP, HTTPS, and POSIX file IO on the client machine.
- We also provide a set of development libraries and APIs for developers wishing to add GridFTP functionality to their application.

Deprecated Features

- None

2. Tested platforms

Tested platforms for GridFTP

- i386 Linux
- ia64 Linux (TeraGrid)
- AIX 5.2
- Solaris 9
- PA-RISC HP/UX 11.11
- ia64 HP/UX 11.22
- Tru64 Unix
- Mac OS X

While the above list includes platforms on which we have tested GridFTP, it does not imply support for a specific platform. However, we are interested in hearing reports of success or bug reports on any platform.

3. Backward compatibility summary

Protocol changes since GT 4.0.x

- None

API changes since GT 4.0.x

- None

Exception changes since GT 4.0.x

- Not Applicable (GridFTP is not Java-based)

Schema changes since GT 4.0.x

- Not Applicable (GridFTP is not SOAP-based)

4. Technology dependencies

GridFTP depends on the following GT components:

- Non-WS (General) Authentication & Authorization
- C Common Libraries
- XIO

GridFTP depends on the following 3rd party software:

- OpenSSL (version is included in release)

Chapter 2. Usage Scenarios

Do you have a need to move large quantities of data rapidly and reliably to remote locations? Globus GridFTP is a software suite optimized for the gamut of data access issues — from bulk file transfer to the details of getting data out of complex storage systems within sites.

Are you concerned about authenticating and authorizing the users? Globus GridFTP supports various authentication and authorization mechanisms. In fact, it is easy to plugin in different authorization mechanism.

Do you need to move data in and out of a complex storage systems? In addition to the POSIX file systems, Globus GridFTP can move data in and out of HPSS and SRB. Capability to access other storage systems can be easily added by implementing a well-defined pluggable interface called Data Storage Interface (DSI).

Do you need transfer huge volume of data and you do not want to babysit the transfer? Use the GridFTP client 'Reliable File Transfer Service'.

Do you need to protect the data while moving it to the remote location? Globus GridFTP provides support for data integrity and data encryption.

Do you need move your data to remote location that is far away and TCP limits the performance? Globus GridFTP supports UDT as an alternate transport protocol for UDT. GridFTP also allows you to use parallel TCP streams.

Do you want to move data from one location to many locations efficiently? Multicasting capability in Globus GridFTP allows you to do it.

Do you have a network link between source and destination that supports much higher data rate than the data rate supported by individual nodes at either end? GridFTP supports cluster-to-cluster data movement – coordinated data transfer using multiple computer nodes at source and destination.

Do you want to limit the resources used for each client connection on the server node? Globus GridFTP can be configured to run with GFork to limit the resource usage. Customized resource control modules can be plugged in easily.

Globus GridFTP has pervasive use in the e-Science Grid community. The high energy physics community in particular has been a huge user from the start. The Relativistic Heavy Ion Collider (RHIC) community in Brookhaven used Globus GridFTP to sustain 600 megabytes per second of data transfer (from Long Island, New York, to Japan) over 11 days.

Frequent large file transfer demands for the British Broadcasting Corporation (BBC) are met by GridFTP. Typical broadcast hour today requires 280 GB for all pre-processed media streams. "Everything in Gridcast is built using Globus Toolkit," said Terry Harmer, Technical Director at the Belfast e-Science Centre, in an interview (<http://www.globusconsortium.org/journal/20050816/harmer.html>) in '05 with the Globus Consortium Journal. "We use it as a means by which we create, define, and deploy services. We are big users of GridFTP."

Recently, US Department of Energy's Advanced Photon Source user facility at Argonne transferred more than a terabyte of data (partitioned into lots of small files) to Australia using GridFTP at a rate 30 times faster than traditional data transfer mechanisms such as SCP. The Laser Interferometer Gravitational Wave Observatory (LIGO) project moved 1.5 Terabytes of data from University of Wisconsin at Milwaukee, USA to Hannover, Germany at a sustained rate of 80 MB/s.

Chapter 3. Tutorials

There is an online tutorial available at: <http://gridftp.globus.org/tutorials/>.

Chapter 4. Architecture and design overview

GridFTP represents a service that a host is providing. Therefore, the service must be listening on a port waiting for *client* to request access to that service. This is generally handled one of two ways:

- Either an application daemon is running listening for connections, or
- inetd/xinetd is used.

1. GridFTP Listening

The following list describes the process between the service listening for connection and an exchange of data taking place:

1. These services (application daemon or inetd/xinetd) listen for connections.
2. When a connection is received on a "well known" port such as 2811 for GridFTP, inetd does a fork/exec to start up a GridFTP *server* process and then does a Switch User (SU) so that the server is running in a user account rather than as root for security reasons. At this point, the client has established a control channel to the server.
3. The client will then send a series of commands to configure or describe the transfer that it wants to take place.

2. GridFTP Transfer

There are basically four important components of the exchange:

1. The first is security. You must authenticate, and for GridFTP, you must establish encryption on the control channel. The control channel is encrypted by default, though it can be switched off (see the security section for more detail).
2. The second is setup and informational exchanges. The client may specify the type of the file (Binary or ASCII), the *MODE* of the transfer, he might request the size of a file before transferring it, etc..
3. Third, the information and negotiation for the data channel must be done. How this is handled, depends on whether you are doing a *client/server transfer* or *third party transfer*.
4. Finally, a store (STOR), retrieve (RETR), extended store (ESTO) or extended retrieve (ERET) to indicate direction of the transfer and to start data moving.

3. GridFTP Server and DSIs

3.1. GridFTP and DSIs

The following information is helpful if you want to use GridFTP to access data in DSIs (such as HPSS and SRB), and non-POSIX data sources.

Architecturally, the Globus GridFTP *server* can be divided into 3 modules:

- the GridFTP protocol module,

- the (optional) data transform module, and
- the Data Storage Interface (DSI).

In the GT 4.2.1 implementation, the data transform module and the DSI have been merged, although we plan to have separate, chainable, data transform modules in the future.

Note

This architecture does NOT apply to the WU-FTPD implementation (GT3.2.1 and lower).

3.1.1. GridFTP Protocol Module

The GridFTP protocol module is the module that reads and writes to the network and implements the GridFTP protocol. This module should not need to be modified since to do so would make the server non-protocol compliant, and unable to communicate with other servers.

3.1.2. Data Transform Functionality

The data transform functionality is invoked by using the ERET (extended retrieve) and ESTO (extended store) commands. It is seldom used and bears careful consideration before it is implemented, but in the right circumstances can be very useful. In theory, any computation could be invoked this way, but it was primarily intended for cases where some simple pre-processing (such as a partial get or sub-sampling) can greatly reduce the network load. The disadvantage to this is that you remove any real option for planning, brokering, etc., and any significant computation could adversely affect the data transfer performance. Note that the *client* must also support the ESTO/ERET functionality as well.

3.1.3. Data Storage Interface (DSI) / Data Transform module

The Data Storage Interface (DSI) / Data Transform module knows how to read and write to the "local" storage system and can optionally transform the data. We put local in quotes because in a complicated storage system, the storage may not be directly attached, but for performance reasons, it should be relatively close (for instance on the same LAN).

The interface consists of functions to be implemented such as send (get), receive (put), command (simple commands that simply succeed or fail like mkdir), etc..

Once these functions have been implemented for a specific storage system, a client should not need to know or care what is actually providing the data. The server can either be configured specifically with a specific DSI, i.e., it knows how to interact with a single class of storage system, or one particularly useful function for the ESTO/ERET functionality mentioned above is to load and configure a DSI on the fly.

See [Appendix A, Developing DSIs for GridFTP](#) for more information.

3.2. Latest information about HPSS

Last Update: August 2005

Working with Los Alamos National Laboratory and the High Performance Storage System (HPSS) collaboration (<http://www.hpss-collaboration.org>), we have written a Data Storage Interface (DSI) for read/write access to HPSS. This DSI would allow an existing application that uses a GridFTP compliant client to utilize an HPSS data resources.

This DSI is currently in testing. Due to changes in the HPSS security mechanisms, it requires HPSS 6.2 or later, which is due to be released in Q4 2005. Distribution for the DSI has not been worked out yet, but it will *probably* be available from both Globus and the HPSS collaboration. While this code will be open source, it requires underlying HPSS libraries which are NOT open source (proprietary).



Note

This is a purely server side change, the client does not know what DSI is running, so only a site that is already running HPSS and wants to allow GridFTP access needs to worry about access to these proprietary libraries.

3.3. Latest information about SRB

Last Update: August 2005

Working with the SRB team at the San Diego Supercomputing Center, we have written a Data Storage Interface (DSI) for read/write access to data in the Storage Resource Broker (SRB) (<http://www.npaci.edu/DICE/SRB>). This DSI will enable GridFTP compliant clients to read and write data to an SRB server, similar in functionality to the sput/sget commands.

This DSI is currently in testing and is not yet publicly available, but will be available from both the SRB web site ([here](#)) and the Globus web site ([here](#)). It will also be included in the next stable release of the toolkit. We are working on performance tests, but early results indicate that for wide area network (WAN) transfers, the performance is comparable.

When might you want to use this functionality:

- You have existing tools that use GridFTP clients and you want to access data that is in SRB
- You have distributed data sets that have some of the data in SRB and some of the data available from GridFTP servers.

Chapter 5. API Summary

1. Programming Model Overview

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 [GridFTP Protocol document](#)¹.

In addition to protocol support for grid applications, the FTP Client library provides a [plugin architecture](#)² for installing application or grid-specific fault recovery and performance tuning algorithms within the library. Application writers may then target their code toward the FTP Client library and, by simply enabling the appropriate plugins, easily tune their application to run it on a different grid.

All applications which use the Globus FTP Client API must include the header file `globus_ftp_client.h` and activate the `GLOBUS_FTP_CLIENT_MODULE`³.

To use the Globus FTP Client API, one must create an [FTP Client handle](#)⁴. This structure contains:

- context information about FTP operations which are being executed,
- a cache of FTP control and data connections, and
- information about plugins which are being used.

The specifics of the connection caching and plugins are found in the "[Handle Attributes](#)"⁵ section of the API documentation.

Once the handle is created, one may begin transferring files or doing other FTP operations by calling the functions in the "[FTP Operations](#)"⁶ section of the API documentation. 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 `operationattr` section, associated with them to tune various FTP parameters. The data structures and functions needed to restart a file transfer are described in the "[Restart Markers](#)"⁷ section of the API documentation. For operations which require the user to send to or receive data from an FTP *server* they must call the functions described in the "`globus_ftp_client_data`" section of the manual.

The `globus_ftp_control` library provides low-level services needed to implement FTP clients and servers. The API provided is protocol specific. The data transfer portion of this API provides support for the standard data methods described in the FTP Specification as well as extensions for parallel, striped, and partial data transfer.

2. Component API

- [C Client Library API](#)⁸
- [C Control Library API](#)⁹

¹ <http://www.globus.org/alliance/publications/papers/GFD-R.0201.pdf>

² http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_plugins.html

³ http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_activation.html

⁴ http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_handle.html

⁵ http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_handleattr.html

⁶ http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_operations.html

⁷ http://www.globus.org/api/c-globus-4.2.1/globus_ftp_client/html/group_globus_ftp_client_restart_marker.html

⁸ http://www.globus.org/api/c-globus-3.9.x/globus_ftp_client/html/index.html

⁹ http://www.globus.org/api/c-globus-3.9.x/globus_ftp_control/html/index.html

For information on the internationalization API, see [Chapter 1, APIs](#).

GridFTP Commands

Name

globus-url-copy -- Multi-protocol data movement

globus-url-copy

Tool description

globus-url-copy is a scriptable command line tool that can do multi-protocol data movement. It supports gsiftp:// (GridFTP), ftp://, http://, https://, and file:/// protocol specifiers in the URL. For GridFTP, globus-url-copy supports all implemented functionality. Versions from GT 3.2 and later support file globbing and directory moves.

- [Before you begin](#)
- [Command syntax](#)
- [Command line options](#)
 - [Informational options](#)
 - [Utility options](#)
 - [Reliability options](#)
 - [Performance options](#)
 - [Security-related options](#)
- [Default usage](#)
- [MODES in GridFTP](#)
- [If you run a GridFTP server by hand](#)
- [How do I choose a value for the TCP buffer size \(-tcp-bs\) option?](#)
- [How do I choose a value for the parallelism \(-p\) option?](#)
- [Limitations](#)
- [Interactive clients for GridFTP](#)

Before you begin

Important

To use gsiftp:// and https:// protocols, you must have a [certificate](#) to use globus-url-copy. However, you may use ftp:// or http:// protocols without a certificate.

1. First, as with all things Grid, you *must* have a valid proxy certificate to run globus-url-copy in certain protocols (gsiftp:// and https://, as noted above). If you are using ftp:// or http:// protocols, security is *not* mandatory and you may skip the rest of this table.

If you do not have a certificate, you must [obtain one](#).

If you are doing this for testing in your own environment, the [SimpleCA](#) provided with the Globus Toolkit should suffice.

If not, you must contact the Virtual Organization (VO) with which you are associated to find out whom to ask for a certificate.

One common source is the [DOE Science Grid CA](#)¹, although you must confirm whether or not the resources you wish to access will accept their certificates.

Instructions for proper installation of the certificate should be provided from the source of the certificate.

Please note when your certificates expire; they will need to be renewed or you may lose access to your resources.

2. Now that you have a certificate, you must generate a temporary proxy. Do this by running:

```
grid-proxy-init
```

Further documentation for **grid-proxy-init** can be found [here](#).

3. You are now ready to use **globus-url-copy**! See the following sections for syntax and command line options and other considerations.

Command syntax

The basic syntax for **globus-url-copy** is:

```
globus-url-copy [optional command line switches] Source_URL Destination_URL
```

where:

[optional command line switches]	See Command line options below for a list of available options.
<i>Source_URL</i>	Specifies the original URL of the file(s) to be copied. If this is a directory, all files within that directory will be copied.
<i>Destination_URL</i>	Specifies the URL where you want to copy the files. If you want to copy multiple files, this must be a directory.

Note

Any url specifying a directory must end with */*.

URL prefixes

As of GT 3.2, we support the following URL prefixes:

- **file://** (on a local machine only)
- **ftp://**
- **gsiftp://**
- **http://**

¹ <http://www.doe grids.org/pages/cert-request.htm>

- **https://**

By default, **globus-url-copy** expects the same kind of host certificates that **globusrun** expects from gatekeepers.



Note

We do *not* provide an interactive client similar to the generic FTP client provided with Linux. See the [Interactive Clients](#) section below for information on an interactive client developed by NCSA/NMI/TeraGrid.

URL formats

URLs can be any valid URL as defined by RFC 1738 that have a protocol we support. In general, they have the following format: ***protocol://host:port/path***.



Note

If the path ends with a trailing / (i.e. `/path/to/directory/`) it will be considered to be a directory and all files in that directory will be moved. If you want a recursive directory move, you need to add the `-r/-recurse` switch described below.

Table 1. URL formats

<code>gsiftp://myhost.mydomain.com:2812/data/foo.dat</code>	Fully specified.
<code>http://myhost.mydomain.com/mywebpage/default.html</code>	Port is not specified; therefore, GridFTP uses protocol default (in this case, 80).
<code>file:///foo.dat</code>	Host is not specified; therefore, GridFTP uses your local host. Port is not specified; therefore, GridFTP uses protocol default (in this case, 80).
<code>file:/foo.dat</code>	This is also valid but is not recommended because, while many servers (including ours) accept this format, it is <i>not</i> RFC conformant and is not recommended.



Important

For GridFTP (`gsiftp://`) and FTP (`ftp://`), it is legal to specify a user name and password in the the URL as follows:

```
gsiftp://myname:[mypassword]@myhost.mydomain.com/foo.dat
```

If you are using GSI security, then you may specify the username (but you may *not* include the `:` or the password) and the `grid-mapfile` will be searched to see if that is a valid account mapping for your distinguished name (DN). If it is found, the *server* will setuid to that account. If not, it will fail. It will NOT fail back to your default account.

If you are using anonymous FTP, the username *must* be one of the usernames listed as a valid anonymous name and the password can be anything.

If you are using password authentication, you must specify both your username and password. **THIS IS HIGHLY DISCOURAGED, AS YOU ARE SENDING YOUR PASSWORD IN THE CLEAR ON THE NETWORK.** This is worse than no security; it is a false illusion of security.

Command line options

Informational Options

-help -usage	Prints help.
-version	Prints the version of this program.
-versions	Prints the versions of all modules that this program uses.
-q -quiet	Suppresses all output for successful operation.
-vb -verbose	During the transfer, displays: <ul style="list-style-type: none"> • number of bytes transferred, • performance since the last update (currently every 5 seconds), and • average performance for the whole transfer.
-dbg -debugftp	Debugs FTP connections and prints the entire control channel protocol exchange to STDERR. Very useful for debugging. Please provide this any time you are requesting assistance with a globus-url-copy problem.
-list <url>	This option will display a directory listing for the given url.

Utility Ease of Use Options

-a -ascii	Converts the file to/from ASCII format to/from local file format.
-b -binary	Does not apply any conversion to the files. This option is turned on by default.
-f <i>filename</i>	Reads a list of URL pairs from a filename. Each line should contain: <i>sourceURL destURL</i> Enclose URLs with spaces in double quotes ("). Blank lines and lines beginning with the hash sign (#) will be ignored.
-r -recurse	Copies files in subdirectories.
-notpt -no-third-party-transfers	Turns third-party transfers off (on by default). Site firewall and/or software configuration may prevent a connection between the two servers (a <i>third party transfer</i>). If this is the case, globus-url-copy will "relay" the data. It will do a GET from the source and a PUT to the destination. This obviously causes a performance penalty but will allow you to complete a transfer you otherwise could not do.

Reliability Options

-rst -restart	Restarts failed FTP operations.
-----------------	---------------------------------

- rst-retries <retries> Specifies the maximum number of times to retry the operation before giving up on the transfer.
Use 0 for infinite.
The default value is 5.
- rst-interval <seconds> Specifies the interval in seconds to wait after a failure before retrying the transfer.
Use 0 for an exponential backoff.
The default value is 0.
- rst-timeout <seconds> Specifies the maximum time after a failure to keep retrying.
Use 0 for no timeout.
The default value is 0.

Performance Options

- tcp-bs <size> | -tcp-buffer-size <size> Specifies the size (in bytes) of the TCP buffer to be used by the underlying ftp data channels.

 **Important**

This is critical to good performance over the WAN.

[How do I pick a value?](#)

- p <parallelism> | -parallel <parallelism> Specifies the number of parallel data connections that should be used.

 **Note**

This is one of the most commonly used options.

[How do I pick a value?](#)

- bs <block size> | -block-size <block size> Specifies the size (in bytes) of the buffer to be used by the underlying transfer methods.

- pp **(New starting with GT 4.1.3)** Allows pipelining. GridFTP is a command response protocol. A client sends one command and then waits for a "Finished response" before sending another. Adding this overhead on a per-file basis for a large data set partitioned into many small files makes the performance suffer. Pipelining allows the client to have many outstanding, unacknowledged transfer commands at once. Instead of being forced to wait for the "Finished response" message, the client is free to send transfer commands at any time.

- mc *filename source_url* **(New starting with GT 4.2.1)** Transfers a single file to many destinations. File-name is a line-separated list of destination urls. For more information on this option, click [here](#).

Multicasting must be [enabled for use](#) on the server side.

Security Related Options

-s <subject> | -subject <subject> Specifies a subject to match with both the source and destination servers.



Note

Used when the server does not have access to the host certificate (usually when you are running the server as a user). See [the section called “If you run a GridFTP server by hand...”](#).

-ss <subject> | -source-subject <subject> Specifies a subject to match with the source server.



Note

Used when the server does not have access to the host certificate (usually when you are running the server as a user). See [the section called “If you run a GridFTP server by hand...”](#).

-ds <subject> | -dest-subject <subject> Specifies a subject to match with the destination server.



Note

Used when the server does not have access to the host certificate (usually when you are running the server as a user). See [the section called “If you run a GridFTP server by hand...”](#).

-nodcau | -no-data-channel-authentication Turns off data channel authentication for FTP transfers (the default is to authenticate the data channel).



Warning

We *do not* recommend this option, as it is a security risk.

-dcsafe | -data-channel-safe Sets data channel protection mode to SAFE.

Otherwise known as *integrity* or *checksumming*.

Guarantees that the data channel has not been altered, though a malicious party may have observed the data.



Warning

Rarely used as there is a substantial performance penalty.

-dcpriv | -data-channel-private Sets data channel protection mode to PRIVATE.

The data channel is encrypted and checksummed.

Guarantees that the data channel has not been altered and, if observed, it won't be understandable.



Warning

VERY rarely used due to the VERY substantial performance penalty.

Default globus-url-copy usage

A **globus-url-copy** invocation using the **gsift** protocol with no options (i.e., using all the defaults) will perform a transfer with the following characteristics:

- binary
- stream mode (which implies no parallelism)
- host default TCP buffer size
- encrypted and checksummed control channel
- an authenticated data channel

MODES in GridFTP

GridFTP (as well as normal FTP) defines multiple wire protocols, or MODES, for the data channel.

Most normal FTP servers only implement *stream mode* (MODE S), i.e. the bytes flow in order over a single TCP connection. GridFTP defaults to this mode so that it is compatible with normal FTP servers.

However, GridFTP has another MODE, called Extended Block Mode, or *MODE E*. This mode sends the data over the data channel in blocks. Each block consists of 8 bits of flags, a 64 bit integer indicating the offset from the start of the transfer, and a 64 bit integer indicating the length of the block in bytes, followed by a payload of length bytes. Because the offset and length are provided, out of order arrival is acceptable, i.e. the 10th block could arrive before the 9th because you know explicitly where it belongs. This allows us to use multiple TCP channels. If you use the `-p | -parallelism` option, **globus-url-copy** automatically puts the servers into MODE E.



Note

Putting `-p 1` is not the same as no `-p` at all. Both will use a single stream, but the default will use stream mode and `-p 1` will use MODE E.

If you run a GridFTP server by hand...

If you run a GridFTP server by hand, you will need to explicitly specify the subject name to expect. The subject option provides **globus-url-copy** with a way to validate the remote servers with which it is communicating. Not only must the server trust **globus-url-copy**, but **globus-url-copy** must trust that it is talking to the correct server. The validation is done by comparing host DNs or subjects.

If the GridFTP server in question is running under a host certificate then the client assumes a subject name based on the server's canonical DNS name. However, if it was started under a user certificate, as is the case when a server is started by hand, then the expected subject name must be explicitly stated. This is done with the `-ss`, `-sd`, and `-s` options.

`-ss` Sets the `sourceURL` subject.

`-ds` Sets the `destURL` subject.

- s If you use this option alone, it will set both urls to be the same. You can see an example of this usage under the [Troubleshooting](#) section.



Note

This is an *unusual* use of the client. Most times you need to specify both URLs.

How do I choose a value?

How do I choose a value for the TCP buffer size (-tcp-bs) option?

The value you should pick for the TCP buffer size (-tcp-bs) depends on how fast you want to go (your bandwidth) and how far you are moving the data (as measured by the Round Trip Time (RTT) or the time it takes a packet to get to the destination and back).

To calculate the value for -tcp-bs, use the following formula (this assumes that Mega means 1000^2 rather than 1024^2, which is typical for bandwidth):

$$-tcp-bs = \text{bandwidth in Megabits per second (Mbs)} * \text{RTT in milliseconds (ms)} * 1000 / 8$$

As an example, if you are using fast ethernet (100 Mbs) and the RTT was 50 ms it would be:

$$-tcp-bs = 100 * 50 * 1000 / 8 = 625,000 \text{ bytes.}$$

So, how do you come up with values for bandwidth and RTT? To determine RTT, use either ping or traceroute. They both list RTT values.



Note

You must be on one end of the transfer and ping the other end. This means that if you are doing a third party transfer you have to run the ping or traceroute between the two server hosts, not from your client.

The bandwidth is a little trickier. Any point in the network can be the bottleneck, so you either need to talk with your network engineers to find out what the bottleneck link is or just assume that your host is the bottleneck and use the speed of your network interface card (NIC).



Note

The value you pick for -tcp-bs limits the top speed you can achieve. You will NOT get bandwidth any higher than what you used in the calculation (assuming the RTT is actually what you specified; it varies a little with network conditions). So, if for some reason you want to limit the bandwidth you get, you can do that by judicious choice of -tcp-bs values.

So where does this formula come from? Because it uses the bandwidth and the RTT (also known as the latency or delay) it is called the *bandwidth delay product*. The very simple explanation is this: TCP is a reliable protocol. It must save a copy of everything it sends out over the network until the other end acknowledges that it has been received.

As a simple example, if I can put one byte per second onto the network, and it takes 10 seconds for that byte to get there, and 10 seconds for the acknowledgment to get back (RTT = 20 seconds), then I would need at least 20 bytes of storage. Then, hopefully, by the time I am ready to send byte 21, I have received an acknowledgement for byte 1 and I can free that space in my buffer. If you want a more detailed explanation, try the following links on TCP tuning:

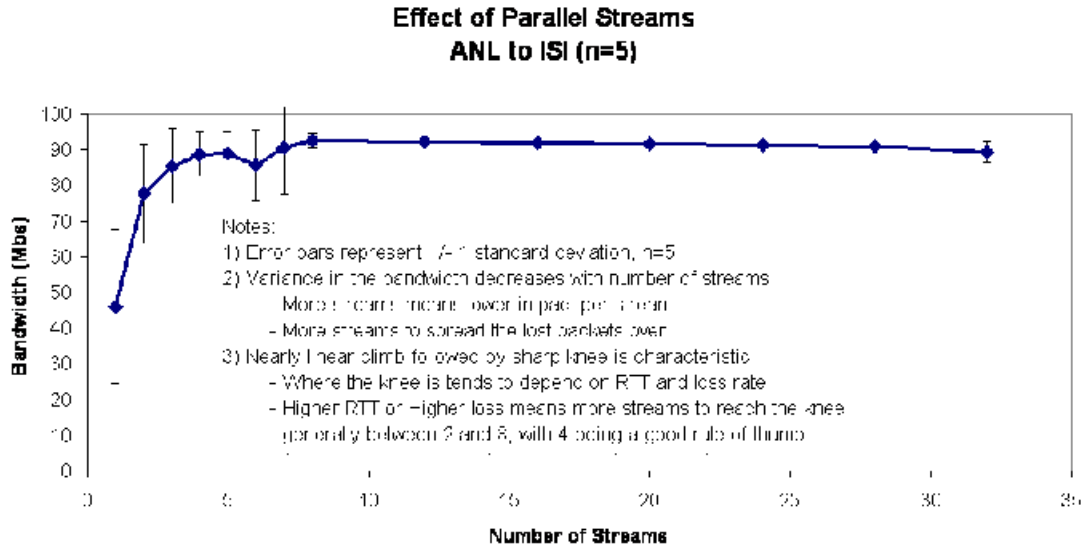
- http://www.psc.edu/networking/perf_tune.html

- <http://www.didc.lbl.gov/TCP-tuning/>
- <http://www.ncne.nlanr.net/research/tcp/>

How do I choose a value for the parallelism (-p) option?

For most instances, using 4 streams is a very good rule of thumb. Unfortunately, there is not a good formula for picking an exact answer. The shape of the graph shown here is very characteristic.

Figure 1. Effect of Parallel Streams in GridFTP



You get a strong, nearly linear, increase in bandwidth, then a sharp knee, after which additional streams have very little impact. Where this knee is depends on many things, but it is generally between 2 and 10 streams. Higher bandwidth, longer round trip times, and more congestion in the network (which you usually can only guess at based on how applications are behaving) will move the knee higher (more streams needed).

In practice, between 4 and 8 streams are usually sufficient. If things look really bad, try 16 and see how much difference that makes over 8. However, anything above 16, other than for academic interest, is basically wasting resources.

Limitations

There are no limitations for **globus-url-copy** in GT 4.2.1.

Interactive clients for GridFTP

The Globus Project does *not* provide an interactive client for GridFTP. Any normal FTP client will work with a GridFTP server, but it cannot take advantage of the advanced features of GridFTP. The interactive clients listed below take advantage of the advanced features of GridFTP.

There is no endorsement implied by their presence here. We make no assertion as to the quality or appropriateness of these tools, we simply provide this for your convenience. We will *not* answer questions, accept bugs, or in any way shape or form be responsible for these tools, although they should have mechanisms of their own for such things.

UberFTP was developed at the NCSA under the auspices of NMI and TeraGrid:

- NCSA Uerftp only download: <http://dims.ncsa.uiuc.edu/set/uberftp/download.html>
- UberFTP User's Guide: <http://dims.ncsa.uiuc.edu/set/uberftp/userdoc.html>

Name

globus-gridftp-server -- Configures the GridFTP Server

globus-gridftp-server

Tool description

globus-gridftp-server configures the GridFTP server using a config file and/or commandline options.



Note

Command line options and configuration file options may both be used, but the command line *overrides* the config file.

The configuration file for the GridFTP *server* is read from the following locations, in the given order. Only the first file found will be loaded:

- Path specified with the `-c <configfile>` command line option.
- `$GLOBUS_LOCATION/etc/gridftp.conf`
- `/etc/grid-security/gridftp.conf`

Options are one per line, with the format:

```
<option> <value>
```

If the value contains spaces, they should be enclosed in double-quotes ("). Flags or boolean options should only have a value of 0 or 1. Blank lines and lines beginning with # are ignored.

For example:

```
port 5000
allow_anonymous 1
anonymous_user bob
banner "Welcome!"
```

Developer notes

The Globus implementation of the GridFTP *server* draws on:

- three IETF RFCs:
 - RFC 959
 - RFC 2228
 - RFC 2389
- an IETF Draft: MLST-16
- the GridFTP protocol specification, which is Global Grid Forum (GGF) Standard GFD.020.

The command line tools and the *client* library completely hide the details of the protocol from the user and the developer. Unless you choose to use the control library, it is not necessary to have a detailed knowledge of the protocol.

Command syntax

The basic syntax for **globus-gridftp-server** is:

```
globus-gridftp-server [optional command line switches]
```

To use **globus-gridftp-server** with a config file, make sure to use the `-c <configfile>` option.

Command line options

The table below lists config file options, associated command line options (if available) and descriptions.



Note

Any boolean option can be negated on the command line by preceding the specified option with '-no-' or '-n'.
example: -no-cas or -nf.

Informational Options

help <0 1>, -h, -help	Show usage information and exit. Default value: FALSE
version <0 1> , -v, -version	Show version information for the server and exit. Default value: FALSE
versions <0 1>, -v, -versions	Show version information for all loaded globus libraries and exit. Default value: FALSE

Modes of Operation

inetd <0 1>, -i, -inetd	Run under an inetd service. Default value: FALSE
daemon <0 1>, -s, -daemon	Run as a daemon. All connections will fork off a new process and setuid if allowed. See Section 4, “Running in daemon mode” for more information. Default value: TRUE
detach <0 1>, -S, -detach	Run as a background daemon detached from any controlling terminals. See Section 4, “Running in daemon mode” for more information. Default value: FALSE
exec <string> , -exec <string>	For statically compiled or non-GLOBUS_LOCATION standard binary locations, specify the full path of the server binary here. Only needed when run in daemon mode . Default value: not set

`chdir <0|1>, -chdir` Change directory when the server starts. This will change directory to the dir specified by the `chdir_to` option.

Default value: TRUE

`chdir_to <string>, -chdir-to <string>` Directory to `chdir` to after starting. Will use `/` if not set.

Default value: not set

`fork <0|1>, -f, -fork` Server will fork for each new connection. Disabling this option is only recommended when debugging. Note that non-forked servers running as 'root' will only accept a single connection and then exit.

Default value: TRUE

`single <0|1>, -1, -single` Exit after a single connection.

Default value: FALSE

Authentication, Authorization, and Security Options

`auth_level <number>, -auth-level <number>`

- 0 = Disables all authorization checks.
- 1 = Authorize identity only.
- 2 = Authorize all file/resource accesses.

If not set, the GridFTP Server uses level 2 for front ends and level 1 for data nodes.

Default value: not set

`allow_from <string>, -allow-from <string>` Only allow connections from these source IP addresses. Specify a comma-separated list of IP address fragments. A match is any IP address that starts with the specified fragment. Example: '192.168.1.' will match and allow a connection from 192.168.1.45. Note that if this option is used, any address not specifically allowed will be denied.

Default value: not set

`deny_from <string>, -deny-from <string>` Deny connections from these source IP addresses. Specify a comma-separated list of IP address fragments. A match is any IP address that starts with the specified fragment. Example: '192.168.2.' will match and deny a connection from 192.168.2.45.

Default value: not set

`cas <0|1>, -cas` Enable Community Authorization Service (CAS) authorization. For complete instructions on setting up a GridFTP server to use CAS, click [here](#).

Default value: TRUE

`secure_ipc <0|1>, -si, -secure-ipc` Use GSI security on the IPC channel.

Default value: TRUE

secure_ipc <0 1>, -si, -secure-ipc	Use GSI security on the IPC channel. Default value: TRUE
ipc_auth_mode <string>, -ia <string>, -ipc-auth- mode <string>	Set GSI authorization mode for the IPC connection. Options are one of the following: <ul style="list-style-type: none"> • none • host • self • subject:[subject] Default value: host
allow_anonym- ous <0 1>, -aa, -allow- anonymous	Allow cleartext anonymous access. If server is running as root, anonymous_user must also be set. Disables IPC security. Default value: FALSE
anonym- ous_names_al- lowed <string>, -an- onymous- names-allowed <string>	Comma-separated list of names to treat as anonymous users when allowing anonymous access. If not set, the default names of 'anonymous' and 'ftp' will be allowed. Use '*' to allow any user-name. Default value: not set
anonym- ous_user <string>, -an- onymous-user <string>	User to setuid to for an anonymous connection. Only applies when running as root. Default value: not set
anonym- ous_group <string>, -an- onymous-group <string>	Group to setgid to for an anonymous connection. If not set, the default group of anonymous_user will be used. Default value: not set
pw_file <string>, -password- file <string>	Enable cleartext access and authenticate users against this /etc/passwd formatted file. Default value: not set
connec- tions_max <number>, -connections- max <number>	Maximum concurrent connections allowed. Only applies when running in <u>daemon mode</u> . Unlimited if not set. Default value: not set
connec- tions_dis- abled <0 1>, Default value: FALSE	Disable all new connections. Does not affect ongoing connections. This must be set in the configuration file and then a SIGHUP issued to the server in order to reload the configuration.

-connections-
disabled

Logging Options

log_level Log level. A comma-separated list of levels from the following:

<string>, -d
<string>,
-log-level
<string>

- ERROR
- WARN
- INFO
- DUMP
- ALL

For example:

```
globus-gridftp-server -d error,warn,info
```

You may also specify a numeric level of 1-255.

Default value: ERROR

log_module Indicates the globus_logging module that will be loaded. If not set, the default stdio module will be used and the logfile options (see next option) will apply.

<string>,
-log-module
<string>

Built-in modules are stdio and syslog. Log module options may be set by specifying module:opt1=val1:opt2=val2. Available options for the built-in modules are:

- interval - Indicates buffer flush interval. Default is 5 seconds. A 0 second flush interval will disable periodic flushing, and the buffer will only flush when it is full.
- buffer - Indicates buffer size. Default is 64k. A value of 0k will disable buffering and all messages will be written immediately.

Example:

```
-log-module stdio:buffer=4096:interval=10
```

Default value: not set

log_single Indicates the path of a single file to which you want to log all activity. If neither this option nor log_unique is set, logs will be written to stderr, unless the execution mode is detached, or inetd, in which case logging will be disabled.

<string>, -l
<string>,
-logfile
<string>

Default value: not set

log_unique Partial path to which gridftp.(pid).log will be appended to construct the log filename. Example:

<string>, -L
<string>, -l
gdir <string>

```
-L /var/log/gridftp/
```

will create a separate log (/var/log/gridftp/gridftp.xxxx.log) for each process (which is normally each new *client* session). If neither this option nor log_single is set, logs will be written to stderr, unless the execution mode is detached, or inetd, in which case logging will be disabled.

	Default value: not set
log_transfer <string>, -Z <string>, -log-transfer <string>	Log NetLogger-style info for each transfer into this file. Default value: not set Example: DATE=20050520163008.306532 HOST=localhost PROG=globus-gridftp-server NL.EVNT=FTP_INFO START=20050520163008.305913 USER=ftp FILE=/etc/group BUF- FER=0 BLOCK=262144 NBYTES=542 VOLUME=/ STREAMS=1 STRIPES=1 DEST=[127.0.0.1] TYPE=RETR CODE=226 Time format is YYYYMMDDHHMMSS.UUUUUU (microsecs). <ul style="list-style-type: none"> • DATE: time the transfer completed. • START: time the transfer started. • HOST: hostname of the server. • USER: username on the host that transferred the file. • BUFFER: tcp buffer size (if 0 system defaults were used). • BLOCK: the size of the data block read from the disk and posted to the network. • NBYTES: the total number of bytes transferred. • VOLUME: the disk partition where the transfer file is stored. • STREAMS: the number of parallel TCP streams used in the transfer. • STRIPES: the number of stripes used on this end of the transfer. • DEST: the destination host. • TYPE: the transfer type, RETR is a send and STOR is a receive (ftp 959 commands). • CODE: the FTP rfc959 completion code of the transfer. 226 indicates success, 5xx or 4xx are failure codes.
log_filemode <string>, -log-filemode <string>	File access permissions of log files. Should be an octal number such as 0644 (the leading 0 is required). Default value: not set
disable_us- age_stats <0 1>, -dis- able-usage- stats	Disable transmission of per-transfer usage statistics. See the Usage Statistics ¹ section in the online documentation for more information. Default value: FALSE
us- age_stats_tar- get <string>, -usage-stats-	Comma-separated list of contact strings for usage statistics listeners. The format of <string> is host:port. Default value: usage-stats.globus.org:4810

¹ ../../Usage_Stats.html

target **Example:**
 <string>
 -usage-stats-target usage-stats.globus.org:4810,usage-stats.uc.teragrid.org

In this example, the usage statistics will be transmitted to the default Globus target (usage-stats.globus.org:4810) and another target (usage-stats.uc.teragrid.org:5920).

Single and Striped Remote Data Node Options

remote_nodes Comma-separated list of remote node contact strings. See [Remote data-nodes and striped operations](#) and [Separation of processes for higher security](#) for examples of using this option.
 <string>, -r
 <string>, -remote-nodes
 <string>
 Default value: not set

data_node This server is a back end data node. See [Separation of processes for higher security](#) for an example of using this option.
 <0|1>, -dn,
 -data-node
 Default value: FALSE

stripe_blocksize Size in bytes of sequential data that each stripe will transfer.
 <number>,
 -sbs <number>
 , -stripe-blocksize
 <number>
 Default value: 1048576

stripe_layout Stripe layout. 1 = Partitioned, 2 = Blocked.
 <number>, -sl
 <number>,
 -stripe-layout
 <number>
 Default value: 2

stripe_blocksize_locked Do not allow client to override stripe blocksize with the **OPTS RETR** command.
 <0|1>,
 -stripe-blocksize-locked;
 Default value: FALSE

stripe_layout_locked Do not allow client to override stripe layout with the **OPTS RETR** command.
 <0|1>,
 -stripe-layout-locked
 Default value: FALSE

Disk Options

blocksize Size in bytes of data blocks to read from disk before posting to the network.
 <number>, -bs
 <number>,
 -blocksize
 <number>
 Default value: 262144

`sync_writes` <0|1>, `-sync-writes` Flush disk writes before sending a restart marker. This attempts to ensure that the range specified in the restart marker has actually been committed to disk. This option will probably impact performance and may result in different behavior on different storage systems. See the man page for `sync()` for more information.

Default value: FALSE

Network Options

`port` <number>, `-p` <number>, `-port` <number> Port on which a front end will listen for client control channel connections or on which a data node will listen for connections from a front end. If not set, a random port will be chosen and printed via the logging mechanism. See [Remote data-nodes and striped operations](#) and [Separation of processes for higher security](#) for examples of using this option.

Default value: not set

`control_interface` <string>, `-control-interface` <string> Hostname or IP address of the interface to listen for control connections on. If not set, will listen on all interfaces.

Default value: not set

`data_interface` <string>, `-data-interface` <string> Hostname or IP address of the interface to use for data connections. If not set will use the current control interface.

Default value: not set

`ipc_interface` <string>, `-ipc-interface` <string> Hostname or IP address of the interface to use for IPC connections. If not set, will listen on all interfaces.

Default value: not set

`hostname` <string>, `-hostname` <string> Effectively sets the above `control_interface`, `data_interface` and `ipc_interface` options.

Default value: not set

`ipc_port` <number>, `-ipc-port` <number> Port on which the front end will listen for data node connections.

Default value: not set

Timeouts

`control_preauth_timeout` <number>, `-control-preauth-timeout` <number> Time in seconds to allow a client to remain connected to the control channel without activity before authenticating.

Default value: 30

`control_idle_timeout` <number>;, `-control-` Time in seconds to allow a client to remain connected to the control channel without activity.

Default value: 600

idle-timeout
<number>

ipc_idle_timeout Idle time in seconds before an unused IPC connection will close.
<number> ,
-ipc-idle- Default value: 600
timeout <num-
ber>

ipc_con- Time in seconds before cancelling an attempted IPC connection.
nect_timeout
<number> , Default value: 60
-ipc-connect-
timeout <num-
ber>

User Messages

banner Message that is displayed to the client before authentication.
<string> ,
-banner Default value: not set
<string>

banner_file Read banner message from this file.
<string> ,
-banner-file Default value: not set
<string>

banner_terse When this is set, the minimum allowed banner message will be displayed to unauthenticated
<0|1> , -ban- clients.
ner-terse
Default value: FALSE

login_msg Message that is displayed to the client after authentication.
<string> , -lo-
gin-msg Default value: not set
<string>

lo- Read login message from this file.
gin_msg_file
<string> , -lo- Default value: not set
gin-msg-file
<string>

Module Options

load_dsi_mod- Load this Data Storage Interface module. File and remote modules are defined by the server. If
ule <string> , not set, the file module is loaded, unless the remote option is specified, in which case the remote
-dsi <string> module is loaded. An additional configuration string can be passed to the DSI using the format
[module name]:[configuration string]. The format of the configuration string is
defined by the DSI being loaded.

Default value: not set

`allowed_modules <string>` Comma-separated list of ERET/ESTO modules to allow and, optionally, specify an alias for.
Example:
`, -allowed-modules <string>` `-allowed-modules module1,alias2:module2,module3`
(module2 will be loaded when a client asks for alias2).
Default value: not set

Other Options

`configfile <string>, -c <string>` Path to configuration file that should be loaded. Otherwise will attempt to load `$GLOBUS_LOCATION/etc/gridftp.conf` and `/etc/grid-security/gridftp.conf`.
Default value: not set

`use_home_dirs <0|1>, -use-home-dirs` Set the startup directory to the authenticated user's home dir.
Default value: TRUE

`debug <0|1>, -debug` Set options that make the server easier to debug. Forces `no-fork`, `no-chdir`, and allows core dumps on bad signals instead of exiting cleanly. Not recommended for production servers. Note that non-forked servers running as root will only accept a single connection and then exit.
Default value: FALSE

Limitations

For transfers using parallel data transport streams and for transfers using multiple computers at each end, the direction of the connection on the data channels must go from the sending to the receiving side. For more information about this limitations see <http://www.ogf.org/documents/GFD.20.pdf>.

Globus GridFTP server does not run on windows

Chapter 6. Graphical User Interface

Globus does not provide any interactive client for GridFTP, either GUI or text based. However, NCSA, as part of their TeraGrid activity, produces a text based interactive client called UberFTP, which you may want to check out. See [the section called “Interactive clients for GridFTP”](#) for more information.

Chapter 7. Configuring GridFTP

1. GridFTP server configuration overview

The configuration interface for GridFTP is the admin tool, [globus-gridftp-server](#), which can be used with a configuration file and/or run-time options.



Note

Command line options and configuration file options may both be used, but the command line *overrides* the config file.

The configuration file for the GridFTP *server* is read from the following locations, in the given order. Only the first file found will be loaded:

- Path specified with the `-c <configfile>` command line option.
- `$GLOBUS_LOCATION/etc/gridftp.conf`
- `/etc/grid-security/gridftp.conf`

Options are one per line, with the format:

```
<option> <value>
```

If the value contains spaces, they should be enclosed in double-quotes ("). Flags or boolean options should only have a value of 0 or 1. Blank lines and lines beginning with # are ignored.

For example:

```
port 5000
allow_anonymous 1
anonymous_user bob
banner "Welcome!"
```

For complete command documentation including all options, see [globus-gridftp-server\(1\)](#).

This page includes information about general configuration of the GridFTP server. Security options are discussed [here](#), and more advanced configuration is described [here](#).

2. Types of configurations

The following describes different GridFTP configurations of the front end (control channel) and back end (data channels).

1. **Typical configuration:** this is the default where the data channel and control channel are separate socket connections within the same process. The client sends a command and waits to finish before issuing the next command. This is good for a single host, traditional-type user. If you have a single host and you want an ultra-reliable and light weight file transfer service, this is a good choice. Also good for testing purposes.
2. **Separate processes (or split process):** control channel and data channel are on different ports - with front end run as a non-privileged user (typically the `globus` user) with very limited access to the machine and the back end is run as root, but configured to only allow connections from the front end from a local machine. This means

an external user is never connected to a root running process and thus minimizes the impact of an exploit. This does, however, require that a copy of the [host cert and host key](#) be owned by the non-privileged user. If you use this configuration, the non-privileged user should not have write permission to executables, configuration files, etc. Provides greater security and also allows for proxying and load balancing. Many backend data movers can be behind a single point of client contact. Each client is assigned a different backend in a round robin fashion. For more information about this configuration, see [Section 2, “Separation of Processes \(Split Process\)”](#).

3. **Striped servers:** single control channel (front end), multiple data channels (back end) This is recommended for improved performance of large (1GB+) file transfers. Can also be useful if you want to use full data encryption and need to tether together many hosts to handle the processing load. For more information about this configuration, see [Section 1, “Remote data-nodes and striped operation”](#).

Note

Furthermore, #2 and #3 can be combined. You can have an 8 node cluster that only uses 2 nodes at a time in a striped server configuration and load balances across the rest of the nodes.

3. globus-gridftp-server quickstart

The following is a quick guide to running the server and using the client:

Look through the list of options for globus-gridftp-server:

```
globus-gridftp-server --help
```

Start the server in anonymous mode (discussed more fully [here](#)):

```
globus-gridftp-server -control-interface 127.0.0.1 -aa -p 5000
```

where:

- control-interface is the hostname or IP address of the interface to listen for control connections on . This option is only needed here as a rudimentary means of security for this simple example.
- aa enables anonymous mode
- p indicates on which port the server listens.

Run a two party transfer with client:

```
globus-url-copy -v file:///etc/group ftp://localhost:5000/tmp/group
```

Run 3rd party transfer:

```
globus-url-copy -v ftp://localhost:port/etc/group ftp://localhost:port/tmp/group2
```

Experiment with -dbg, and -vb options for debugging and checking the performance of your setup:

```
globus-url-copy -dbg file:///etc/group ftp://localhost:5000/tmp/group
```

```
globus-url-copy -vb file:///dev/zero ftp://localhost:5000/dev/null
```

where:

- dbg A useful option when something is not working. It results in a GridFTP control channel protocol dump (along with other useful information) to stderr. If you understand the GridFTP protocol, or you have ambition to

understand it, this can be a very useful tool to discover various problems in your setup such as overloaded servers and firewalls. When submitting a bug report or asking a question on the support email lists one should always send along the `-dbg` output.

`-vb` Provides a type of progress bar of the user to observe the rate at which their transfer is progressing.

`Ctrl-c` - Kill the server.

Note

There are many possible options and configurations with **globus-gridftp-server**. For some guidelines on setting it up for your situation, see [Chapter 5, Key Admin Settings and Tuning Recommendations](#).

4. Running in daemon mode

The server should generally be run as root in daemon mode, although it is possible to run it as a user (see below). When run as root you will need to have a [host certificate](#).

Run the server:

```
globus-gridftp-server < -s | -S > <args>
```

where:

- `-s` Runs in the foreground (this is the default mode).
- `-S` Detaches from the terminal and runs in the background.

The following additional steps may be required when running as a user other than root (for more details, review [Basic Security Configuration](#)):

- Create a `~/ .gridmap` file, containing the DNs of any clients you wish to allow, mapped to the current username.
- Create a proxy with **grid-proxy-init**.

5. Running under inetd or xinetd

Note

We also feature a user-configurable, super-server daemon plugin called GFork. Click [here](#) for more information.

5.1. Set up xinetd/inetd config file

Note

The service name used (`gsiftp` in this case) should be defined in `/etc/services` with the desired port.

Here is a sample GridFTP server xinetd config entry in `/etc/xinetd.conf`:

```
service gsiftp
{
instances          = 100
socket_type        = stream
```

```
wait                = no
user                = root
env                 += GLOBUS_LOCATION=(globus_location)
env                 += LD_LIBRARY_PATH=(globus_location)/lib
server              = (globus_location)/sbin/globus-gridftp-server
server_args         = -i
log_on_success      += DURATION
nice                = 10
disable             = no
}
```

Here is a sample gridftp server inetd config entry in `/etc/inetd.conf` (read as a single line):

```
gsiftp stream tcp nowait root /usr/bin/env env \
  GLOBUS_LOCATION=(globus_location) \
  LD_LIBRARY_PATH=(globus_location)/lib \
  (globus_location)/sbin/globus-gridftp-server -i
```



Note

On Mac OS X, you must set `DYLD_LIBRARY_PATH` instead of `LD_LIBRARY_PATH` in the above examples.

On IRIX, you may need to set either `LD_LIBRARYN32_PATH` or `LD_LIBRARY64_PATH`.



Note

You should NOT include `USERID` in the log lines. See [Section 5, “High latency for GridFTP server connections”](#) for more information.

5.2. globus-gridftp-server -i

Use the `-i` commandline option with `globus-gridftp-server`:

```
globus-gridftp-server -i
```

Chapter 8. Environment variable interface

1. Environment variables for GridFTP

The GridFTP *server* or *client* libraries do not read any environment variable directly, but the security and networking related variables described below may be useful.

- [Non-WS \(General\) Authentication & Authorization Environment Variables.](#)
- [XIO Network Driver Environment Variables.](#)

Chapter 9. Debugging

You can find information on sys admin logs in [Chapter 7, Debugging](#).

Chapter 10. Troubleshooting

If you are having problems using the GridFTP [server](#), try the steps listed below. If you have an error, try checking the server logs if you have access to them. By default, the server logs to stderr, unless it is running from inetd, or its execution mode is detached, in which case logging is disabled by default.

The command line options `-d`, `-log-level`, `-L` and `-logdir` can affect where logs will be written, as can the configuration file options `log_single` and `log_unique`. See the [globus-gridftp-server\(1\)](#) for more information on these and other configuration options.

You should also be familiar with the [security considerations](#).

For a list of common errors in GT, see [Error Codes](#).

1. Error Codes in GridFTP

Table 10.1. GridFTP Errors

Error Code	Definition	Possible Solutions
<pre>globus_ftp_client: the server responded with an error 530 530-glo- bus_xio: Authentication Error 530-OpenSSL Error: s3_srvr.c:2525: in lib- rary: SSL routines, function SSL3_GET_CLI- ENT_CERTIFICATE: no cer- tificate returned 530- globus_gsi_callback_mod- ule: Could not verify credential 530-glo- bus_gsi_callback_module: Can't get the local trusted CA certificate: Untrusted self-signed certificate in chain with hash d1b603c3 530 End.</pre>	<p>This error message indicates that the GridFTP server doesn't trust the certificate authority (CA) that issued your certificate.</p>	<p>You need to ask the GridFTP server administrator to install your CA certificate chain in the GridFTP server's trusted certificates directory.</p>
<pre>globus_ftp_control: gss_init_sec_context failed OpenSSL Error: s3_clnt.c:951: in lib- rary: SSL routines, function SSL3_GET_SERV- ER_CERTIFICATE: certific- ate verify failed glo- bus_gsi_callback_module: Could not verify creden- tial globus_gsi_call- back_module: Can't get the local trusted CA certificate: Untrusted self-signed certificate in chain with hash d1b603c3</pre>	<p>This error message indicates that your local system doesn't trust the certificate authority (CA) that issued the certificate on the resource you are connecting to.</p>	<p>You need to ask the resource administrator which CA issued their certificate and install the CA certificate in the local trusted certificates directory.</p>

2. Establish control channel connection

Verify that you can establish a control channel connection and that the server has started successfully by telnetting to the port on which the server is running:

```
% telnet localhost 2811
      Trying 127.0.0.1...
      Connected to localhost.
```

```
Escape character is '^]'.
220 GridFTP Server mldev.mcs.anl.gov 2.0 (gcc32dbg, 1113865414-1) ready.
```

If you see anything other than a 220 banner such as the one above, the server has not started correctly.

Verify that there are no configuration files being unexpectedly loaded from `/etc/grid-security/gridftp.conf` or `$GLOBUS_LOCATION/etc/gridftp.conf`. If those files exist, and you did not intend for them to be used, rename them to `.save`, or specify `-c none` on the command line and try again.

If you can log into the machine where the server is, try running the server from the command line with only the `-s` option:

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -s
```

The server will print the port it is listening on:

```
Server listening at gridftp.mcs.anl.gov:57764
```

Now try and telnet to that port. If you still do not get the banner listed above, something is preventing the socket connection. Check firewalls, `tcp-wrapper`, etc.

If you now get a correct banner, add `-p 2811` (you will have to disable `(x)inetd` on port 2811 if you are using them or you will get port already in use):

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -s -p 2811
```

Now telnet to port 2811. If this does not work, something is blocking port 2811. Check firewalls, `tcp-wrapper`, etc.

If this works correctly then re-enable your normal server, but remove all options but `-i`, `-s`, or `-S`.

Now telnet to port 2811. If this does not work, something is wrong with your service configuration. Check `/etc/services` and `(x)inetd` config, have `(x)inetd` restarted, etc.

If this works, begin adding options back one at a time, verifying that you can telnet to the server after each option is added. Continue this till you find the problem or get all the options you want.

At this point, you can establish a control connection. Now try running `globus-url-copy`.

3. Try running `globus-url-copy`

Once you've verified that you can establish a control connection, try to make a transfer using `globus-url-copy`.

If you are doing a *client/server* transfer (one of your URLs has `file:` in it) then try:

```
globus-url-copy -vb -dbg gsiftp://host.server.running.on/dev/zero file:///dev/null
```

This will run until you control-c the transfer. If that works, reverse the direction:

```
globus-url-copy -vb -dbg file:///dev/zero gsiftp://host.server.running.on/dev/null
```

Again, this will run until you control-c the transfer.

If you are doing a *third party transfer*, run this command:

```
globus-url-copy -vb -dbg gsiftp://host.server1.on/dev/zero gsiftp://host.server2.on/dev/nu
```

Again, this will run until you control-c the transfer.

If the above transfers work, try your transfer again. If it fails, you likely have some sort of file permissions problem, typo in a file name, etc.

4. If your server starts...

If the server has started correctly, and your problem is with a security failure or gridmap lookup failure, verify that you have security configured properly [here](#).

If the server is running and your client successfully authenticates but has a problem at some other time during the session, please ask for help on gt-user@globus.org¹. When you send mail or submit bugs, please always include as much of the following information as possible:

- Specs on all hosts involved (OS, processor, RAM, etc).
- `globus-url-copy -version`
- `globus-url-copy -versions`
- Output from the telnet test above.
- The actual command line you ran with `-dbg` added. Don't worry if the output gets long.
- Check that you are getting a FQDN and `/etc/hosts` that is sane.
- The server configuration and setup (`/etc/services` entries, `(x)inetd` configs, etc.).
- Any relevant lines from the server logs (not the entire log please).

5. High latency for GridFTP server connections

If you run GridFTP servers via Xinetd and notice high latency for connections and/or transfers, check if `/etc/xinetd.conf` or the `gsiftp` service configuration inside `/etc/xinetd.d` is set to log USERID as follows:

```
log_on_success += USERID
log_on_failure += USERID
```

Such a configuration tells Xinetd to log the remote user using the method defined in RFC 1413, which causes an ident client to attempt to query the machine that the connection is coming from before the service will run. Even when this succeeds, the response can't be trusted, and more often than not it is rejected or simply dropped (which results in the longest delays) by the remote firewall.

Latency can be reduced by making sure Xinetd does *not* log the USERID.

¹ http://dev.globus.org/wiki/Mailing_Lists

Chapter 11. Related Documentation

- [The Globus Striped GridFTP Framework and Server](http://www.globus.org/alliance/publications/papers/gridftp_final.pdf)¹

¹ http://www.globus.org/alliance/publications/papers/gridftp_final.pdf

Appendix A. Developing DSIs for GridFTP

The GridFTP server provides high speed remote access to data stores. There are many different types of data storage systems from standard file systems to arrays of magnetic tape. To allow GridFTP to be a transfer interface to as many data storage systems as possible the Data Storage Interface (DSI) was created.

The DSI presents a modular abstraction layer to a storage system. It consists of several function signatures and a set of semantics.

- When a new DSI is created, a programmer implements the functions to provide the semantics associated with them.
- DSIs can be loaded and switched at runtime.
- When the server requires action from the storage system (be it data, meta-data, directory creation, etc), it passes a request to the loaded DSI module.
- The DSI then services that request and tells the function when it is finished.

This document provides an introduction to the DSI and how to create one.

1. DSI Interface

The set of interface functions that define the DSI can be found in `globus_gridftp_server.h`.

All type definitions starting with `globus_gfs_storage_*` are part of the DSI interface.

2. DSI utility API

An API is provided to the DSI author to assist in implementation. The most interesting parts of this API provide functions that abstract away the details of sending data across the data channel. The DSI author is not expected to know the intimate details of the data channel protocols involved in a GridFTP transfer. Instead this API provides functions for reading and writing data to and from the net.

FIXME - link to api doc

3. Implementation

The following is a brief description of part of the DSI implementation process.

An FTP session is defined from the time a client is authorized to use the server until the time the connection is disconnected (disconnect can happen due to the client sending QUIT, error, or timeout, etc). In the lifetime of the session, the client issues various commands to the FTP server. Some of these commands require access to the storage system, and thus require action by the DSI. Whenever such a command is received, the server calls out to the appropriate DSI interface function requesting that the specific operation be performed.

The server passes a

`globus_gfs_operation_t`

data type as a parameter to all DSI request functions. When the DSI is finished performing that operation, it calls a corresponding

```
globus_gridftp_server_finished_<type>()
```

function, passing it this `globus_gfs_operation_t` structure (and whatever other data is needed for any given operation). This lets the server know that the operation is completed and it can respond to the client appropriately.

As an example we will look at how a simple unix file system DSI would implement the `stat` function.

The DSI's function signature for `stat` is:

```
void
(*globus_gfs_storage_stat_t)(
    globus_gfs_operation_t      op,
    globus_gfs_stat_info_t *    stat_info,
    void *                       user_arg);
```

When it is called, the DSI is expected to:

- determine all information associated with the path: `stat_info->pathname`,
- fill in a `globus_gfs_stat_t` with that information,
- and then call `globus_gridftp_server_finished_stat()` with that structure.

```
static
void
globus_gfs_storage_example_stat(
    globus_gfs_operation_t      op,
    globus_gfs_stat_info_t *    stat_info,
    void *                       user_arg)
{
    globus_gfs_stat_t           stat_out;
    struct stat                 stat_in;

    stat(stat_info->pathname, &stat_in);

    stat_out.mode               = stat_in.st_mode;
    stat_out.nlink              = stat_in.st_nlink;
    stat_out.uid                = stat_in.st_uid;
    stat_out.gid                = stat_in.st_gid;
    stat_out.size               = stat_in.st_size;
    stat_out.mtime              = stat_in.st_mtime;
    stat_out.atime              = stat_in.st_atime;
    stat_out.ctime              = stat_in.st_ctime;
    stat_out.dev                = stat_in.st_dev;
    stat_out.ino                = stat_in.st_ino;

    stat_out.name = strdup(stat_info->pathname);

    globus_gridftp_server_finished_stat(op, GLOBUS_SUCCESS, &stat_out, 1);
}
```

This is obviously a very basic example but it should serve for the purposes of understanding.

4. DSI Bones

Every DSI must register itself with the Globus extensions module properly. This can be a tedious task yet must be done properly. For this reason, we created a distribution that provides a skeleton DSI upon which a developer can build.

The distribution includes a script to generate C stubs for a DSI with all of the proper shared library hooks and names needed to work with the **globus-gridftp-server**. The DSI implementor must fill in the stubbed-out functions with the necessary code specific to their needs.

```
% ./generate-stubs.sh dsi name flavor
```

This command will generate the c source file. "dsi name" is the string that will be associated with the DSI. It must be unique to your Globus installation. To load it into the server use the `-dsi dsi name` option to the server.

```
% make
```

This will compile the DSI and create the dynamically loadable library. To include additional compile dependencies or libraries, open `Makefile` and add them to the appropriate `MACRO` line.

```
% make install
```

This will copy the library to `$GLOBUS_LOCATION/lib`, thereby making it ready for use.

Appendix B. GridFTP Multicast

GridFTP is a well known, extremely fast and efficient protocol for transferring data from one destination to another. Here we present how GridFTP can be used to transfer a single file to many destinations in a multicast/broadcast.

1. Architecture

The purpose of this work is to efficiently transfer a single data set to many locations. Our goal is to use all available network cycles, effectively moving the total amount of data (destinations * file size) at network speeds. Ideally, we would like to transfer to all destinations in the same time it takes to transfer to a single destination.

Distributing the data to many endpoints is not difficult, and has been a feature of **globus-url-copy** (a popular GridFTP client) for some time. It would be quite simple to have the client loop over the destination set and send to each destination in series. Of course, this would be quite slow and the transfer time would scale linearly with the data set.

In our architecture, destination servers are configured so that they can forward packets along to other endpoints. Thus, the GridFTP servers act as both servers receiving data and clients sending data. The server is set up as a tree such that the first destination writes the data to disk and then forwards it on to N more hosts (by default N is 2). This process is repeated until all destinations have received the data.

For further explanation, the architecture can be thought of as a directed graph. Each destination is a vertex with N edges connecting it to N other vertices. A spanning tree is formed connecting all vertices. The degree of any one vertex is a client -side configuration option.

The following image illustrates this:

Figure B.1. GridFTP Spanning Tree



In the image, data blocks are purple and are sent first from the client to a root destination. The root destination then forwards it on to two more servers.

2. Globus XIO

The figure in the previous section shows 4 colored boxes. Orange represents client logic, blue is server logic, and as stated above, purple is for data block. The final box type is yellow and it is used to show globus XIO drivers.

More information on Globus XIO can be found [here](#). For our purposes we can think of each XIO driver as a modular protocol interpreter that can be plugged in to an IO stack without involving the application using it. In this way, we can add functionality to an existing application without disturbing its tested code base.

Because the **globus-gridftp-server** uses Globus XIO for all of its IO, we are able to forward data at the block level. We achieve this by allowing the client to add a new XIO driver, the `gridftp_multicast` driver, to the GridFTP server's disk stack. Because of the modular driver abstraction that Globus XIO provides as the GridFTP server writes data blocks to its file system, the data blocks are first passed through the `gridftp_multicast` driver. As the `gridftp_multicast` driver passes the data block on to be written to disk, it also forwards the block on to other GridFTP servers in the tree.

Using this approach to add the multicast functionality is minimally invasive to the tested and robust GridFTP server and is entirely modular. The driver is written to a well defined and clean abstraction. Enabling this feature is a simple matter of inserting the driver in the disk stack and passing the driver stack the destination list.

3. Network Overlay

In addition to allowing for multicast, the `gridftp_multicast` driver and this architecture allow us to create a network overlay where many GridFTP servers act as routers forwarding packets along to each other until they get to the final destination where there are written to disk. The advantage of this type of system is actively researched by [Phoebus](http://e2epi.internet2.edu/phoebus.html)¹.

The `gridftp_multicast` driver can be configured to only forward data along to the next server, and to not write it to disk. Furthermore, it can be told to only forward to a single endpoint. When configured in this way, we achieve the network overlay described above.

The following images illustrate this. In the first image we show the standard case where data is sent from a client to a server through the Internet. The routing is done by the Internet outside of the clients control.

Figure B.2. From client to server through Internet



In the next image we show how the `gridftp_multicast` driver can route data through the network via GridFTP servers. This allows the user to have greater control over the network path which the data takes.

Figure B.3. Routing data through network via multicast



4. Results

To show the effectiveness of this architecture we ran experiments on the [UC TeraGrid](http://www.uc.teragrid.org/tg-docs/)². We show some of those results here.

4.1. Experiment 1

In the first experiment, we leased nodes from the UC TeraGrid. 29 hosts were designated as destinations and we ran `gridftp_multicast`-enabled GridFTP servers on them. 1 node was designated as the client node and from it all transfers were started.

All transfers were performed with `globus-url-copy` and a tcp buffer size of 128KB. As a control group, we ran a transfer using `globus-url-copy` with the `-f` option. This caused the source file to be sent to each endpoint in serial. We then transferred the source to all destinations using this architecture.

The first graph shows the completion time of a multicast session against the number of destinations. The first line is when the transfers are performed in a serial fashion from the client. All other lines are multicast sessions performed using this architecture. Each line represents a different vertex degree in the spanning tree (ie, each server forwarding to a different number of destinations).

¹ <http://e2epi.internet2.edu/phoebus.html>

² <http://www.uc.teragrid.org/tg-docs/>

Figure B.4.

As expected, the results for the serial transfer scales linearly while the multicast sessions very slowly increase with more destinations.

The next graph shows the same experiment; but instead of graphing completion time, we graph the clients sending throughput. This is the size of the files being sent (1GB) divided by the time it takes for this file to reach all destinations.

Figure B.5.

The final graph shows the collective bandwidth of a transfer. The graphing function is (# of destinations * file size) / time.

Figure B.6.

4.2. Future Experiments

We created another XIO driver that allows each endpoint in the session to buffer the data. This prevents stalls in sending data transfers due to the latency required to reach a leaf node, and gets data to the disks of nodes higher in the tree faster. Early results show slight improvements on a LAN, but we expect greater results when broadcasting across WANs.

5. Protocol Details

The additions to the protocol are exceptionally minor. Every server in the tree (except for leaf nodes) becomes a client to another server, but that client speaks the standard GridFTP protocol. The only change needed is a command to add the driver to the file system stack, and that command has existed in the GridFTP server for some time.

The command is:

```
SITE SETDISKSTACK 1*{driver name[:driver options]},
```

The second parameter to the site command is a comma-separated list of driver names optionally followed by a colon (:) and a set of driver-specific URL-encoded options. From left to right, the driver names form a stack from bottom to top.

Adding the `gridftp_multicast` driver to this list will enable the multicast functionality. The set of options are the same as those specified in the previous section. The only difference is that each url in the `urls=` options must be url encoded.

6. Usage

The broadcast functionality can be used with **globus-url-copy**. We added the following option:

```
-mc filename
```

The file must contain a line separated list of destination urls. For example:

```
gsiftp://localhost:5000/home/user/tst1
gsiftp://localhost:5000/home/user/tst2
gsiftp://localhost:5000/home/user/tst4
```

The source url is specified on the command line as always. A single destination url may also be specified on the command line in addition to the urls in the file. An example globus-url-copy command is:

```
% globus-url-copy -MC multicast.file gsiftp://localhost/home/user/src_file
```

6.1. Advanced multicasting options

Along with specifying the list of destination urls in a file, a set of options for each url can be specified. This is done by appending a ? to the resource string in the url followed by semicolon-separated key value pairs. For example:

```
gsiftp://dst1.domain.com:5000/home/user/tst1?cc=1;tcpbs=10M;P=4
```

This indicates that the receiving host `dst1.domain.com` will use 4 parallel stream, a tcp buffer size of 10 MB, and will select 1 host when forwarding on data blocks. This url is specified in the `-mc` file as described above.

The following is a list of key=value options and their meanings:

<code>P=<i>integer</i></code>	The number of parallel streams this node will use when forwarding.
<code>cc=<i>integer</i></code>	The number of urls to which this node will forward data.
<code>tcpbs=<i>format-integer</i></code>	The TCP buffer size this node will use when forwarding.
<code>urls=<i>string list</i></code>	The list of urls that must be children of this node when the spanning tree is complete.
<code>local_write=<i>boolean: y/n</i></code>	Determines if this data will be written to a local disk, or just forwarded on to the next hop. This is explained more in the Network Overlay section.
<code>subject=<i>string</i></code>	The DN name to expect from the servers this node is connecting to.

6.2. Required Server Options

For security reasons the GridFTP server does not allow clients to load arbitrary xio drivers into the server. The GridFTP server admin must whitelist the driver individually. White-listing the mlink driver is done with the following parameter to the server:

```
-fs-whitelist file,gridftp_multicast
```

Notice that `file` must also be specified. Without this option, the `file` driver is the default; however, if used, you must specifically list it.

Glossary

some terms not in the docs but wanted in glossary: client/server transfer stream mode (MODE S) improved extended block mode (MODE X)

C

client A process that sends commands and receives responses. Note that in GridFTP, the client may or may not take part in the actual movement of data.

client/server transfer In a client/server transfer, there are only two entities involved in the transfer, the client entity and the server entity. We use the term entity here rather than process because in the implementation provided in GT4, the server entity may actually run as two or more separate processes.

The client will either move data from or to his local host. The client will decide whether or not he wishes to connect to the server to establish the data channel or the server should connect to him (MODE E dictates who must connect).

If the client wishes to connect to the server, he will send the PASV (passive) command. The server will start listening on an ephemeral (random, non-privileged) port and will return the IP and port as a response to the command. The client will then connect to that IP/Port.

If the client wishes to have the server connect to him, the client would start listening on an ephemeral port, and would then send the PORT command which includes the IP/Port as part of the command to the server and the server would initiate the TCP connect. Note that this decision has an impact on traversing firewalls. For instance, the client's host may be behind a firewall and the server may not be able to connect.

Finally, now that the data channel is established, the client will send either the RETR "filename" command to transfer a file from the server to the client (GET), or the STOR "filename" command to transfer a file from the client to the server (PUT).

E

extended block mode (MODE E) MODE E is a critical GridFTP components because it allows for out of order reception of data. This in turn, means we can send the data down multiple paths and do not need to worry if one of the paths is slower than the others and the data arrives out of order. This enables parallelism and striping within GridFTP. In MODE E, a series of "blocks" are sent over the data channel. Each block consists of:

- an 8 bit flag field,
- a 64 bit field indicating the offset in the transfer,
- and a 64 bit field indicating the length of the payload,
- followed by length bytes of payload.

Note that since the offset and length are included in the block, out of order reception is possible, as long as the receiving side can handle it, either via something like a seek on a file, or via some application level buffering and ordering logic that will wait for the out of order blocks.

M

MODE command

In reality, GridFTP is not one protocol, but a collection of several protocols. There is a protocol used on the control channel, but there is a range of protocols available for use on the data channel. Which protocol is used is selected by the MODE command. Four modes are defined: STREAM (S), BLOCK (B), COMPRESSED (C) in RFC 959 for FTP, and EXTENDED BLOCK (E) in GFD.020 for GridFTP. There is also a new data channel protocol, or mode, being defined in the GGF GridFTP Working group which, for lack of a better name at this point, is called MODE X.

See also [extended block mode \(MODE E\)](#)⁸.

See also [stream mode \(MODE S\)](#)⁹.

S

server

A process that receives commands and sends responses to those commands. Since it is a server or service, and it receives commands, it must be listening on a port somewhere to receive the commands. Both FTP and GridFTP have IANA registered ports. For FTP it is port 21, for GridFTP it is port 2811. This is normally handled via inetd or xinetd on Unix variants. However, it is also possible to implement a daemon that listens on the specified port. This is described more fully in in the Architecture section of the GridFTP Developer's Guide.

stream mode (MODE S)

The only mode normally implemented for FTP is MODE S. This is simply sending each byte, one after another over the socket in order, with no application level framing of any kind. This is the default and is what a standard FTP server will use. This is also the default for GridFTP.

T

third party transfers

In the simplest terms, a third party transfer moves a file between two GridFTP servers.

The following is a more detailed, programmatic description.

In a third party transfer, there are three entities involved. The client, who will only orchestrate, but not actually take place in the data transfer, and two servers one of which will be sending data to the other. This scenario is common in Grid applications where you may wish to stage data from a data store somewhere to a super-computer you have reserved. The commands are quite similar to the client/server transfer. However, now the client must establish two control channels, one to each server. He will then choose one to listen, and send it the PASV command. When it responds with the IP/port it is listening on, the client will send that IP/port as

⁸ #extended-block-mode

⁹ #stream-mode

part of the PORT command to the other server. This will cause the second server to connect to the first server, rather than the client. To initiate the actual movement of the data, the client then sends the RETR “filename” command to the server that will read from disk and write to the network (the “sending” server) and will send the STOR “filename” command to the other server which will read from the network and write to the disk (the “receiving” server).

See Also [client/server transfer](#).

Index

A

- accessing data
 - HPSS, 7
 - non-POSIX data source, 6
 - SRB, 8
- admin scenarios
 - running in daemon mode, 35
- API information for GridFTP, 9
- architecture (GridFTP), 6

C

- commandline tool
 - globus-gridftp-server, 22
 - globus-url-copy, 12
- configuration interface for GridFTP, 33
- configuring GridFTP, 33
 - overview, 33
 - types, 33

D

- deploying
 - running under inetd or xinetd, 35

E

- environment variable interface for GridFTP, 37
- errors, 40

G

- globus-gridftp-server, 22
- globus-url-copy, 12
- GUI information for GridFTP, 32

I

- interactive clients
 - UberFTP, 20

L

- logging, 38

M

- moving files
 - single file to many destinations
 - advanced options, 50

R

- running in daemon mode, 35

T

- troubleshooting for GridFTP, 39
- tutorial, 5