GT 4.0 GridFTP

GT 4.0 GridFTP

Table of Contents

List of Tables

Chapter 1. Data Management: Key Concepts

1. Overview of Data Management in GT4

The Globus Toolkit provides a number of components for doing data management. A very high level overview is presented here and then detailed information is given for the individual components by following the component links.

The components available for data management fall into two basic categories: data movement and data replication.

2. Data movement

There are two components related to data movement in the Globus Toolkit: the Globus GridFTP tools and the Globus Reliable File Transfer (RFT) service.

2.1. GridFTP

GridFTP is a protocol defined by Global Grid Forum Recommendation GFD.020, RFC 959, RFC 2228, RFC 2389, and a draft before the IETF FTP working group. The GridFTP protocol provides for the secure, robust, fast and efficient transfer of (especially bulk) data. The Globus Toolkit provides the most commonly used implementation of that protocol, though others do exist (primarily tied to proprietary internal systems).

The Globus Toolkit provides:

- a server implementation called `globus-gridftp-server`,
- a scriptable command line client called `globus-url-copy`, and
- a set of development libraries for custom clients.

While the Globus Toolkit does not provide an interactive client, the GridFTP User's Guide[1] does provide information on at least one interactive client developed by other projects.

If you wish to make data available to others, you need to install a server on a host that can access that data and make sure that there is an appropriate Data Storage Interface (DSI) available for the storage system holding the data. This typically means a standard POSIX file system, but DSIs do exist for the Storage Resource Broker (SRB), the High Performance Storage System (HPSS), and NeST from the Condor team at the University of Wisconsin – Madison. A complete list of DSIs is available [here]. If you need an interface to a storage system not listed here, please contact us. While we certainly cannot offer to write DSIs for every storage system, we can assist in the development, or if a broad enough community can be identified that uses the system, we may be able to obtain joint funding to develop the necessary interface.

If you simply wish to access data that others have made available, you need a GridFTP client. The Globus Toolkit provides a client called `globus-url-copy` for this purpose. This client is capable of accessing data via a range of protocols (http, https, ftp, gsiftp, and file). As noted above this is not an interactive client, but a command line interface, suitable for scripting. For example, the following command:

```
globus-url-copy gsiftp://remote.host.edu/path/to/file file:///path/on/local/host
```

[1] ../gridftp/user-index.html

would transfer a file from a remote host to the locally accessible path specified in the second URL.

Finally, if you wish to add access to files stored behind GridFTP servers, or you need custom client functionality, you can use our very powerful client library to develop custom client functionality.

For more information about GridFTP, see:

- the documentation[2].
- The Globus Striped GridFTP Framework and Server[3]

2.2. Reliable File Transfer (RFT) Service

While globus-url-copy and GridFTP in general are a very powerful set of tools, there are characteristics which may not always be optimal. First, the GridFTP protocol is not a web service protocol (it does not employ SOAP, WSDL, etc). Second, GridFTP requires that the client maintain an open socket connection to the server throughout the transfer. For long transfers this may not be convenient, such as if running from your laptop. While globus-url-copy uses the robustness features of GridFTP to recover from remote failures (network outages, server failures, etc), a failure of the client or the client's host means that recovery is not possible since the information needed for recovery is held in the client's memory. What is needed to address these issues is a service interface based on web services protocols that persists the transfer state in reliable storage. We provide such a service and call it the Reliable File Transfer (RFT) service.

RFT is a Web Services Resource Framework (WSRF) compliant web service that provides “job scheduler"-like functionality for data movement. You simply provide a list of source and destination URLs (including directories or file globs) and then the service writes your job description into a database and then moves the files on your behalf. Once the service has taken your job request, interactions with it are similar to any job scheduler. Service methods are provided for querying the transfer status, or you may use standard WSRF tools (also provided in the Globus Toolkit) to subscribe for notifications of state change events. We provide the service implementation which is installed in a web services container (like all web services) and a very simple client. There are Java classes available for custom development, but due to lack of time and resources, work is still needed to make this easier.

For more information about RFT, see the documentation[4].

3. Data replication

The Replica Location Service (RLS) is one component of data management services for Grid environments. RLS is a tool that provides the ability keep track of one or more copies, or replicas, of files in a Grid environment. This tool, which is included in the Globus Toolkit, is especially helpful for users or applications that need to find where existing files are located in the Grid.

3.1. Replica Location Service (RLS)

RLS is a simple registry that keeps track of where replicas exist on physical storage systems. Users or services register files in RLS when the files are created. Later, users query RLS servers to find these replicas.

RLS is a distributed registry, meaning that it may consist of multiple servers at different sites. By distributing the RLS registry, we are able to increase the overall scale of the system and store more mappings than would be possible in a single, centralized catalog. We also avoid creating a single point of failure in the Grid data management system. If desired, RLS can also be deployed as a single, centralized server.

[2] ../gridftp/
[3] http://www.globus.org/alliance/publications/papers/gridftp_final.pdf
[4] ../rft/

Before explaining RLS in detail, we need to define a few terms.

- A *logical file name* is a unique identifier for the contents of a file.
- A *physical file name* is the location of a copy of the file on a storage system.

These terms are illustrated in Figure 1 (below). The job of RLS is to maintain associations, or mappings, between logical file names and one or more physical file names of replicas. A user can provide a logical file name to an RLS server and ask for all the registered physical file names of replicas. The user can also query an RLS server to find the logical file name associated with a particular physical file location.

In addition, RLS allows users to associate attributes or descriptive information (such as size or checksum) with logical or physical file names that are registered in the catalog. Users can also query RLS based on these attributes.

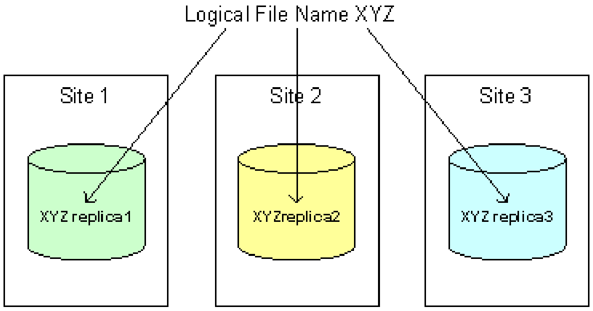


Figure 1. Example of the associations between a logical file name and three replicas on different storage sites.

3.2. Using RLS: An Example

One example of a system that uses RLS as part of its data management infrastructure is the Laser Interferometer Gravitational Wave Observatory (LIGO) project. LIGO scientists have instruments at two sites that are designed to detect the existence of gravitational waves. During a run of scientific experiments each LIGO instrument site produces millions of data files. Scientists at eight other sites want to copy these large data sets to their local storage systems so that they can run scientific analysis on the data. Therefore, each LIGO data file may be replicated at up to ten physical locations in the Grid. LIGO deploys RLS servers at each site to register local mappings and to collect information about mappings at other LIGO sites. To find a copy of a data file a scientist requests the file from LIGO's data management system, called the Lightweight Data Replicator (LDR). LDR queries the Replica Location Service to find out whether there is a local copy of the file; if not, RLS tells the data management system where the file exists in the Grid. Then the LDR system generates a request to copy the file to the local storage system and registers the new copy in the local RLS server.

LIGO currently uses the Replica Location Service in its production data management environment. The system registers mappings between more than 3 million logical file names and 30 million physical file locations.

3.3. For more information

For more detailed key concepts about RLS, click here[5].

For more information about RLS, see the documentation[6].

[5] rls.html
[6] ../rls/

4. Higher level data services

GT 4.0 also provides a higher-level data management service that combines two existing data management components: RFT and RLS.

4.1. Data Replication Service (DRS)

For the Technical Preview of the Globus Toolkit 4.0 release we have designed and implemented a Data Replication Service (DRS) that provides a pull-based replication capability for Grid files. The DRS is a higher-level data management service that is built on top of two GT data management components: the Reliable File Transfer (RFT) Service and the Replica Location Service (RLS).

The function of the DRS is to ensure that a specified set of files exists on a storage site. The DRS begins by querying RLS to discover where the desired files exist in the Grid. After the files are located, the DRS creates a transfer request that is executed by RFT. After the transfers are completed, DRS registers the new replicas with RLS.

DRS is implemented as a Web service and complies with the Web Services Resource Framework (WSRF) specifications. When a DRS request is received, it creates a WS-Resource that is used to maintain state about each file being replicated, including which operations on the file have succeeded or failed.

4.1.1. For more information

For more information about DRS, go to the Tech Preview documentation[7].

[7] ../../techpreview/datarep/

Chapter 2. GT 4.0.0 Release Notes: GridFTP

1. Component Overview

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. We have selected a set of protocol features and extensions defined already in IETF RFCs and added a few additional features to meet requirements from current data grid projects.

2. Feature Summary

Features new in GT 4.0

- A new, complete reimplementation of the *server*.
- Support for *striping*.
- This new implementation will greatly ease new feature additions and modifications of the server (new commands, new data sources such as mass storage devices, etc.), maintainability, and resolves a licensing issue that was discovered.

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*).
- 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.

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

3. Bug Fixes

- Bug 1883[1]: globus_ftp_control server does not decode ADAT properly.
- Bug 1928[2]: Strange problems: Busy wait and address in use.
- Bug 2036[3]: New gridftp, globus-gridftp-server, does not use globus_gss_assist_map_and_authorize
- Bug 2128[4]: gridftp server checks ownership of hostcert after setuid
- Bug 2129[5]: Gridftp logging levels (is 7>10?)
- Bug 2132[6]: Server responds with "Valid credentials could not be found..."
- Bug 2452[7]: non-striped gridFTP not working w/ round robin
- Bug 2463[8]: striped server, small partial transfers hang
- Bug 2464[9]: striped server to non-striped server doesn't work

4. Known Problems

- GridFTP Server
  - There are some small memory leaks, though they should not grow much.
  - Threaded builds should work, but increased pollers may produce hangs (please report these).
  - Some error responses are unclear.

[1] http://bugzilla.globus.org/globus/show_bug.cgi?id=1883
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=1928
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=2036
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=2128
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=2129
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=2132
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=2452
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=2463
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=2464

- Bug 2547[10]: gridftp config detach option doesn't work
- See the known issues section of source-trees/gridftp/server/src/TODO in the source installer for other issues.

5. Technology Dependencies

GridFTP depends on the following GT components:

- Pre-WS Authentication / Authorization
- C Common Libraries
- XIO

GridFTP depends on the following 3rd party software:

- OpenSSL (version included in release)

6. 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.

7. Backward Compatibility Summary

Protocol changes since GT 3.2

- None

API changes since GT 3.2

- None

Exception changes since GT 3.2

[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=2547

- Not Applicable (GridFTP is not Java-based)

Schema changes since GT 3.2

- Not Applicable (GridFTP is not SOAP-based)

8. For More Information

Click here[11] for more information about this component.

[11] index.html

Chapter 3. GT 4.0.1 Incremental Release Notes: GridFTP

1. Introduction

These release notes are for the incremental release 4.0.1. It includes a summary of changes since 4.0.0, bug fixes since 4.0.0 and any known problems that still exist at the time of the 4.0.1 release. This page is in addition to the top-level 4.0.1 release notes at http://www.globus.org/toolkit/releasenotes/4.0.1.

For release notes about 4.0.0 (including feature summary, technology dependencies, etc) go to the GridFTP 4.0.0 Release Notes[1].

2. Changes Summary

The following change has occurred for GridFTP:

- A protocol change was made to match a wuftpd extension regarding listings. This resolved an issue with a function in the Java Cog Kit. This breaks technical protocol compatibility with RFC959 (FTP), but matches the behavior of 2.4.x servers. See Bug 3507[2] for details.

3. Bug Fixes

The following bugs were fixed for GridFTP:

- Bug 3351:[3] Initialization errors are ignored
- Bug 3369:[4] globus-gridftp-server single mode operation fails.
- Bug 3364:[5] authz callback bug in case of negative authorization decisions?
- Bug 3487:[6] SRB DSI does not authenticate properly with SRB
- Bug 3453:[7] ~ expansion does not work in CWD argument
- Bug 3402:[8] Running GridFTP in privilege separation mode on non-Linux platforms
- Bug 3465:[9] SITE VERSION
- Bug 3386:[10] hostname and interface options require an ip address

[1] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=3507
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=3351
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=3369
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=3364
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=3487
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=3453
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=3402
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=3465
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=3386

- Bug 3507:[11] cannot list with jglobus 1.2
- Bug 3568:[12] list function terminates after 200 files
- Bug 3608:[13] threaded gridftp hangs on lists (LIST, MLSx)

4. Known Problems

The following problems are known to exist for GridFTP at the time of the 4.0.1 release:

- Bug 3324:[14] gridftpd croaking on startup - globus_l_io_tcp_contact_string failed
- Bug 3214:[15] Backend of GT4 GridFTPd should ignore ~/.globus/certificates
- Bug 3387:[16] client library doesn't respect GLOBUS_HOSTNAME
- Bug 3618:[17] GridFTP fails to transfer a file if I've only group permissions
- Bug 3622:[18] gridftp server improperly handles detach option in config file

5. For More Information

Click here[19] for more information about this component.

[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=3507
[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=3568
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=3608
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=3324
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3214
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3387
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=3618
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=3622
[19] index.html

Chapter 4. GT 4.0.2 Incremental Release Notes: GridFTP

1. Introduction

These release notes are for the incremental release 4.0.2. It includes a summary of changes since 4.0.1, bug fixes since 4.0.1 and any known problems that still exist at the time of the 4.0.2 release. This page is in addition to the top-level 4.0.2 release notes at http://www.globus.org/toolkit/releasenotes/4.0.2.

For release notes about 4.0.0 (including feature summary, technology dependencies, etc) go to the GridFTP 4.0.0 Release Notes[1].

2. Changes Summary

Aside from bug fixes, no changes have been made for GridFTP since 4.0.1. However, please remember that a protocol change was made in 4.0.1 to match a wuftpd extension regarding listings and resolve an issue with a function in the Java Cog Kit. This breaks technical protocol compatibility with RFC959 (FTP), but matches the behavior of 2.4.x servers. See Bug 3507[2] for details.

3. Bug Fixes

The following bugs were fixed for GridFTP:

- Bug 3618:[3] GridFTP fails to transfer a file if I've only group permissions
- Bug 3632:[4] CDUP/CWD dir parsing problems
- Bug 3769:[5] -cd wont create more than one level of directories
- Bug 3902:[6] 500 Command failed. : globus_xio: An end of file occurred
- Bug 3919:[7] GT 4.0.1 g-u-c dumps core in batch mode with restartability
- Bug 3956:[8] Globus-url-copy does not create multilevel directories.
- Bug 4149:[9] env GLOBUS_HOSTNAME=invalid globus-gass-server abort
- Bug 4185:[10] Recursive directory transfer failure in globus-url-copy
- Bug 4208:[11] gridftp server segfaults if it can't open its logfile

[1] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=3507
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=3618
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=3632
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=3769
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=3902
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=3919
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=3956
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=4149
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=4185
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=4208

- Bug 4280:[12] GridFtp server could not take long home directory name

4. Known Problems

The following problems are known to exist for GridFTP at the time of the 4.0.2 release:

- Bug 3324:[13] gridftpd croaking on startup - globus_l_io_tcp_contact_string failed
- Bug 3214:[14] Backend of GT4 GridFTPd should ignore ~/.globus/certificates
- Bug 3387:[15] client library doesn't respect GLOBUS_HOSTNAME
- Bug 3622:[16] gridftp server improperly handles detach option in config file
- Bug 3761:[17] globus_ftp_control library hangs at various places (depending on os)
- Bug 3807:[18] "not a plain file" could be "file not found"
- Bug 3845:[19] data channel authentication failure
- Bug 4119:[20] Malformed UTF-8 character problem in globus-ftp-client tests
- Bug 4274:[21] client library has no default striping layout.
- Bug 4344:[22] Globus-url-copy client does not reuse the connection when run in 3rd party mode

5. For More Information

Click here[23] for more information about this component.

[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=4280
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=3324
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=3214
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3387
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3622
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=3761
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=3807
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=3845
[20] http://bugzilla.globus.org/globus/show_bug.cgi?id=4119
[21] http://bugzilla.globus.org/globus/show_bug.cgi?id=4274
[22] http://bugzilla.globus.org/globus/show_bug.cgi?id=4344
[23] index.html

Chapter 5. GT 4.0.3 Incremental Release Notes: GridFTP

1. Introduction

These release notes are for the incremental release 4.0.3. It includes a summary of changes since 4.0.2, bug fixes since 4.0.2 and any known problems that still exist at the time of the 4.0.3 release. This page is in addition to the top-level 4.0.3 release notes at http://www.globus.org/toolkit/releasenotes/4.0.3.

For release notes about 4.0.0 (including feature summary, technology dependencies, etc) go to the GridFTP 4.0.0 Release Notes[1].

2. Changes Summary

No changes have been made for GridFTP since 4.0.2. However, please remember that a protocol change was made in 4.0.1 to match a wuftpd extension regarding listings and resolve an issue with a function in the Java Cog Kit. This breaks technical protocol compatibility with RFC959 (FTP), but matches the behavior of 2.4.x servers. See Bug 3507[2] for details.

3. Bug Fixes

No bugs for GridFTP were fixed since 4.0.2.

4. Known Problems

The following problems are known to exist for GridFTP at the time of the 4.0.3 release:

- Bug 3324:[3] gridftpd croaking on startup - globus_l_io_tcp_contact_string failed
- Bug 3214:[4] Backend of GT4 GridFTPd should ignore ~/.globus/certificates
- Bug 3387:[5] client library doesn't respect GLOBUS_HOSTNAME
- Bug 3622:[6] gridftp server improperly handles detach option in config file
- Bug 3761:[7] globus_ftp_control library hangs at various places (depending on os)
- Bug 3807:[8] "not a plain file" could be "file not found"
- Bug 3845:[9] data channel authentication failure

[1] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=3507
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=3324
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=3214
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=3387
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=3622
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=3761
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=3807
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=3845

- Bug 4119:[10] Malformed UTF-8 character problem in globus-ftp-client tests
- Bug 4274:[11] client library has no default striping layout.
- Bug 4344:[12] Globus-url-copy client does not reuse the connection when run in 3rd party mode

5. For More Information

Click here[13] for more information about this component.

[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=4119
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=4274
[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=4344
[13] index.html

Chapter 6. GT 4.0.4 Incremental Release Notes: GridFTP

1. Introduction

These release notes are for the incremental release 4.0.4. It includes a summary of changes since 4.0.3, bug fixes since 4.0.3 and any known problems that still exist at the time of the 4.0.4 release. This page is in addition to the top-level 4.0.4 release notes at http://www.globus.org/toolkit/releasenotes/4.0.4.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the GridFTP 4.0 Release Notes[1].

2. Changes Summary

No changes have been made for GridFTP since 4.0.3 other than the listed bugfixes.

3. Bug Fixes

- Bug 3141:[2] globus-url-copy segfaults when attempting to glob with a server-side auth error
- Bug 4344:[3] Globus-url-copy client does not reuse the connection when run in 3rd party mode.
- Bug 4571:[4] 4.0.2 globus-url-copy core dumps with -cd -f option on x86_64
- Bug 4735:[5] globus_l_ftp_control_read_cb() message pull up bug
- Bug 4786:[6] Globus Gridftp server represent pipe as 'x' instead of 'p'?
- Bug 4825:[7] gridftp-server process fails to exit and busy waits if you have a zero length CRL
- Bug 4945:[8] GridFTP client accesses freed memory
- Bug 4975:[9] globus-url-copy fails on some transfers to gass server

4. Known Problems

- Bug 3324:[10] gridftpd croaking on startup - globus_l_io_tcp_contact_string failed
- Bug 3214:[11] Backend of GT4 GridFTPd should ignore ~/.globus/certificates

[1] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Release_Notes.html
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=3141
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=4344
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=4571
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=4735
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=4786
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=4825
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=4945
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=4975
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=3324
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=3214

- Bug 3387:[12] client library doesn't respect GLOBUS_HOSTNAME
- Bug 3622:[13] gridftp server improperly handles detach option in config file
- Bug 3761:[14] globus_ftp_control library hangs at various places (depending on os)
- Bug 3807:[15] "not a plain file" could be "file not found"
- Bug 3845:[16] data channel authentication failure
- Bug 4119:[17] Malformed UTF-8 character problem in globus-ftp-client tests
- Bug 4274:[18] client library has no default striping layout.
- Bug 4861:[19] Problem with HTTP chunked mode transfer in globus-url-copy

5. For More Information

Click here[20] for more information about this component.

[12] http://bugzilla.globus.org/globus/show_bug.cgi?id=3387
[13] http://bugzilla.globus.org/globus/show_bug.cgi?id=3622
[14] http://bugzilla.globus.org/globus/show_bug.cgi?id=3761
[15] http://bugzilla.globus.org/globus/show_bug.cgi?id=3807
[16] http://bugzilla.globus.org/globus/show_bug.cgi?id=3845
[17] http://bugzilla.globus.org/globus/show_bug.cgi?id=4119
[18] http://bugzilla.globus.org/globus/show_bug.cgi?id=4274
[19] http://bugzilla.globus.org/globus/show_bug.cgi?id=4861
[20] index.html

Chapter 7. GT 4.0 GridFTP : System Administrator's Guide

1. Introduction

This guide contains advanced configuration information for system administrators working with GridFTP. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation. This guide should help you configure and run the GridFTP *server* in some standard configurations.

Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the GT 4.0 System Administrator's Guide[1]. Read through this guide before continuing!

2. Building and Installing

GridFTP is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide[2]. No extra installation steps are required for this component.

2.1. Building only GridFTP and Utilities

If you wish to install GridFTP without installing the rest of the Globus Toolkit, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[3]. Perform steps 1-3, as written (Note that you do not need Ant, a JDK, or a JDBC database to build only GridFTP). However, instead of running "make" as directed in step 4,

Run:

```
globus$ make gridftp
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make gridftp 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

2.2. Building only the GridFTP server

If you wish to install only the GridFTP server, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[4] for prerequisites. Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
globus$ make gpt globus_gridftp_server
```

[1] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[2] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[3] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html
[4] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html

If you wish to have a log file of the build, use **tee**:

```
globus$ make gpt globus_gridftp_server 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

2.3. Building only the GridFTP client

If you wish to install only the GridFTP *client*, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[5] for prerequisites. Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
globus$ make globus-data-management-client
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make globus-data-management-client 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

2.4. Building only the GridFTP SDK

If you wish to install only the GridFTP SDK, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[6] for prerequisites. Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
globus$ make globus-data-management-sdk
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make globus-data-management-sdk 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

2.5. Building a combination of GridFTP elements

If you wish to build a combination of GridFTP elements, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[7] for prerequisites. Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
globus$ make [any combination of the above commands, each separated by a space]
```

For example, if you just want to install the GridFTP server and client, the command would be:

```
globus$ make gpt globus_gridftp_server globus-data-management-client
```

[5] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html
[6] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html
[7] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html

If you wish to have a log file of the build, use **tee**:

```
globus$ make [any combination of the above commands, each separated by a space] 2>&1 | tee
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

2.6. Building and Installing a static GridFTP server

If you wish to build and install a statically linked set of GridFTP binaries, refer to the Installing GT 4.0 section of the GT 4.0 System Administrator's Guide[8] for prerequisites. Follow steps 1-2 as written. In step 3, however, you should

Run:

```
globus$ export GLOBUS_LOCATION=/usr/local/globus-4.0.0
globus$ ./configure --prefix=$GLOBUS_LOCATION --with-buildopts="--static"

globus$ make gpt globus_gridftp_server
```

If you wish to have a log file of the build, use **tee**:

```
globus$ make gpt globus_gridftp_server 2>&1 | tee build.log
```

The syntax above assumes a Bourne shell. If you are using another shell, redirect stderr to stdout and then pipe it to **tee**.

3. Configuring

3.1. GridFTP server configuration overview

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 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
```

[8] http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html

```
anonymous_user bob
banner "Welcome!"
```

3.2. GridFTP server configuration options

The table below lists config file options, associated command line options (if available) and descriptions. Note that any boolean option can be negated on the command line by preceding the specified option with '-no-' or '-n'. example: -no-cas or -nf.

Table 7.1. Informational Options

help <0|1> -h -help	Show usage information and exit. Default value: FALSE
longhelp <0|1> -hh -longhelp	Show more 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

Table 7.2. 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. Default value: TRUE
detach <0|1> -S -detach	Run as a background daemon detached from any controlling terminals. 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

Table 7.3. 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, it 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 CAS authorization. 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: none, host, self or subject:[subject]. Default value: host
allow_anonymous <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
anonymous_names_allowed <string> -anonymous-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 username. Default value: not set
anonymous_user <string> -anonymous-user <string>	User to setuid to for an anonymous connection. Only applies when running as root. Default value: not set
anonymous_group <string> -anonymous-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

connections_max <number> -connections-max <number>	Maximum concurrent connections allowed. Only applies when running in daemon mode. Unlimited if not set. Default value: not set
connections_disabled <0|1> -connections-disabled	Disable all new connections. Does not affect ongoing connections. This would have to be set in the configuration file and then the server issued a SIGHUP in order to reload the config. Default value: FALSE

Table 7.4. Logging Options

log_level <string> -d <string> -log-level <string>	Log level. A comma separated list of levels from: 'ERROR, WARN, INFO, DUMP, ALL'. Example: error,warn,info. You may also specify a numeric level of 1-255. Default value: ERROR
log_module <string> -log-module <string>	globus_logging module that will be loaded. If not set the default 'stdio' module will be used, and the logfile options apply. 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' and 'buffer', for buffer flush interval and buffer size, respectively. The default options are a 64k buffer size and a 5 second flush interval. A 0 second flush interval will disable periodic flushing, and the buffer will only flush when it is full. A value of 0 for buffer will disable buffering and all messages will be written immediately. Example: -log-module stdio:buffer=4096:interval=10 Default value: not set
log_single <string> -l <string> -logfile <string>	Path of a single file to log all activity to. 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. Default value: not set
log_unique <string> -L <string> -logdir <string>	Partial path to which 'gridftp.(pid).log' will be appended to construct the log filename. Example: -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 ex: DATE=20050520163008.306532 HOST=localhost PROG=globus-gridftp-server NL.EVNT=FTP_INFO START=20050520163008.305913 USER=ftp FILE=/etc/group BUFFER=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). DATE: time the transfer completed. START: time the transfer started. HOST: hostname of the server. USER: username on the host that transfered 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 transfered. 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_usage_stats <0|1> -disable-usage-stats	Disable transmission of per-transfer usage statistics. See the Usage Statistics[9] section in the online documentation for more information. Default value: FALSE
usage_stats_target <string> -usage-stats-target <string>	Comma separated list of contact strings for usage statistics listeners. Default value: not set

[9] ../../Usage_Stats.html

Table 7.5. Single and Striped Remote Data Node Options

remote_nodes <string> -r <string> -remote-nodes <string>	Comma separated list of remote node contact strings. Default value: not set
data_node <0|1> -dn -data-node	This server is a back end data node. Default value: FALSE
stripe_blocksize <number> -sbs <number> -stripe-blocksize <number>	Size in bytes of sequential data that each stripe will transfer. Default value: 1048576
stripe_layout <number> -sl <number> -stripe-layout <number>	Stripe layout. 1 = Partitioned, 2 = Blocked. Default value: 2
stripe_blocksize_locked <0|1> -stripe-blocksize-locked	Do not allow client to override stripe blocksize with the OPTS RETR command. Default value: FALSE
stripe_layout_locked <0|1> -stripe-layout-locked	Do not allow client to override stripe layout with the OPTS RETR command. Default value: FALSE

Table 7.6. Disk Options

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

Table 7.7. 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. 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

Table 7.8. 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-idle-timeout <number>	Time in seconds to allow a client to remain connected to the control channel without activity. Default value: 600
ipc_idle_timeout <number> -ipc-idle-timeout <number>	Idle time in seconds before an unused ipc connection will close. Default value: 600
ipc_connect_timeout <number> -ipc-connect-timeout <number>	Time in seconds before cancelling an attempted ipc connection. Default value: 60

Table 7.9. User Messages

banner <string> -banner <string>	Message to display to the client before authentication. Default value: not set
banner_file <string> -banner-file <string>	File to read banner message from. Default value: not set
banner_terse <0|1> -banner-terse	When this is set, the minimum allowed banner message will be displayed to unauthenticated clients. Default value: FALSE
login_msg <string> -login-msg <string>	Message to display to the client after authentication. Default value: not set
login_msg_file <string> -login-msg-file <string>	File to read login message from. Default value: not set

Table 7.10. Module Options

load_dsi_module <string> -dsi <string>	Data Storage Interface module to load. File and remote modules are defined by the server. If not set the file module is loaded, unless the 'remote' option is specified, in which case the remote 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> -allowed-modules <string>	Comma separated list of ERET/ESTO modules to allow and, optionally, specify an alias for. Example: module1,alias2:module2,module3 (module2 will be loaded when a client asks for alias2). Default value: not set

Table 7.11. Other

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

3.3. Configuring the GridFTP server to run under xinetd/inetd

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:

```
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 (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. However, on OS X you could also use launchd, as shown below.

Here is a sample Mac OS X launchd config entry. Create a "/Library/LaunchDaemons/gsiftp.plist" file. An example is below. Edit it to have the right paths for your installation. Then run "sudo launchctl load /Library/LaunchDaemons/gsiftp.plist".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http:// www.apple.com/DTDs/
<plist version="1.0">
<dict>
         <key>Debug</key>
         <true/>
         <key>EnvironmentVariables</key>
         <dict>
                 <key>DYLD_LIBRARY_PATH</key>
                 <string>/usr/local/gt4/lib</string>
                 <key>GLOBUS_LOCATION</key>
                 <string>/usr/local/gt4</string>
         </dict>
         <key>GroupName</key>
         <string>admin</string>
         <key>Label</key>
         <string>org.globus.gridftp</string>
         <key>OnDemand</key>
         <true/>
         <key>ProgramArguments</key>
         <array>
                 <string>/usr/local/gt4/sbin/globus-gridftp-server</
string>
                 <string>-i</string>
         </array>
         <key>ServiceDescription</key>
         <string>GridFTP</string>
         <key>Sockets</key>
         <dict>
                 <key>Listeners</key>
                 <dict>
                         <key>SockFamily</key>
                         <string>IPv4</string>
                         <key>SockPassive</key>
                         <true/>
                         <key>SockServiceName</key>
                         <string>gsiftp</string>
                         <key>SockType</key>
                         <string>stream</string>
                 </dict>
         </dict>
         <key>UserName</key>
         <string>root</string>
         <key>inetdCompatibility</key>
         <dict>
                 <key>Wait</key>
                 <false/>
         </dict>
</dict>
</plist>
```

3.4. Configuring GridFTP to run with the Community Authorization Service (CAS)

The Community Authorization Service (CAS)[10] is used to administer access rights to files and directories and the GridFTP server can be configured to enforce those rights.

For more information, see How to Set Up CAS to Use with GridFTP[11].

[10] http://www.globus.org/toolkit/docs/4.0/security/cas/
[11] http://www.globus.org/toolkit/docs/4.0/security/cas/WS_AA_CAS_HOWTO_Setup_GridFTP.html

4. Deploying the GridFTP Server: `globus-gridftp-server`

It is assumed that the toolkit installation was successful and that Globus security is properly configured. For more information, see the Installation Guide[12].

4.1. Running in daemon mode

The server should generally be run as root in daemon mode, though 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:

- 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`.

4.2. Running under inetd or xinetd

The -i command line option enables the server to be run under inetd or xinetd.

See Configuring GridFTP for example xinetd and inetd configuration entries.

4.3. Remote data-nodes and striped operation

The GridFTP server now supports separate front end (client control connection) and back end (data node) processes. In addition, a single front end process may connect to multiple back end data nodes.

When multiple back end data nodes are available, the server is said to be in a striped configuration, or simply, is a striped server. In this mode transfers are divided over all available data nodes, thus allowing the combined bandwidth of all data nodes to be used.

Note: The connection between the front end and data nodes is referred to as the *ipc channel*.

The ability to use inetd or daemon execution modes applies to both front end servers and data nodes, and the same certificate and user requirements apply.

To start the front end:

```
globus-gridftp-server <args> -r <host:port>[,<host:port>,...]
```

[12] http://www.globus.org/toolkit/docs/4.0/admin/docbook/

To start the data-node:

```
globus-gridftp-server -p <port> -dn
```

The `-p <port>` option used on the data-node is the port that will be used for ipc connections. This is the port that you will register with the front end server.

For example:

```
machineB> globus-gridftp-server -p 6000 -dn
machineC> globus-gridftp-server -p 7000 -dn
machineA> globus-gridftp-server -p 5000 -r machineB:6000,machineC:7000
```

The client would only connect to the front end at `machineA:5000`, for example, using globus-url-copy with the -stripe option:

```
globus-url-copy -stripe gsiftp://machineA:5000/file file:///destination
   or
globus-url-copy -stripe gsiftp://machineA:5000/file gsiftp://machineX/destination
```

Where machineX may be another striped server or a standard GridFTP server.

4.4. Separation of Processes

As is illustrated above, the GridFTP server can be separated into front end and data node processes. This is the architecture used to achieve a striped server, but it can also be exploited to achieve a higher level of security.

Running the server as root is often desirable because it allows the server to fork and setuid on a child processes related to an authenticated user. This allows the server to leverage the operating systems file system permissions and other security devices. However, it is not at all desirable to have a root running process listening on a port open to the world. If an attacker were to compromise the process they could obtain root level access to the machine.

To overcome this security risk the gridftp server can be run in a front end/back end manner. The front end can be run as any user, say user globus, that has very limited access to the machine. The front end is the processes open to the outside world. If it is compromised an attacker has only gained access to that limited account. The back end is run as root, but configured to only allow connections from the front end.

To start the front end:

```
globus-gridftp-server -p 7000 -r localhost:7001
```

and the back end:

```
globus-gridftp-server -p 7001 -dn -allow-from 127.0.0.1
```

5. Testing

If the globus-ftp-client-test package has been installed, our standard test suite may be run to verify functionality on your platform. Simply set up the globus environment, chdir to `$GLOBUS_LOCATION/test/globus_ftp_client_test/` and run `./TESTS.pl`.

Please also see the Call for Community Testing[13].

[13] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Call_for_Testing.html

6. Security Considerations

The following are points to consider relative to security:

6.1. Two ways to configure your *server*

We now provide two ways to configuring your server:

- The classic installation. This is equivalent to any FTP server you would normally install. It is run as a root setuid process. Once the user is authenticated, the process does a setuid to the appropriate non-privileged user account.
- A new split process installation. In this configuration, the server consists of two processes:
  - The control channel (the process the external user connects to) runs as a non-privileged user (typically the globus user).
  - The data channel (the process that access the file system and moves the data) runs as a root setuid program as before but is only contacted by the control channel process 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.

6.2. New authentication options

There are new authentication options available for the server in GT4.0.0:

- Anonymous: The server now supports anonymous access. In order for this to work, a configuration switch must explicitly enable it, a list of acceptable usernames must be defined, and an account under which the anonymous user should run must be defined. If the necessary configurations are in place, and the *client* presents a username that is in the list of acceptable anonymous users, then the session will be accepted and the process will setuid to the anonymous user account. We do not support chroot in this version of the server.
- Username / Password: This is standard FTP authentication. It uses a separate password file, used only by the GridFTP server, *NOT* the system password file.

Warning

WE HIGHLY RECOMMEND YOU NOT USE THIS. YOU WILL BE SENDING YOUR PASSWORD IN CLEAR TEXT OVER THE NETWORK.

We do, however, have some user communities who run only on internal networks for testing purposes and who do not wish to deal with obtaining GSI credentials. If you are considering this, we would recommend that you look at Simple CA and set up your own testbed CA. This can be done in less than an hour and then provides you full GSI security.

6.3. Firewall requirements

If the GridFTP *server* is behind a firewall:

1. Contact your network administrator to open up port 2811 (for GridFTP control channel connection) and a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.
2. Set the environment variable GLOBUS_TCP_PORT_RANGE:

```
export GLOBUS_TCP_PORT_RANGE=min,max
```

where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP server to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_SOURCE_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP server to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Note

If the server is behind NAT, the `--data-interface <real ip/hostname>` option needs to be used on the server.

If the GridFTP *client* is behind a firewall:

1. Contact your network administrator to open up a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

2. Set the environment variable GLOBUS_TCP_PORT_RANGE

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP client to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP client to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Additional information on Globus Toolkit Firewall Requirements is available here[14].

7. 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.

[14] http://www.globus.org/toolkit/security/firewalls/

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 Configuration information for more information on these and other configuration options.

7.1. 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.

7.2. 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.

7.3. 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 discuss@globus.org. 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).

8. Usage statistics collection by the Globus Alliance

The following GridFTP-specific usage statistics are sent in a UDP packet at the end of each transfer, in addition to the standard header information described in the Usage Stats[15] section.

- Start time of the transfer
- End time of the transfer
- Version string of the server
- TCP buffer size used for the transfer

[15] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

- Block size used for the transfer
- Total number of bytes transferred
- Number of parallel streams used for the transfer
- Number of stripes used for the transfer
- Type of transfer (STOR, RETR, LIST)
- FTP response code -- Success or failure of the transfer

Note

The client (globus-url-copy) does NOT send any data. It is the *servers* that send the usage statistics.

We have made a concerted effort to collect only data that is not too intrusive or private and yet still provides us with information that will help improve and gauge the usage of the GridFTP server. Nevertheless, if you wish to disable this feature for GridFTP only, see the Logging section of Section 3.2, “GridFTP server configuration options”. Note that you can disable transmission of usage statistics globally for all C components by setting "GLOBUS_USAGE_OPTOUT=1" in your environment.

Also, please see our policy statement[16] on the collection of usage statistics.

[16] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 8. GT 4.0 GridFTP : User's Guide

1. Introduction

The GridFTP User's Guide provides general end user-oriented information.

2. Usage scenarios

2.1. Basic procedure for using GridFTP (globus-url-copy)

If you just want the "rules of thumb" on getting started (without all the details), the following options using `globus-url-copy` will normally give acceptable performance:

```
globus-url-copy -vb -tcp-bs 2097152 -p 4 source_url destination_url
```

The source/destination URLs will normally be one of the following:

- *`file:///path/to/my/file`* if you are accessing a file on a file system accessible by the host on which you are running your *client*.
- *`gsiftp://hostname/path/to/remote/file`* if you are accessing a file from a GridFTP *server*.

2.1.1. Putting files

One of the most basic tasks in GridFTP is to "put" files, i.e., moving a file from your file system to the server. So for example, if you want to move the file `/tmp/foo` from a file system accessible to the host on which you are running your client to a file name `/tmp/bar` on a host named `remote.machine.my.edu` running a GridFTP server, you would use this command:

```
globus-url-copy -vb -tcp-bs 2097152 -p 4 file:///tmp/foo gsiftp://remote.machine.my.edu/tm
```

Note

In theory, `remote.machine.my.edu` could be the same host as the one on which you are running your client, but that is normally only done in testing situations.

2.1.2. Getting files

A get, i.e, moving a file from a server to your file system, would just reverse the source and destination URLs:

Tip

Remember *`file:`* always refers to your file system.

```
globus-url-copy -vb -tcp-bs 2097152 -p 4 gsiftp://remote.machine.my.edu/tmp/bar file:///tm
```

2.1.3. Third party transfers

Finally, if you want to move a file between two GridFTP servers (a *third party transfer*), both URLs would use `gsiftp:` as the protocol:

```
globus-url-copy -vb -tcp-bs 2097152 -p 4 gsiftp://other.machine.my.edu/tmp/foo gsiftp://re
```

2.1.4. For more information

If you want more information and details on URLs and the command line options, the Key Concepts Guide[1] gives basic definitions and an overview of the GridFTP protocol as well as our implementation of it.

2.2. Accessing data in...

2.2.1. Accessing data in a non-POSIX file data source that has a POSIX interface

If you want to access data in a non-POSIX file data source that has a POSIX interface, the standard server will do just fine. Just make sure it is really POSIX-like (out of order writes, contiguous byte writes, etc).

2.2.2. Accessing data in HPSS

The following information is helpful if you want to use GridFTP to access data in HPSS.

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 GT4.0.x 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).

2.2.2.1. 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.

2.2.2.2. 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.

[1] http://www.globus.org/toolkit/docs/4.0/data/key/

2.2.2.3. 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.

2.2.2.4. HPSS info

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.

2.2.3. Accessing data in SRB

The following information is helpful if you want to use GridFTP to access data in SRB.

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 GT4.0.x 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).

2.2.3.1. 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.

2.2.3.2. 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.

2.2.3.3. 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.

2.2.3.4. SRB info

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.

2.2.4. Accessing data in some other non-POSIX data source

The following information is helpful If you want to use GridFTP to access data in a non-POSIX data source.

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 GT4.0.x 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).

2.2.4.1. 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.

2.2.4.2. 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.

2.2.4.3. 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.

3. Command line tools

Please see the GridFTP Command Reference.

4. Graphical user interfaces

Globus does not provide any interactive client for GridFTP, either GUI or text based. However, NCSA, as part of there TeraGrid activity, produces a text based interactive client called UberFTP, which you may want to check out. See Interactive Clients[2] for more information.

[2] rn01re01.html#interactiveclients

5. Security Considerations

The following are points to consider relative to security:

5.1. Two ways to configure your *server*

We now provide two ways to configuring your server:

- The classic installation. This is equivalent to any FTP server you would normally install. It is run as a root setuid process. Once the user is authenticated, the process does a setuid to the appropriate non-privileged user account.
- A new split process installation. In this configuration, the server consists of two processes:
  - The control channel (the process the external user connects to) runs as a non-privileged user (typically the globus user).
  - The data channel (the process that access the file system and moves the data) runs as a root setuid program as before but is only contacted by the control channel process 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.

5.2. New authentication options

There are new authentication options available for the server in GT4.0.0:

- Anonymous: The server now supports anonymous access. In order for this to work, a configuration switch must explicitly enable it, a list of acceptable usernames must be defined, and an account under which the anonymous user should run must be defined. If the necessary configurations are in place, and the *client* presents a username that is in the list of acceptable anonymous users, then the session will be accepted and the process will setuid to the anonymous user account. We do not support chroot in this version of the server.
- Username / Password: This is standard FTP authentication. It uses a separate password file, used only by the GridFTP server, *NOT* the system password file.

Warning

WE HIGHLY RECOMMEND YOU NOT USE THIS. YOU WILL BE SENDING YOUR PASSWORD IN CLEAR TEXT OVER THE NETWORK.

We do, however, have some user communities who run only on internal networks for testing purposes and who do not wish to deal with obtaining GSI credentials. If you are considering this, we would recommend that you look at Simple CA and set up your own testbed CA. This can be done in less than an hour and then provides you full GSI security.

5.3. Firewall requirements

If the GridFTP *server* is behind a firewall:

1. Contact your network administrator to open up port 2811 (for GridFTP control channel connection) and a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.
2. Set the environment variable GLOBUS_TCP_PORT_RANGE:

```
export GLOBUS_TCP_PORT_RANGE=min,max
```

where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP server to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_SOURCE_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP server to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Note

If the server is behind NAT, the `--data-interface <real ip/hostname>` option needs to be used on the server.

If the GridFTP *client* is behind a firewall:

1. Contact your network administrator to open up a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

2. Set the environment variable GLOBUS_TCP_PORT_RANGE

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP client to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP client to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Additional information on Globus Toolkit Firewall Requirements is available here[3].

6. 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.

[3] http://www.globus.org/toolkit/security/firewalls/

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 Configuration information for more information on these and other configuration options.

6.1. 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.

6.2. 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.

6.3. 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 discuss@globus.org. 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).

7. Usage statistics collection by the Globus Alliance

The following GridFTP-specific usage statistics are sent in a UDP packet at the end of each transfer, in addition to the standard header information described in the Usage Stats[4] section.

- Start time of the transfer
- End time of the transfer
- Version string of the server
- TCP buffer size used for the transfer

[4] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

- Block size used for the transfer
- Total number of bytes transferred
- Number of parallel streams used for the transfer
- Number of stripes used for the transfer
- Type of transfer (STOR, RETR, LIST)
- FTP response code -- Success or failure of the transfer

Note

The client (globus-url-copy) does NOT send any data. It is the *servers* that send the usage statistics.

We have made a concerted effort to collect only data that is not too intrusive or private and yet still provides us with information that will help improve and gauge the usage of the GridFTP server. Nevertheless, if you wish to disable this feature for GridFTP only, see the Logging section of Section 3.2, “GridFTP server configuration options”. Note that you can disable transmission of usage statistics globally for all C components by setting "GLOBUS_USAGE_OPTOUT=1" in your environment.

Also, please see our policy statement[5] on the collection of usage statistics.

[5] http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 9. GT 4.0 GridFTP : Developer's Guide

1. 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.

2. Before you begin

2.1. Feature summary

Features new in GT 4.0

- A new, complete reimplementation of the *server*.
- Support for *striping*.
- This new implementation will greatly ease new feature additions and modifications of the server (new commands, new data sources such as mass storage devices, etc.), maintainability, and resolves a licensing issue that was discovered.

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*).
- 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.

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

2.2. 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.

2.3. Backward compatibility summary

Protocol changes since GT 3.2

- None

API changes since GT 3.2

- None

Exception changes since GT 3.2

- Not Applicable (GridFTP is not Java-based)

Schema changes since GT 3.2

- Not Applicable (GridFTP is not SOAP-based)

2.4. Technology dependencies

GridFTP depends on the following GT components:

- Pre-WS Authentication / Authorization
- C Common Libraries
- XIO

GridFTP depends on the following 3rd party software:

- OpenSSL (version included in release)

2.5. Security considerations

The following are points to consider relative to security:

2.5.1. Two ways to configure your *server*

We now provide two ways to configuring your server:

- The classic installation. This is equivalent to any FTP server you would normally install. It is run as a root setuid process. Once the user is authenticated, the process does a setuid to the appropriate non-privileged user account.
- A new split process installation. In this configuration, the server consists of two processes:
  - The control channel (the process the external user connects to) runs as a non-privileged user (typically the globus user).
  - The data channel (the process that access the file system and moves the data) runs as a root setuid program as before but is only contacted by the control channel process 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.

2.5.2. New authentication options

There are new authentication options available for the server in GT4.0.0:

- Anonymous: The server now supports anonymous access. In order for this to work, a configuration switch must explicitly enable it, a list of acceptable usernames must be defined, and an account under which the anonymous user should run must be defined. If the necessary configurations are in place, and the *client* presents a username that is in the list of acceptable anonymous users, then the session will be accepted and the process will setuid to the anonymous user account. We do not support chroot in this version of the server.
- Username / Password: This is standard FTP authentication. It uses a separate password file, used only by the GridFTP server, *NOT* the system password file.

Warning

WE HIGHLY RECOMMEND YOU NOT USE THIS. YOU WILL BE SENDING YOUR PASSWORD IN CLEAR TEXT OVER THE NETWORK.

We do, however, have some user communities who run only on internal networks for testing purposes and who do not wish to deal with obtaining GSI credentials. If you are considering this, we would recommend that you look at Simple CA and set up your own testbed CA. This can be done in less than an hour and then provides you full GSI security.

2.5.3. Firewall requirements

If the GridFTP *server* is behind a firewall:

1. Contact your network administrator to open up port 2811 (for GridFTP control channel connection) and a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

2. Set the environment variable GLOBUS_TCP_PORT_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP server to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_SOURCE_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP server to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Note

If the server is behind NAT, the `--data-interface <real ip/hostname>` option needs to be used on the server.

If the GridFTP *client* is behind a firewall:

1. Contact your network administrator to open up a range of ports (for GridFTP data channel connections) for the incoming connections. If the firewall blocks the outgoing connections, open up a range of ports for outgoing connections as well.

2. Set the environment variable GLOBUS_TCP_PORT_RANGE

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the incoming connections on the firewall. This restricts the listening ports of the GridFTP client to this range. Recommended range is 1000 (e.g., 50000-51000) but it really depends on how much use you expect.

3. If you have a firewall blocking the outgoing connections and you have opened a range of ports, set the environment variable GLOBUS_TCP_SOURCE_RANGE:

   ```
   export GLOBUS_TCP_PORT_RANGE=min,max
   ```

   where min,max specify the port range that you have opened for the outgoing connections on the firewall. This restricts the outbound ports of the GridFTP client to this range. Recommended range is twice the range used for GLOBUS_TCP_PORT_RANGE, because if parallel TCP streams are used for transfers, the listening port would remain the same for each connection but the connecting port would be different for each connection.

Additional information on Globus Toolkit Firewall Requirements is available here[1].

3. 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.

3.1. 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.

3.2. 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.

[1] http://www.globus.org/toolkit/security/firewalls/

3.3. GridFTP Server

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 GT4.0.x 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).

3.3.1. 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.

3.3.2. 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.

3.3.3. 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.

3.3.4. 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.

3.3.5. 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.

4. Public interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in Chapter 11, *GT 4.0 Component Guide to Public Interfaces: GridFTP*.

5. Usage scenarios

There is no content available at this time.

6. Tutorials

There is no content available at this time.

7. Debugging

There is no content available at this time.

8. Troubleshooting

There is no content available at this time.

9. Related Documentation

- The Globus Striped GridFTP Framework and Server[2]

[2] http://www.globus.org/alliance/publications/papers/gridftp_final.pdf

Chapter 10. GT 4.0 Component Fact Sheet: GridFTP

1. Brief component overview

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. We have selected a set of protocol features and extensions defined already in IETF RFCs and added a few additional features to meet requirements from current data grid projects.

2. Summary of features

Features new in GT 4.0

- A new, complete reimplementation of the *server*.
- Support for *striping*.
- This new implementation will greatly ease new feature additions and modifications of the server (new commands, new data sources such as mass storage devices, etc.), maintainability, and resolves a licensing issue that was discovered.

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*).
- 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.

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

3. Usability summary

The following areas have received substantial effort in order to make using GridFTP easier, more enjoyable, etc:

- 100% protocol compatibility with previous versions. Any *client*/*server* combination since 2.4.0 should work (barring a new client trying to invoke new functionality on an old server).
- Greatly improved documentation. It is not complete, and we will continue to develop it, but it is far better than past versions.
- For GT 4.0.0 there is a completely new server implementation that provides the following benefits:
  - More stable (one project went from on the order of 1000 failures in a large workflow with the 2.4.3 server to zero failures with the new server).
  - More scalable (we had a single server supporting 1800 clients).
  - Easier to extend and modify. The Data Storage Interface (DSI) allows a clean way to access things other than POSIX file systems, such as tape systems, custom storage systems, etc.
  - Ability to run threaded servers (previous versions of the server did not support this).
  - *striping* is now available (allows multi-Gigabit speeds).
  - Easier server configuration.

4. Backward compatibility summary

Protocol changes since GT 3.2

- None

API changes since GT 3.2

- None

Exception changes since GT 3.2

- Not Applicable (GridFTP is not Java-based)

Schema changes since GT 3.2

- Not Applicable (GridFTP is not SOAP-based)

5. Technology dependencies

GridFTP depends on the following GT components:

- Pre-WS Authentication / Authorization
- C Common Libraries
- XIO

GridFTP depends on the following 3rd party software:

- OpenSSL (version included in release)

6. 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.

7. Associated standards

Associated standards for GridFTP:

- RFC 959 Base FTP protocol[1]
- RFC 2228 gssapi security extensions[2] for FTP RFC 2389 FEAT, OPTS, etc.

[1] http://rfc.net/rfc959.html

[2] http://rfc.net/rfc2228.html

- extensions to FTP (IETF FTP Working group draft)[3] for structured directory listings, SIZE, MDTM commands.
- GFD.020 GridFTP extensions[4]

8. For More Information

Click here[5] for more information about this component.

[3] http://www.ietf.org/internet-drafts/draft-ietf-ftpext-mlst-16.txt
[4] http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf
[5] index.html

Chapter 11. GT 4.0 Component Guide to Public Interfaces: GridFTP

1. Semantics and syntax of APIs

1.1. 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 Grid FTP Protocol document.

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

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

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

1.2. Component API

- C Client Library API[1]
- C Control Library API[2]

For information on the internationalization API, see the C Common Libraries Public Interface[3].

[1] http://www.globus.org/api/c-globus-3.9.x/globus_ftp_client/html/index.html
[2] http://www.globus.org/api/c-globus-3.9.x/globus_ftp_control/html/index.html
[3] http://www.globus.org/toolkit/docs/4.0/common/ccommonlib/C_Common_Libraries_Public_Interfaces.html#apis

2. Semantics and syntax of the WSDL

GridFTP has no WSDL as it is not Web Service based at this time.

3. Command-line tools

Please see the GridFTP Command Reference.

4. Overview of Graphical User Interface

Globus does not provide any interactive client for GridFTP, either GUI or text based. However, NCSA, as part of there TeraGrid activity, produces a text based interactive client called UberFTP, which you may want to check out. See Interactive Clients[4] for more information.

5. Semantics and syntax of domain-specific interface

5.1. Interface introduction

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.

5.2. Syntax of the interface

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.

6. Configuration interface

Please see Section 3.2, “GridFTP server configuration options” in the System's Administrator Guide.

7. Environment variable interface

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.

[4] rn01re01.html#interactiveclients

- Pre-WS Authentication & Authorization Environment Variables[5]
- XIO Network Driver Environment Variables[6]

[5] http://www.globus.org/toolkit/docs/4.0/security/prewsaa/Pre_WS_AA_Public_Interfaces.html#prewsaa-env

[6] http://www.globus.org/toolkit/docs/4.0/common/xio/XIO_Public_Interfaces.html#s-xio-env

Chapter 12. GT 4.0 GridFTP: Quality Profile

1. Test coverage reports

There are no test coverage reports available at this time.

2. Code analysis reports

There are no code analysis reports available at this time.

3. Outstanding bugs

- Bug 2547[1]: gridftp config detach option doesn't work

4. Bug Fixes

- Bug 1883[2]: globus_ftp_control server does not decode ADAT properly.
- Bug 1928[3]: Strange problems: Busy wait and address in use.
- Bug 2036[4]: New gridftp, globus-gridftp-server, does not use globus_gss_assist_map_and_authorize
- Bug 2128[5]: gridftp server checks ownership of hostcert after setuid
- Bug 2129[6]: Gridftp logging levels (is 7>10?)
- Bug 2132[7]: Server responds with "Valid credentials could not be found..."
- Bug 2452[8]: non-striped gridFTP not working w/ round robin
- Bug 2463[9]: striped server, small partial transfers hang
- Bug 2464[10]: striped server to non-striped server doesn't work

5. Performance reports

- Performance of Globus Striped GridFTP Server on TeraGrid[11]

[1] http://bugzilla.globus.org/globus/show_bug.cgi?id=2547
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=1883
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=1928
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=2036
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=2128
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=2129
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=2132
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=2452
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=2463
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=2464
[11] gridftp_performance.doc

- GridFTP Scalability and Performance Results[12]

[12] gridftp_scalability.doc

Chapter 13. GT 4.0.x Migrating Guide for GridFTP

The following provides available information about migrating from previous versions of the Globus Toolkit.

1. Migrating from GT2

If you are running a version 2.2 or earlier, it is deprecated, unsupported, has major bugs leading to stability problems, has known potential security exploits via the wuftpd server[1], and has a protocol incompatibility with later versions due to an error in the security code. Your GT 2.2 clients[2] will not work with newer servers (GT 2.4.0 and greater) and new clients will not work with GT 2.2 servers. You should immediately upgrade to GT 4.0.0.

If you are running GT 2.4.0 or greater, the migration for GridFTP is relatively painless. There were only new features added. No changes were made to the existing protocol or APIs, so any existing client or application built using our client APIs will work unchanged. You may install new clients and servers on an as-opportunity-permits basis and will have no problems. Any combination of old/new client/server will work.

To upgrade your server, either install it in a $GLOBUS_LOCATION different than the GT 2 installation (either as part of an entire GT 4 installation or by just doing `make gridftp`). Alternately, you can statically link the new server to avoid versioning issues and replace the existing executable. The configuration files are very different, so you will need to update the configuration.

To upgrade your client, simply build the client and use the new client as you would the old one.

Below is a list of new functionality available in GT 3.2 and higher (note that the MLST/MLSD feature is used by RFT in GT 3.2 and higher and is required if you want to be able to specify a directory to move):

New Functionality in 3.2:

- Server Improvements
  - Structured File Info
  - MLST, MLSD
  - checksum support
  - chmod support
- globus-url-copy changes
  - File globbing support
  - Recursive dir moves
  - RFC 1738 support
  - Control of restart
  - Control of DC security

[1] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Glossary.html#server
[2] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Glossary.html#client

2. Migrating from GT3

If you are running GT 3.0.0 or greater, the migration for GridFTP is relatively painless. There were only new features added. No changes were made to the existing protocol or APIs, so any existing client[3] or application built using our client APIs will work unchanged. You may install new clients and servers on an as-opportunity-permits basis and will have no problems. Any combination of old/new client/server will work.

To upgrade your server, either install it in a $GLOBUS_LOCATION different than the GT 3 installation (either as part of an entire GT 4 installation or by doing just `make gridftp`). Alternately, you can statically link the new server to avoid versioning issues and replace the existing executable. The configuration files are very different, so you will need to update the configuration.

To upgrade your client, simply build the client and use the new client as you would the old one.

Below is a list of new functionality available in GT 3.2 and higher (note that the MLST/MLSD feature is used by RFT in GT 3.2 and higher and is required if you want to be able to specify a directory to move):

New Functionality in 3.2:

- Server Improvements
  - Structured File Info
  - MLST, MLSD
  - checksum support
  - chmod support
- globus-url-copy changes
  - File globbing support
  - Recursive dir moves
  - RFC 1738 support
  - Control of restart
  - Control of DC security

[3] http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_Glossary.html#client

GT 4.0: GridFTP Command Reference

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

Important

YOU MUST HAVE A CERTIFICATE TO USE globus-url-copy!

Table 12. Prerequisites to using globus-url-copy

1	First, as with all things Grid, you *must* have a valid proxy certificate to run globus-url-copy. If you do not have a certificate, you must obtain one. If you are doing this for testing in your own environment, the Simple CA[1] provided with the Globus Toolkit should suffice. If not, you must contact the Virtual Organization (VO) with which you are associated to see from whom you should request a certificate. One common source is the DOE Science Grid CA, 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.
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[2].
3	You are now ready to use **globus-url-copy**! See the following sections for syntax and command line options.

Command syntax

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

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

where:

[1] http://www.globus.org/toolkit/docs/4.0/admin/docbook/apa.html
[2] http://www.globus.org/toolkit/docs/4.0/security/prewsaa/rn01re05.html

[optional command line switches]	See Command line options[3] below for a list of available options.
<sourceURL>	Specifies the original URL of the file(s) to be copied. If this is a directory, all files within that directory will be copied.
<destURL>	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://**
- **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 Interactive Client[4] 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.

[3] #commandlineoptions

[4] #interactiveclients

Table 13. URL formats

gsiftp://myhost.mydomain.com:2812/data/foo.dat	Fully specified.
http://myhost.mydomain.com/mywebpage/default.html	Port not specified so uses protocol default, 80 in this case.
file:///foo.dat	Host not specified so it uses your local host, port not specified as before.
file:/foo.dat	This is also valid but is not recommended because, while many servers (including ours) accept this format, it is *not* 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

Table 14. 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: • 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.

Utility Ease of Use Options

Table 15. 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 <filename>	Reads a list of URL pairs from a filename. Each line should contain: `<sourceURL> <destURL>` Enclose URLs with spaces in double quotes ("). Blank lines and lines beginning with # will be ignored.
-r | -recurse	Copies files in subdirectories **Note** If you specify a URL directory and do not add this option, only the files in the stated directory will be moved.
-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 *third party transfer*). 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

Table 16. 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

Table 17. 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. 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. 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.

Security Related Options

Table 18. 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).
-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).
-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).
-nodcau | -no-data-channel-authentication	Turns off data channel authentication for FTP transfers (the default is to authenticate the data channel). 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. 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. VERY rarely used due to the VERY substantial performance penalty.

Notes about globus-url-copy

- A **globus-url-copy** invocation using the **gsiftp** protocol with no options (using all the defaults) will do a binary, stream mode (which implies no parallelism) transfer with the host default TCP buffer size, an encrypted and checksummed control channel, and an authenticated data channel.

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

Most normal FTP servers only implement *stream mode*, 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,* you will need to explicitly specify the subject name to expect. You can use the **-ss** flag to set the sourceURL subject, and **-ds** to set the destURL subject. If you use **-s** alone, it will set both to be the same. You can see an example of this usage under the Verification section of this guide. *Please note:* This is an *unusual* use of the client. Most times you only need to specify both URLs.

How do I choose a value?

How do I choose a value for the TCP buffer size (-tcp) 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 = bandwidth in Megabits per second (Mbs) * 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 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[5] 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.

[5] user-index.html#s-gridftp-user-thirdpartytransfers

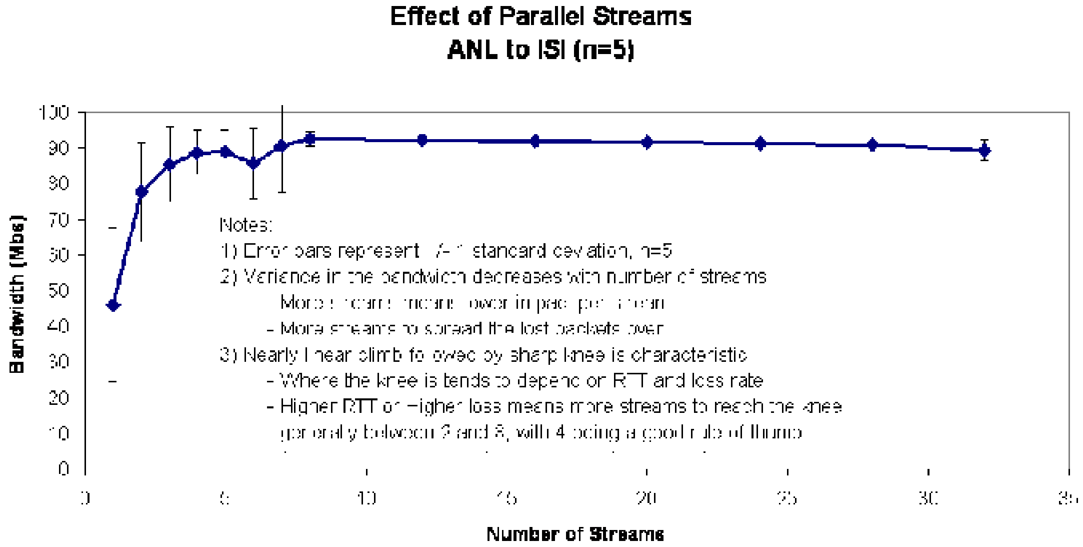
So where does this formula come from? Because it uses the bandwidth and the RTT (also known as the latency or delay) it is called the *bandwidth delay product*. The very simple explanation is this: TCP is a reliable protocol. It must save a copy of everything it sends out over the network until the other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



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 4.0.

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.

was developed at the NCSA under the auspices of NMI and TeraGrid. It is available through NMI (also a convenient place to get Globus and other tools), or directly from NCSA:

- NMI Download: http://www.nsf-middleware.org/
- NCSA Uberftp only download: http://dims.ncsa.uiuc.edu/set/uberftp/download/index.html [6]
- UberFTP User's Guide: http://teragrid.ncsa.uiuc.edu/Doc/Data/uberftp.html [7]

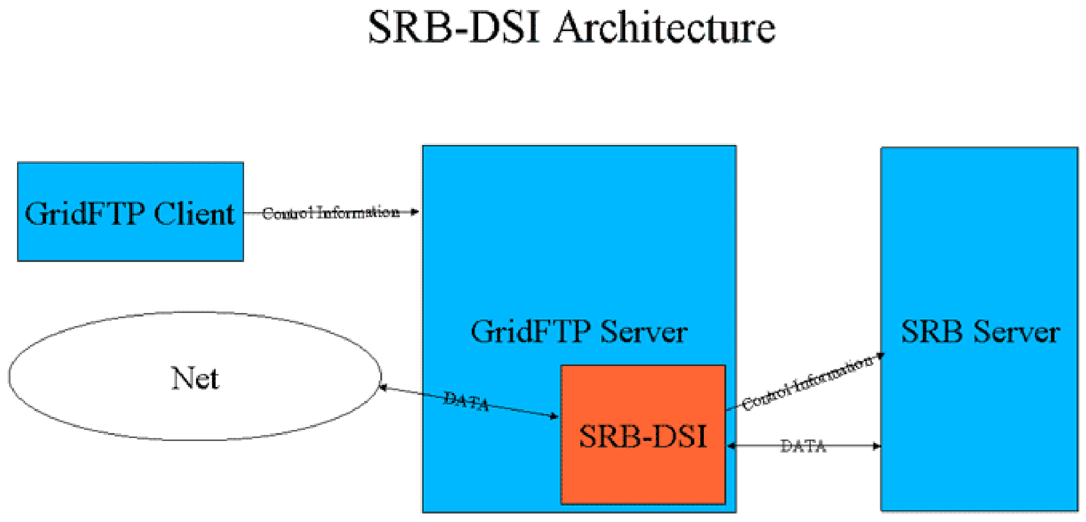
[6] http://dims.ncsa.uiuc.edu/set/uberftp/download/index.html
[7] http://teragrid.ncsa.uiuc.edu/Doc/Data/uberftp.html

Chapter 14. GT 4.0 GridFTP: Storage Resource Broker (SRB)

1. Introduction

The Storage Resource Broker Data Storage Interface (SRB-DSI) is an extension to the GridFTP server that allows it to interact with SRB. Plugging this extension into a GridFTP server allows the GridFTP server to access a SRB resource and serve it to any GridFTP client as though it were a filesystem.

2. Architecture



The above image shows the architecture of the system. Their are 4 major components:

- *SRB Server* - This is where the data is stored. It is accessed by the GridFTP server via the standard SRB protocols using GSI_AUTH.
- *SRB-DSI* - This component is the bridge between GridFTP and SRB. All operation requests and data are routed through this component. The GridFTP server makes requests of it, then it translates these requests into SRB client commands.
- *GridFTP Server* - A standard GridFTP 4.0.1 server is loaded with the SRB-DSI. Clients contact this server to access data in a SRB resource. The server passes the request to the SRB-DSI which, as described above, passes the request on to the SRB server. The responses to the requests return along the same path.
- *GridFTP Client* - A stock GridFTP client (like globus-url-copy). No modifications to the client are needed.

3. Software

You need the following items to use the SRB-DSI:

- *Globus 4.0.1* - You need the GridFTP distributed with globus 4.0.1 or later. You can find that here[1].
- *SRB Client 3.4.0* - You only need the client libraries to build the SRB-DSI, but you will need access to a running SRB server and resource. You can find the client libraries here[2].
- *SRB-DSI* - You can find the SRB-DSI here[3].

4. Building

Instructions for building Globus[4] and SRB[5] are well documented in the above links. The following sections describe one way of building these two packages. However, if any questions or errors are discovered, the reader should look to the above links for solutions.

4.1. Building Globus

Download the source installer, choose a path on your filesystem for your GLOBUS_LOCATION, and run the following:

```
% bunzip2 gt4.0.1-all-source-installer.tar.bz2
% tar -xvf gt4.0.1-all-source-installer.tar
% export GLOBUS_LOCATION=<path you chose for your GLOBUS_LOCATION>
% ./configure --prefix=$GLOBUS_LOCATION
% make gridftp globus_gridftp_server-thr
% source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

4.2. Building SRB

```
% ./configure  --enable-gsi-auth --enable-globus-location=$GLOBUS_LOCATION --enable-globus
% make
```

4.3. Building SRB-DSI

The SRB-DSI is a GPT package. More information about GPT package installation can be found here[6]. Most users should simply need:

```
gpt-build -force CONFIGOPTS_GPTMACRO="--with-srb-path=<location of SRB source tree>"  glob
```

5. Administration

Before you can run the GridFTP server with the SRB-DSI, you need to set up some files.

[1] http://www.globus.org/toolkit/downloads/4.0.1/
[2] http://www.sdsc.edu/srb/tarfiles/main.html
[3] http://www-unix.mcs.anl.gov/~bresnaha/SRB_DSI_Doc/globus_srb_dsi-latest.tar.gz
[4] http://www.globus.org/toolkit//
[5] http://www.sdsc.edu/srb
[6] http://www.gridpackagingtools.org/

5.1. Creating and setting up the SRB configuration file

A configuration file must be created at:

```
$GLOBUS_LOCATION/etc/gridftp_srb.conf
```

The following values must be set in this file:

```
srb_hostname <host>:<port>
srb_hostname_dn <domain name to expect from SRB server>
srb_default_resource <default srb resource to use>
```

5.2. Setting up the gridmap file

Additionally, the gridmap file must be special for this DSI. Along with the subject name and username, the SRB-DSI needs to know the SRB domain name for the user. This is handled by adding an additional value to the gridmap file:

```
"<user security DN>" <srb user name>@<domain name>
```

6. Running

Once you have the configuration files in place, you can run the server.

Important

All options of the server apply, but the parameter `-dsi srb -auth-level 4` *must* also be used.

For more information on setting these values and running the GridFTP server see Configuring GridFTP.

Most users can run with:

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -p <port> -dsi srb -auth-level 4
```

7. See Also

See the README[7] file for more information.

[7] http://www-unix.mcs.anl.gov/~bresnaha/SRB_DSI_Doc/README.txt

Index

GT 4.0 GridFTP Glossary

C

client

FTP is a command/response protocol. The defining characteristic of a client is that it is the process sending the commands and receiving the responses. It may or may not take part in the actual movement of data.
See Also client/server transfer, third party transfers.

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).
See Also extended block mode (MODE E).

command/response

Both FTP and GridFTP are command/response protocols. What this means is that once a client sends a command to the server, it can only accept responses from the server until it receives a response indicating that the server is finished with that command. For most commands this is not a big deal. For instance, setting the type of the file transfer to binary (called "I" for "image in the protocol"), simply consists of the client sending TYPE I and the server responding with 220 OK. Type set to I. However, the SEND and RETR commands (which actually initiate the movement of data) can run for a long time. Once the command is sent, the client’s only options are to wait until it receives the completion reply, or kill the transfer.

concurrency

When speaking of GridFTP transfers, concurrency refers to having multiple files in transit at the same time. They may all be on the same host or across multiple hosts. This is equivalent to starting up “n” different clients for “n” different files, and having them all running at the same time. This can be effective if you have many small files to move. The Reliable File Transfer (RFT)[8] service utilizes concurrency to improve its performance.

[8] http://www.globus.org/toolkit/docs/4.0/data/rft/

D

dual channel protocol

GridFTP uses two channels:

- One of the channels, called the *control channel*, is used for sending commands and responses. It is low bandwidth and is encrypted for security reasons.
- The second channel is known as the *data channel*. Its sole purpose is to transfer the data. It is high bandwidth and uses an efficient protocol.

By default, the data channel is authenticated at connection time, but no integrity checking or encryption is performed due to performance reasons. Integrity checking and encryption are both available via the client and libraries.

Note that in GridFTP (not FTP) the data channel may actually consist of several TCP streams from multiple hosts.

See Also parallelism, striping.

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. [TODO: LINK TO GRAPHIC]

See Also parallelism, striping.

I

improved extended block mode (MODE X)

This protocol is still under development. It is intended to address a number of the deficiencies found in MODE E. For instance, it will have explicit negotiation for use of a data channel, thus removing the race condition and the requirement for the sender to be the connector. This will help with firewall traversal. A method will be added to allow the filename to be provided prior to the data channel connection being established to help large data farms better allocate resources. Other additions under consideration include block checksumming, resends of blocks that fail checksums, and inclusion of a transfer ID to allow pipelining and de-multiplexing of commands.

See Also extended block mode (MODE E).

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), improved extended block mode (MODE X), stream mode (MODE S).

N

network end points

A network endpoint is generally something that has an IP address (a network interface card). It is a point of access to the network for transmission or reception of data. Note that a single host could have multiple network end points if it haS multiple NICs installed (multi-homed). This definition is necessary to differentiate between parallelism and striping.
See Also parallelism, striping.

P

parallelism

When speaking about GridFTP transfers, parallelism refers to having multiple TCP connections between a single pair of network endpoints. This is used to improve performance of transfers on connections with light to moderate packet loss.

S

server

The compliment to the client is the server. Its defining characteristic is that it 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 http://www.globus.org/toolkit/docs/4.0/data/gridftp/developer-index.htmls-gridftp-developer-archdes in the GridFTP Developer's Guide.
See Also client.

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.

striping

When speaking about GridFTP transfers, striping refers to having multiple network endpoints at the source, destination, or both participating in the transfer of the same file. This is normally accomplished by having a cluster with a parallel shared file system. Each node in the cluster reads a section of the file and sends it over the network. This mode of transfer is necessary if you wish to transfer a single file faster than a single host is capable of. This also tends to only be effective for large files, though how large depends on how many hosts and how fast the end-to-end transfer is. Note that while it is theoretically possible to use NFS for the shared file system, your performance will be poor, and would make using striping pointless.

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 supercomputer 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 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.