

GT 4.2.1 GridFTP : System Administrator's Guide

GT 4.2.1 GridFTP : System Administrator's Guide

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 [Installing GT 4.2.1](#). Read through this guide before continuing!

Table of Contents

1. Building and Installing	1
1. Building only GridFTP and Utilities	1
2. Building only the GridFTP server	1
3. Building only the GridFTP client	1
4. Building only the GridFTP SDK	2
5. Building a combination of GridFTP elements	2
6. Building and Installing a static GridFTP server	2
7. Switching between threaded and non-threaded flavors	3
2. Configuring GridFTP	4
1. GridFTP server configuration overview	4
2. Types of configurations	4
3. globus-gridftp-server quickstart	5
4. Running in daemon mode	6
5. Running under inetd or xinetd	6
3. Configuring Security for GridFTP	8
1. Security Considerations	8
2. Anonymous mode	9
3. Username/password	10
4. SSHFTP (GridFTP-over-SSH)	10
5. GSIFTP	11
6. User permissions	11
4. Advanced Configuration	13
1. Remote data-nodes and striped operation	13
2. Separation of Processes (Split Process)	13
3. Plugging in a Data Storage Interface (DSI)	14
4. Configuring GridFTP to run with the Community Authorization Service (CAS)	14
5. Configuring GridFTP to use UDT instead of TCP	14
6. Running with GFork Master Plugin	15
7. Configuring multicasting/broadcasting	15
8. Configuring GridFTP to enable Netlogger's bottleneck detection	16
5. Key Admin Settings and Tuning Recommendations	17
1. Concurrent Instances	17
2. Disk Block Size	18
3. System TCP Buffer Settings	18
4. Data Interface	19
6. Testing	20
7. Debugging	21
1. Logging	21
8. Troubleshooting	22
1. Error Codes in GridFTP	23
2. Establish control channel connection	23
3. Try running globus-url-copy	24
4. If your server starts... ..	25
5. High latency for GridFTP server connections	25
9. Usage statistics collection by the Globus Alliance	26
1. GridFTP-specific usage statistics	26
A. GridFTP Admin Tool	27
globus-gridftp-server	28
B. Setting up SRB for use with GridFTP	38
1. Introduction	38
2. Architecture	38

3. Software	38
4. Building	38
5. Administration	39
6. Running	40
7. See Also	40
C. Running the Globus GridFTP Server With GFork	41
1. Introduction	41
2. Use Cases	41
3. Configuration	41
4. GFork and Striped Servers	43
5. GFork with Memory Management	44
Glossary	46
Index	48

List of Tables

8.1. GridFTP Errors	23
---------------------------	----

Chapter 1. Building and Installing

GridFTP is built and installed as part of a default GT 4.2.1 installation. For basic installation instructions, see [Installing GT 4.2.1](#). No extra installation steps are required for this component.

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 4.2.1 section of the System Administrator's Guide](#). 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. Building only the GridFTP server

If you wish to install only the GridFTP server, refer to [the Installing GT 4.2.1 section of the System Administrator's Guide](#). Follow steps 1-3 as written. However, instead of running "make" as directed in step 4,

Run:

```
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. Building only the GridFTP client

If you wish to install only the GridFTP client, refer to [the Installing GT 4.2.1 section of the System Administrator's Guide](#). 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**.

4. Building only the GridFTP SDK

If you wish to install only the GridFTP SDK, refer to the [the Installing GT 4.2.1 section of the System Administrator's Guide](#). 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**.

5. Building a combination of GridFTP elements

If you wish to build a combination of GridFTP elements, refer to the [the Installing GT 4.2.1 section of the System Administrator's Guide](#). 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
```

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**.

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 4.2.1 section of the System Administrator's Guide](#). 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`.

7. Switching between threaded and non-threaded flavors

Starting with Globus 4.2.0, threaded flavor is the default flavor for the GridFTP server. Non-threaded flavor is still the default flavor for the client.

7.1. If you are building from the source installer

If you are building from a source installer, both threaded and non-threaded flavors of server and client will be built by default. By default, the server executable in `$GLOBUS_LOCATION/sbin` is threaded and the client executable in `$GLOBUS_LOCATION/bin` is non-threaded.

If you built from the source installer, you will find non-default flavors of both the server and client binaries in `$GLOBUS_LOCATION/sbin/[flavor]/shared` and `$GLOBUS_LOCATION/bin/[flavor]/shared`. To use those flavors by default, simply copy those binaries to `$GLOBUS_LOCATION/sbin` or `$GLOBUS_LOCATION/bin`.

7.1.1. Examples

Changing **globus-url-copy** from non-threaded to threaded:

```
$ cp $GLOBUS_LOCATION/bin/gcc32dbgpthr/shared/globus-url-copy $GLOBUS_LOCATION/bin
```

Changing **globus-gridftp-server** from threaded to non-threaded:

```
$ cp $GLOBUS_LOCATION/sbin/gcc32dbg/shared/globus-gridftp-server $GLOBUS_LOCATION/sbin
```

Checking whether a binary is a threaded flavor or not with the tool **ldd** (no output means non-threaded):

```
$ ldd $GLOBUS_LOCATION/sbin/globus-gridftp-server |grep thread  
libpthread.so.0 => /lib/libpthread.so.0 (0x00cab000)
```

7.2. If you are building from the binary installer

If you are using a binary installer, it will only have a threaded GridFTP server and non-threaded client. If you want a different flavor of server/client, use source installer.

Chapter 2. Configuring GridFTP

1. GridFTP server configuration overview

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



Note

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

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

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

Options are one per line, with the format:

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

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

For example:

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

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

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

2. Types of configurations

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

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

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

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

Note

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

3. globus-gridftp-server quickstart

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

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

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

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

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

where:

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

Run a two party transfer with client:

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

Run 3rd party transfer:

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

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

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

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

where:

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

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

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

`Ctrl-c` - Kill the server.



Note

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

4. Running in daemon mode

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

Run the server:

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

where:

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

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

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

5. Running under inetd or xinetd



Note

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

5.1. Set up xinetd/inetd config file



Note

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

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

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

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

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

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



Note

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

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



Note

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

5.2. globus-gridftp-server -i

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

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

Chapter 3. Configuring Security for GridFTP

There are many security options in GridFTP ranging from no security to higher security via GSI .

1. Security Considerations

1.1. Ways to configure your server

As discussed in [Section 2, “Types of configurations”](#), there are three ways to configure your GridFTP server: the default configuration (like any normal FTP server), separate (split) process configuration and striped configuration. The latter two provide greater levels of security as described [here](#).

1.2. New authentication option

There is a new authentication option available for GridFTP in GT 4.2.1:

- **SSH Authentication** Globus GridFTP now supports SSH based authentication for the control channel. In order for this to work:

- Configure server to support SSH authentication,

Configure client(globus-url-copy) to support SSH authentication,

Use sshftp:// urls in globus-url-copy

For more information, see [Section 4, “SSHFTP \(GridFTP-over-SSH\)”](#).

1.3. 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.

2. Anonymous mode

Anonymous mode (using the `-aa` option) allows any user with an FTP client to read and write (and delete) files that the server process can similarly access (it is also a quick way to test that your server works).

```
% globus-gridftp-server -aa
    Server listening at 127.0.0.1:58806
```



Warning

When the server is run in this way, anyone who can connect to the server will possess all the same rights as the user that the process is run as (directly or via `-anonymous-user`). If using this mode intentionally for open access, it is best to run under a dedicated account with limited filesystem permissions. You can also use the option below to disable FTP commands such as STOR, ESTO, DELE, RDEL, RNTO, etc to make sure that users can only read from the server and not write to it.

```
-disable-command-list <string>
```

Where `<string>` represents a comma separated list of client commands that will be disabled. Default: not set.

¹ <http://www.globus.org/toolkit/security/firewalls/>

3. Username/password

If you trust your network and want a minimal amount of security, you can run the `globus-gridftp-server` with clear text passwords. This security model is the one originally introduced in RFC959.

Warning

We do not recommend it for long running servers open to the internet.

3.1. Create password file

To run the server in clear text password mode, we first need to create a password file dedicated to it. The format of the password file is the same as standard system password files; however, it is ill-advised to use a system password file. To create an entry in a GridFTP password file, run the following commands:

```
% touch pwfile
% gridftp-password.pl >> pwfile
Password:
```

This will ask you for a password and then create an entry in the password file for the current user name and the given password. Take a look at the file created. You will notice that the password you typed in is not in the file in a clear text form. We have run it though a one way hash algorithm before storing it in the file.

3.2. Run the server in password mode

Simply start the server pointing it at the password file you just created.

```
% globus-gridftp-server -password-file /full/path/of/pwfile
Server listening at 127.0.0.1:5555
```

3.3. Make a transfer

To run `globus-url-copy` with the password, use the following syntax:

```
globus-url-copy file:///etc/group ftp://username:pw@localhost:5000/tmp/group
```

4. SSHFTP (GridFTP-over-SSH)

This type of security introduces the `sshftp` control channel (frontend) protocol. This is a very simple means of obtaining strong security on the control channel only (the data channel is *not* authenticated). With this approach, you can run a GridFTP transfer anywhere that you can `ssh`. `sshftp://` leverages the ubiquitous `ssh/ssh` programs to form control channel connections much in the same way that `inetd` forms connections.

4.1. Configure Client-Side `sshftp://`

Every `$GLOBUS_LOCATION` must be configured for client-side `sshftp://` connections. In other words, if we wish to use `globus-url-copy` with `sshftp://` URLs, we must first configure the `$GLOBUS_LOCATION` that contains `globus-url-copy` in the following way:

```
% $GLOBUS_LOCATION/setup/globus/setup-globus-gridftp-sshftp
```

4.2. Configure Server Side `sshftp://`

Every host that wishes to run a **globus-gridftp-server** which can accept `sshftp://` connections must run the following command as root:

```
% $GLOBUS_LOCATION/setup/globus/setup-globus-gridftp-sshftp -server
```

In the absence of root access, a user can configure the server to allow `sshftp://` connections for that user only with the following command:

```
% $GLOBUS_LOCATION/setup/globus/setup-globus-gridftp-sshftp -server -nonroot
```

4.3. Performing `sshftp://` Transfers

In this case, a **globus-gridftp-server** does not need to be running. The server will be started via the `sshd` program. Therefore, the hostname and port should be that of the `sshd` server. Run **globus-url-copy** just as you have before; simply change `ftp://` to `sshftp://`.

```
% globus-url-copy -v file:/etc/group sshftp://127.0.0.1/tmp/group % globus-url-copy -list
```

5. GSIFTP

This security option can be the most involved to set up, but provides the most security. It requires setting up GSI security as described in the GT Installation Guide here: [Basic Security Configuration](#).

Once GSI has been set up (host and user credentials are valid, the gridmap file is updated and you've run `grid-proxy-init` to create a proxy certificate), you simply run the GridFTP server:

```
globus-gridftp-server
```



Note

If run as `root`, it will pick up the host cert; if not, it will pick up the user cert.

Now you are ready to perform a GSI-authenticated transfer:

```
globus-url-copy <-s subject> src_url dst_url
```



Note

The `subject` option is only needed if the server was not started as `root`.

6. User permissions

Users are mapped to a local account on the server machine and file permissions are handled by the operating systems. In the anonymous mode, users that connect to the server will possess all the same rights as the user that the server process is run as (directly or via `-anonymous-user`).

In case of username/password authentication, the users are mapped to the `uid` corresponding to the username in the GridFTP password file and the access permissions for the users is same as that of the `UID` that they are mapped to. If SSH based authentication is used, upon successful authentication, `SSHD` maps users to a local account and the GridFTP server is run as the mapped local user. The access permissions are the same as that of the mapped local user.

If GSI is used, upon successful authentication an authorization callout is invoked to (a) verify authorization and (b) determine the local user id as which the request should be executed. This callout is linked dynamically. Globus GridFTP provides an implementation that supports both a Globus "gridmapfile" and Community Authorization Service credentials, which may encode in SAML assertions the specific files that a user is authorized to read and/or write. Sites can also provide alternative implementations. Server does a setuid to the local user id as determined by the authorization callout and the access permissions are the same as that of the local user id.

GridFTP server provides an option to disable certain FTP commands:

```
-disable-command-list <string>
```

Where <string> represents a comma separated list of client commands that will be disabled. Default: not set.

Chapter 4. Advanced Configuration

It is assumed that the toolkit installation was successful. For more information, see the [Installing GT 4.2.1](#). Also be sure to reference [Chapter 2, Configuring GridFTP](#) and [globus-gridftp-server](#).

1. 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, run:

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

To start the data-node, run:

```
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.

2. Separation of Processes (Split Process)

As 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 process related to an authenticated user. This allows the server to leverage the operating system's 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 process 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, run:

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

To start the back end, run:

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

3. Plugging in a Data Storage Interface (DSI)

FIXME GridFTP can be used as a network interface to existing Data Storage Interfaces (DSIs) using the `-dsi` option. With this option the DSI plugs into the backend (compatible with striping) and is transparent to the client or remote party. It can be used with either the GT standard DSI plugins or with [custom-built DSI plugins](#). The standard DSI plugins available in a default GT installation are:

- [Storage Resource Broker \(SRB\)](#)

The above links point to complete information about setting up and running the GridFTP server with these DSIs.

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

The [Community Authorization Service \(CAS\)](#) 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](#).

5. Configuring GridFTP to use UDT instead of TCP

UDT is bundled with Globus starting with Globus v4.2, so downloading UDT separately is no longer needed.

5.1. Prerequisites

1. Threaded build of the Globus GridFTP server. Note that starting with Globus v4.2, the default flavor of the server is threaded.
2. For client-server transfers, threaded build of `globus-url-copy`. For third-party (server-server) transfers, threaded build of `globus-url-copy` is not needed. Refer to [Section 7, “Switching between threaded and non-threaded flavors”](#) for information on how to switch between threaded and non-threaded flavors of `globus-url-copy`.

5.2. Steps

1. Build and install UDT

```
globus$ make udt ("make gridftp udt" if gridftp is not built and installed already)
```

```
globus$ make install
```

2. Configure GridFTP server

If you the GridFTP server from xinetd, add '-dc-whitelist udt,gsi,tcp' to 'server_args' in /etc/xinetd.d/gsiftp

Alternatively, you can use the file \$GLOBUS_LOCATION/etc/gridftp.conf to configure this. Add the following to that file:

```
dc-whitelist udt,gsi,tcp
```

If you run the server from commandline:

```
$GLOBUS_LOCATION/sbin/globus-gridftp-server -dc-whitelist  
udt,gsi,tcp
```

3. Run globus-url-copy with new command line option "-udt"

6. Running with GFork Master Plugin

GFork is a service like inetd that listens on a TCP port and runs a configurable executable in a child process whenever a connection is made. GFork also creates bi-directional pipes between the child processes and the master service. These pipes are used for interprocess communication between the child process executables and a master process plugin. More information on GFork can be found [here](#).

7. Configuring multicasting/broadcasting

To enable [multicasting](#), you must whitelist the `gridftp_multicast` driver with the `-fs-whitelist file,gridftp_multicast` option:

```
globus-gridftp-server -fs-whitelist file,gridftp_multicast
```

The above command whitelists both the `file` driver and the `gridftp_multicast` driver.



Note

The `file` driver is the default [XIO](#) driver that handles reading and writing to file systems (disks). By default, this driver is already whitelisted. However, if you use the `-fs-whitelist` option, you must set *all* the

drivers you want whitelisted (and the `file` driver will still be needed to allow reads and writes to disk for non-multicast users.

For information about using multicasting, click [here](#).

8. Configuring GridFTP to enable Netlogger's bottleneck detection

For information on enabling bottleneck detection via Netlogger, see the [Gridftp-netlogger](#)¹ page on the CEDPS website.

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

Chapter 5. Key Admin Settings and Tuning Recommendations

The `globus-gridftp-server` is a flexible and tunable piece of software. It is easy for an admin to get lost in all of the options it offers. This document intends to highlight some of the more commonly important options related to performance and robustness. It does not intend to account for all of the options but rather to give the system administrator a better perspective into how to set some of the less obvious controls.

1. Concurrent Instances

A very important option for a system administrator to set is the number of simultaneous GridFTP transfers allowed. In other words, the number of clients that are allowed to connect to the server at the same time.

GridFTP is designed to be a high performance, on-demand data transfer service. Quite a bit of system resources (mainly memory) are allocated to each client connection and this is with the assumption that the session will consume even more system resources (CPU, net/disk bandwidth) when performing a high speed data transfer. For this reason, the system administrator must evaluate the resource their host machine has to offer and set a reasonable limit to the number of client connects allowed at one time.

GridFTP, in both protocol and implementation, is not designed to queue transfer requests. Queuing transfer requests is certainly an important feature for a service to offer, and the Globus Toolkit offers this with the [RFT service](#). Users who want to submit a transfer request and forget about it until they are notified of its completion should use the RFT service. Users that want an on-demand transfer service should use the GridFTP server directly.

When determining the instance concurrency level, there are two major factors to consider: **system memory** helps determine the upper limit of the instance range and **available I/O bandwidth** helps determine the lower limit.

1.1. System memory considerations

First and foremost is system memory. The recommended instance count based on system memory is:

```
instance count = system memory / 38
```

Each instance of a GridFTP server will require about 2MB of memory just to handle the connection in a sane way. Beyond that is the amount of memory required to handle a fast, TCP-based data transfer. A safe rule of thumb here is 32MB. This allows for a TCP buffer size of 16MB (which is a common client selection for high performance WAN bandwidth delay products) and a user space buffer to match that value. $2\text{MB} + 16\text{MB} + 16\text{MB} = 38\text{MB}$, thus the denominator in the above formula.

1.2. I/O bandwidth considerations

Simultaneous clients share the available I/O resources. Most often it is beneficial to allocate enough bandwidth so that each client can transfer data at an acceptable rate. In a simple model, the higher the instance count, the lower the transfer rates for each client. At some point it is beneficial for the GridFTP server to reject connections in an attempt to provide a higher level of service to its currently connected clients. There is also a point where too many simultaneous clients can cause thrashing and drop network packets. Obviously this situation should be avoided.

While no client wants to be rejected, a higher level service can take advantage of this by either trying again later at a more efficient time, reordering its work load, or finding a replica. RFT provides some of this functionality and other such services are being researched and designed.

1.3. Why More Than One?

When considering the right concurrent instance level it is helpful to consider why there should ever be more than one at a time. There are three major reasons for this:

1. **The other side of the connection is the bottleneck.**

If we assume that each transfer moves as fast as our system can send it, then, when considering overall throughput, having two connections going at half speed is roughly the same thing as having 2 full speed connections run one at a time. However, if the remote end of the connection is the bottleneck, then there is unused local bandwidth from which another simultaneous connection and thus the overall system can benefit.

2. **Hide the overhead.**

Another important aspect of simultaneous connections is that the needed overhead of control messaging can be overlapped with the payload of another sessions data transfer. Hiding this processing and messaging latency makes for a more efficient system with a higher overall throughput.

3. **Provide an interactive service.**

In some case, users may find connection rejections unacceptable and would prefer a slower overall system provided they could connect to it immediately for the purpose of an interactive session.

In the case of #3 the the highest safe level of instance count possible is ideal. In the other two case the ideal number is less deterministic. At least 10 instance is always recommended.

1.4. Setting the instance cap

If using GForge or Xinetd, set the instance cap by adding the following line to the configuration file:

```
instance = <integer>
```

If running the GridFTP server as a daemon, use the following option to set the instance cap:

```
-connections-max <integer>
```

2. Disk Block Size

The **globus-gridftp-server** sits on top of various file systems. Each file system has its own ideal access patterns and I/O buffer sizes. To provide the user with some means of control, we offer the option:

```
-blocksize <number>
```

This number indicates the size of the read requests posted to the disk.

3. System TCP Buffer Settings

The most important setting in achieving high performance, TCP-based transfers is the TCP buffer size. It is our experience that this should be set to at least 16MB. On Linux systems, this is done by editing the file `/etc/sysctl.conf` and adding the following lines:

```
net/core/rmem_max = 16777216
```

```
net/core/wmem_max = 16777216  
  
net/ipv4/tcp_rmem = 8192 1048576 16777216  
net/ipv4/tcp_wmem = 8192 1048576 16777216
```

This sets the max to 16MB, the default to 1MB, and the min to 8KB. In most cases this will be a good value, but administrators are encouraged to experiment.

4. Data Interface

On systems that have multiple network interfaces, the system admin likely wants to associate data transfers with the fastest possible NIC available. This can be done in the GridFTP server by using the option:

```
--data-interface <ip address>
```

Chapter 6. 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`.

Chapter 7. Debugging

1. Logging

As of Globus 4.2.0, GridFTP server provides system administration logs in 2 different formats. The CEDPS best practices compliant format is a new format provided by GridFTP server available in Globus 4.2.1. For more details on the CEDPS Logging format, see <http://cedps.net/index.php/LoggingBestPractices>.

1.1. Configuring CEDPS format system administration logs

```
globus-gridftp-server -log-module stdio_ng -log-level info,warn,error -logfile /var/log/gr
```

For more information about the logging options, see [globus-gridftp-server\(1\)](#).

Sample log file: [gridftp.log1](#)¹

1.2. Configuring traditional format system administration logs

```
globus-gridftp-server -log-module stdio -log-level info,warn,error -logfile /var/log/gridf
```

which is the same as

```
globus-gridftp-server -log-level info,warn,error -logfile /var/log/gridftp.log
```

stdio is the default log-module.

Sample log file: [gridftp.log2](#)²

1.3. Netlogger-style logging

Apart from the 2 formats mentioned above, GridFTP server can log netlogger style information for each transfer.

```
globus-gridftp-server -log-transfer /var/log/gridftp.log
```

Sample log file: [gridftp.log3](#)³

¹ /toolkit/docs/4.2/4.2.1/data/gridftp/gridftp.log1

² /toolkit/docs/4.2/4.2.1/data/gridftp/gridftp.log2

³ /toolkit/docs/4.2/4.2.1/data/gridftp/gridftp.log3

Chapter 8. Troubleshooting

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

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

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

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

1. Error Codes in GridFTP

Table 8.1. GridFTP Errors

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

2. Establish control channel connection

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

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

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

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

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

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

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

The server will print the port it is listening on:

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

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

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

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

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

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

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

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

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

3. Try running `globus-url-copy`

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

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

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

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

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

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

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

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

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

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

4. If your server starts...

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

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

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

5. High latency for GridFTP server connections

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

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

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

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

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

Chapter 9. Usage statistics collection by the Globus Alliance

1. GridFTP-specific usage statistics

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](#)¹ section.

- Start time of the transfer
- End time of the transfer
- Version string of the server
- TCP buffer size used for the transfer
- 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, use the `-disable-usage-stats` option of `globus-gridftp-server`. 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](#)² on the collection of usage statistics.

¹ ../../Usage_Stats.html

² ../../Usage_Stats.html

Appendix A. GridFTP Admin Tool

Name

globus-gridftp-server -- Configures the GridFTP Server

globus-gridftp-server

Tool description

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



Note

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

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

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

Options are one per line, with the format:

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

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

For example:

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

Developer notes

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

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

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

Command syntax

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

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

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

Command line options

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



Note

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

Informational Options

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

Modes of Operation

<code>inetd <0 1></code> , <code>-i, -inetd</code>	Run under an inetd service. Default value: FALSE
<code>daemon <0 1></code> , <code>-s, -daemon</code>	Run as a daemon. All connections will fork off a new process and setuid if allowed. See Section 4, “Running in daemon mode” for more information. Default value: TRUE
<code>detach <0 1></code> , <code>-S, -detach</code>	Run as a background daemon detached from any controlling terminals. See Section 4, “Running in daemon mode” for more information. Default value: FALSE
<code>exec <string></code> <code>, -exec</code> <code><string></code>	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

<code>chdir <0 1>, -chdir</code>	Change directory when the server starts. This will change directory to the dir specified by the <code>chdir_to</code> option. Default value: TRUE
<code>chdir_to <string>, -chdir-to <string></code>	Directory to <code>chdir</code> to after starting. Will use <code>/</code> if not set. Default value: not set
<code>fork <0 1>, -f, -fork</code>	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
<code>single <0 1>, -1, -single</code>	Exit after a single connection. Default value: FALSE

Authentication, Authorization, and Security Options

<code>auth_level <number>, -auth-level <number></code>	<ul style="list-style-type: none">• 0 = Disables all authorization checks.• 1 = Authorize identity only.• 2 = Authorize all file/resource accesses. <p>If not set, the GridFTP Server uses level 2 for front ends and level 1 for data nodes.</p> <p>Default value: not set</p>
<code>allow_from <string>, -allow-from <string></code>	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
<code>deny_from <string>, -deny-from <string></code>	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
<code>cas <0 1>, -cas</code>	Enable <u>Community Authorization Service (CAS)</u> authorization. For complete instructions on setting up a GridFTP server to use CAS, click here . Default value: TRUE
<code>secure_ipc <0 1>, -si, -secure-ipc</code>	Use GSI security on the IPC channel. Default value: TRUE

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

-connections-
disabled

Logging Options

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

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

- ERROR
- WARN
- INFO
- DUMP
- ALL

For example:

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

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

Default value: ERROR

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

<string>,
-log-module
<string>

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

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

Example:

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

Default value: not set

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

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

Default value: not set

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

<string>, -L
<string>, -lo-
gdir <string>

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

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

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

¹ ../../Usage_Stats.html

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

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

Single and Striped Remote Data Node Options

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

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

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

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

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

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

Disk Options

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

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

Default value: FALSE

Network Options

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

Default value: not set

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

Default value: not set

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

Default value: not set

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

Default value: not set

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

Default value: not set

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

Default value: not set

Timeouts

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

Default value: 30

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

Default value: 600

idle-timeout
<number>

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

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

User Messages

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

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

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

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

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

Module Options

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

Default value: not set

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

Other Options

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

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

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

Limitations

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

Globus GridFTP server does not run on windows

Appendix B. GT 4.2.1 GridFTP: Setting up 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. There 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 4.2.1 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 Toolkit* - You need the GridFTP distributed with GT (compatible with 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](#) and [SRB](#)⁴ 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.

¹ <http://www.globus.org/toolkit/downloads/4.2.1/>

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.sdsc.edu/srb>

4.1. Building Globus

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

```
% bunzip2 gt4.2.1-all-source-installer.tar.bz2
% tar -xvf gt4.2.1-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](#)⁵. 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.

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>
```

⁵ <http://www.gridpackagingtools.org/>

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 [Chapter 2, 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](#)⁶ file for more information.

⁶ http://www-unix.mcs.anl.gov/~bresnaha/SRB_DSI_Doc/README.txt

Appendix C. Running the Globus GridFTP Server With GFork

1. Introduction

GFork is a user-configurable super-server daemon very similar to xinetd in that it listens on a TCP port. When clients connect to a port, it runs an administrator-defined program which services that client connection, just as xinetd does.

An unfortunate drawback to xinetd is that there is no way to maintain or share long-term information. Every time a client connects, a new process is created; and every time that client disconnects, the process is destroyed. All of the information regarding the specific interactions with a given client is lost with these transient processes. A further disadvantage is that there is no way for these service instances to share service-specific information with each other while they are running.

There are times when it is useful for a service to maintain long-term service-specific state, or for a service to share state across client connections. GFork is designed to address this situation. GFork runs a long term master program (that is user-defined) and forms communication links via UNIX pipes between this process and all client connection child processes. This allows long-term state to be maintained in memory and allows for communication between all nodes.

Associated with a GFork instance is a master process. When GFork starts, it runs a user-defined master program and opens up bi-directional pipes to it. The master program runs for the lifetime of the GFork daemon. The master is free to do whatever it wants; it is a user-defined program. Some master programs listen on alternative TCP connections to have state remotely injected. Others monitor system resources, such as memory, in order to best share resources. As clients connect to the TCP listener, child processes are forked which then service the client connection. Bi-directional pipes are opened up to the child processes as well. These pipes allow for communication between the master program and all child processes. The master program and the child programs have their own protocol for information exchange over these links. GFork is just a framework for safely and quickly creating these links.

2. Use Cases

The creation of GFork was motivated by the Globus GridFTP server. GridFTP can be run as a striped server where there is a frontend and several backends. The backends run in tandem to transfer files faster by tying together many NICs. The frontend is the contact point for the client where transfer requests are made. When the frontend is run out of inetd, the list of possible backends must be statically configured. Unfortunately, backends tend to come and go. Sometimes backends fail, and sometimes backends are added to a pool. We needed a way to have a [fixme good synopsis: dynamic pool of backends for use in live transfers]. To accomplish this we created GFork.

3. Configuration

A major difference between GFork configuration and xinetd is that GFork only runs one service per instance, where xinetd runs many services per instance all associated with many different ports. GFork takes a single configuration file and handles a single service. If there is demand, GFork will be enhanced to handle many services in the way that xinetd does.

Running the globus-gridftp-server under GFork is almost identical to running it under xinetd. First, you need a configuration file:

```
service gridftp2
{
env += GLOBUS_LOCATION=<path to GL>
env += LD_LIBRARY_PATH=<path to GL>/lib
server = <path to GL>/sbin/globus-gridftp-server
server_args = -i
server_args += -d ALL -l <path to GL>/var/gridftp.log
port = 5000
}
```

That portion is identical to xinetd. In fact, an existing xinetd configuration file should work.

When running GridFTP out of GFork, the server should be run with a master program. The master program provides enhanced functionality such as dynamic backend registration for striped servers, managed system memory pools and internal data monitoring for both striped and non-striped servers.

To run with a master program, the following two lines are needed in the config file.

```
master = <path to GL>/libexec/gfs-gfork-master
master_args = <options>
```

These last two options relate to the master program and work in the same way that `server` and `server_args` do. The first line tells GFork what master program to use (for the GridFTP server, we use **gfs-gfork-master**). The second line provides options to the master program.

The full list of master options are as follows (this is to date only, run the program with `--help` for newer options):

<code>-b --reg-cs <contact string></code>	Contact to the frontend registry. This option makes it a data node.
<code>-df --dn-file <path></code>	Path to a file containing the list of acceptable DNs. Default is system gridmap file.
<code>-G --gsi <bool></code>	Enable or disable GSI. Default is on.
<code>-h --help</code>	Print the help message.
<code>-l --logfile <path></code>	Path to the logfile.
<code>-p --port <int></code>	Port where the server listens for connections.
<code>-s --stripe-count <int></code>	The maximum number of stripes to give to each server. A value of 0 indicates all stripes are available.
<code>-u --update-interval <int></code>	Number of seconds between registration updates.

The following is an example GFork configuration file:

```
service gridftp2
{
instances = 100
env += GLOBUS_LOCATION=/home/bresnaha/Dev/Globus-gfork3/GL
env += LD_LIBRARY_PATH=/home/bresnaha/Dev/Globus-gfork3/GL/lib
server = /home/bresnaha/Dev/Globus-gfork3/GL/sbin/globus-gridftp-server
server_args = -i -aa
server_args += -d ALL -l /home/bresnaha/tst.log
}
```

```
server_args += -dsi remote -repo-count 1
nice = 10
port = 5000
master = /home/bresnaha/Dev/Globus-gfork3/GL/libexec/gfs-gfork-master
master_args = -port 6065 -l /home/bresnaha/master.log -G n
master_args += -dn /home/bresnaha/master_gridmap
}
```

Once you have a configuration file, run GFork with:

```
% gfork -c <path to config file>
```

4. GFork and Striped Servers

As mentioned in [Section 1](#), “Remote data-nodes and striped operation”, GridFTP offers a powerful enhancement called striped servers. In this mode a GridFTP server is set up with a single frontend and one or more backends. All of the backends work in concert to transfer a single file and thereby achieve high throughput rates. Here we describe how to configure one frontend and multiple backends for use as a striped server with GFork.

4.1. Frontend Configuration

The frontend server described here is run using dynamic backends. We need additional options for both the GridFTP server and the master program. The following lines are added to the config file:

```
server_args += -dsi remote
master_args = -port 8588
master_args += -df <path to gridmap file>
```

The first line is an additional argument to the GridFTP server. It tells the server that it will be operating in split mode (separate frontend and backend processes) and that it will be using the frontend. (Specifically it tells the server to use the 'remote' DSI).

The second line tells the master program on which port it should listen for backend registrations. Backend services can then connect to this port to notify the frontend of their existence. By default, a registration is good for 10 minutes, but a backend is free to refresh its registration. In this way, a frontend is provided with the list of possible backends (stripes) which may be used for a transfer.

The third line provides the master program with a list of authorized DNs. Each line in the file must contain a GSI DN (certificate subject). In order to register, the backend must authenticate and provide its DN. The provided DN is checked against this file. In other words, the file is a list of DNs that may register with the frontend. If the master program is not given a `-df` option and is given the `-G` option, then there is no registration security at all.

4.2. Backend Configuration

Any striped server setup can have more than one backend service. Furthermore, any one computer can run multiple backends. The following explains how to set up a backend server. These steps should be repeated for each needed backend instance.

A backend server may also be run with GFork, it just needs different options for both the GridFTP server and the master program. A sample backend config file is shown here:

```
service gridftp2
{
```

```
env += GLOBUS_LOCATION=<path to GL>
env += LD_LIBRARY_PATH=<path to GL>/lib

server = <path to GL>/sbin/globus-gridftp-server
server_args = -i
server_args += -dn
master = <path to GL>/libexec/gfs-gfork-master
master_args = -b localhost:8588
}
```

Notable additions to this file are:

```
server_args += -dn
master_args = -b localhost:8588
```

The first line tells the GridFTP server that it will be a 'data node', which is another name for a backend.

The second line tells the master program two things, first that it will be a master of a data node, and second what the frontend's registration contact point is. Note that in our example we have a hostname of 'localhost' and a port of '8588'. 8588 is (and must be) the same port that was provided to the frontend's master program in the previous step.

Once the configuration file is complete, run GFork again as follows:

```
% gfork -c <conf file>
```

This will start up the data node and the master program will register itself to the frontend and refresh its registration every 5 minutes (default setting).

5. GFork with Memory Management

Another feature of the GridFTP GFork plugin is memory usage limiting. Under extreme client loads, it is possible that GridFTP servers require more memory than the system has available. Due to a common kernel memory allocation scheme known as optimistic provisioning, this situation can lead to a full consumption of memory resources and thus trigger the out of memory handler. The OOM handler will kill processes in a difficult-to-predict way in order to free up memory. This will leave the system in an unpredictable and unstable state; obviously, this is a situation that we want to avoid.

To control this situation, the GridFTP GFork plugin has a memory limiting option. This will attempt to limit memory usage to a given value or to the maximum amount of RAM in the system. Most of the memory is given to the first few connections, but when the plugin detects that it is overloaded, each session is limited to half the available memory.

To enable this feature, one of two options must be passed to the master program via the `master_args` in the config file:

<code>-m</code>	Limits memory consumption to amount of RAM in the system.
<code>-M <formatted int></code>	Limits memory to the given value.

Another important option should be provided in the GFork config file: `instance`. When a client connects to GFork, a GridFTP server instance is executed. This instance requires a certain amount of RAM. If connections are coming in too fast, this can act as a DOS attack. Limiting the number of allowed simultaneous connections will help the memory management algorithm do its job. This limit is set with:

```
instance = <int>
```

We recommend a value of 100 or $\lfloor \text{RAM} / 2\text{M} \rfloor$, whichever is smaller.

The following is an example of a GFork configuration file with memory limiting enabled:

```
service gridftp2
{
instance = 100
env += GLOBUS_LOCATION=<path to GL>
env += LD_LIBRARY_PATH=<path to GL>/lib
server = <path to GL>/sbin/globus-gridftp-server
server_args = -i
server_args += -dn
master = <path to GL>/libexec/gfs-gfork-master
master_args = -M 512M
}
```

Glossary

some terms not in the docs but wanted in glossary: *scheduler client/server transfer*

C

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

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

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

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

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

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

S

scheduler Term used to describe a job scheduler mechanism to which GRAM interfaces. It is a networked system for submitting, controlling, and monitoring the workload of batch jobs in one or more computers. The jobs or tasks are scheduled for execution at a time chosen by the subsystem according to an available policy and availability of resources. Popular job schedulers include Portable Batch System (PBS), Platform LSF, and IBM LoadLeveler.

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

T

third party transfers

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

The following is a more detailed, programmatic description.

In a third party transfer, there are three entities involved. The client, who will only orchestrate, but not actually take place in the data transfer, and two servers one of which will be sending data to the other. This scenario is common in Grid applications where you may wish to stage data from a data store somewhere to a super-computer you have reserved. The commands are quite similar to the client/server transfer. However, now the client must establish two control channels, one to each server. He will then choose one to listen, and send it the PASV command. When it responds with the IP/port it is listening on, the client will send that IP/port as part of the PORT command to the other server. This will cause the second server to connect to the first server, rather than the client. To initiate the actual movement of the data, the client then sends the RETR "filename" command to the server that will read from disk and write to the network (the "sending" server) and will send the STOR "filename" command to the other server which will read from the network and write to the disk (the "receiving" server).

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

Index

A

- admin scenarios
 - running in daemon mode, 6
- administrative settings, recommended, 17

B

- building and installing
 - general instructions, 1
- building and installing GridFTP
 - only a combination of certain GridFTP elements, 2
 - only a static GridFTP server, 2
 - only GridFTP and Utilities, 1
 - only the GridFTP client, 1
 - only the GridFTP SDK, 2
 - only the GridFTP server, 1

C

- commandline tool
 - globus-gridftp-server, 28
- configuration interface for GridFTP, 4
- configuring GridFTP, 4
 - multicasting, 15
 - overview, 4
 - run with Community Authorization Service (CAS), 14
 - run with GFork, 15
 - run with UDT for third party transfers, 14
 - security
 - anonymous mode, 9
 - gsiftp, 11
 - over SSH, 10
 - username/password, 10
 - separation of processes, 13
 - split process, 13
 - striped servers, 13
 - types, 4
 - with DSI, 14

D

- deploying
 - running under inetd or xinetd, 6

E

- errors, 23

G

- globus-gridftp-server, 28

L

- logging, 21

P

- performance, 17

R

- running in daemon mode, 6

S

- security considerations for GridFTP, 8

T

- testing, 20
- troubleshooting for GridFTP, 22
- tuning, 17

U

- usage statistics for GridFTP, 26