

GT 4.2.1 Reliable File Transfer (RFT)

Service: User's Guide

GT 4.2.1 Reliable File Transfer (RFT) Service: User's Guide

Introduction

RFT is the Reliable Transfer Service. It allows clients to submit URL transfer requests to a persistent service which will perform the transfers on behalf of the client. RFT Service implementation in GT 4.2.1 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and to delete files using GridFTP. The user creates a RFT resource by submitting a list of URL pairs of files that need to be transferred/deleted to RFT Factory service. The user also specifies the time to live for the resource the user is creating to a GT 4.2.1 Container in which RFT is deployed and configured. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The operations exposed by both RFT factory and RFT service are briefly described below. The resource the user created also exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard command line clients.

Table of Contents

1. Using RFT	1
1. globus-crft	1
I. RFT Commands	3
rft	4
globus-crft	6
rft-delete	8
2. Troubleshooting	9
1. Troubleshooting tips	9
2. RFT fault-tolerance and recovery	9
3. Usage statistics collection by the Globus Alliance	10
1. Usage statistics sent by RFT	10
Index	11

Chapter 1. Using RFT

The Java clients, **rft** and **rft-delete** commands are available for very simple transfers. For more options, use the programming instructions [here](#).

1. globus-crft

Beginning with 4.2.0, RFT also offers a new C client, **globus-crft**.

1.1. Submitting A Transfer

To submit a transfer request the user must first create a 'transfer file'. Each line of this ASCII text file is a source/destination URL pair. There can be any number of lines per file. An example file follows:

```
gsiftp://localhost:2811/etc/group gsiftp://localhost:2811/tmp/test_crft
gsiftp://ftp.globus.org:2811/pub/README gsiftp://myhost.here/home/user/file
```

This file requests two transfers. The first will use the GridFTP server running on the localhost to transfer /etc/group to /tmp/test_crft. The second will transfer the file /pub/README on ftp.globus.org to the file /home/user/file located on myhost.here

Once the transfer file is created globus-crft can be used in a variety of ways to transfer a file. The most simple is the blocking transfer:

```
% globus-crft -c -s -m -vb -f <transfer file> -e <container contact string>
```

Looking at each option individually, this command line does the following

-c Create a new RFT server., **-s** Submit the transfer request. Since RFT is a 2 phase commit we allow the client the ability to do them in separate stages, however it is expected that the vast majority of the time -c and -s will be used together.

-m Monitor the transfers. When this option is used the client will block until all transfers have completed. It monitors the status of the transfers along the way and can report it to the user.

-vb Display verbose output. This just increases the level of diagnostic messages sent to stdout. When combined with -m it will allow the user to see the status of a transfer.

-f <transfer file> This option is a pointer to the transfer file described above.

-e <container contact strings> The contact string is in the following form:

```
https://hostname.com:8443/wsrf/services/
```

The strings ___ and ___ will be appended to the given string in order for the client to interact with that containers delegation service and RFT service.

1.2. Non-blocking Transfer

The client can do non-blocking RFT submission. It can submit an RFT request and then terminate, returning later to monitor the status of the request. To accomplish this the client saves the EPR of the newly created RFT service to disk.

```
% globus-crft -c -s -f <transfer file> -e <container contact string> \  
    -ef <epr output file>
```

At some point later the client uses this same file to monitor the state of the transfer:

```
% globus-crft -ef <epr input file> --getOverallStatus
```



Note

Note that in both cases the option `-ef` is used. In the first case, since the `-c` option is used, we are creating a new service and the `-ef` option is a pointer to an output file. In all cases where `-c` is not used, the `-ef` switch is a pointer to an input file.

1.3. Cleaning Up

Once a transfer request completes, the user should destroy the resources associated with it. If the user stored the EPR of the service it created, this can be done with:

```
% globus-crft -ef <epr input file> --destroy
```

1.4. More

For a list of more options run:

```
globus-crft --help
```

RFT Commands

Name

rft -- Submit and monitor a 3rd party GridFTP transfer

rft

Tool description

Submits a transfer to the Reliable File Transfer Service and prints out the status of the transfer on the console.

Command syntax and options

```
rft [-h <host-ip of the container defaults to localhost>
-r <port, defaults to 8080>
-l <lifetime for the resource default 60mins>
-m <security mechanism. 'msg' for secure message or 'conv' for
  secure conversation and 'trans' for transport. Defaults to
  secure transport.>
-p <protection type, 'sig' signature and 'enc' encryption,
  defaults to signature >
-z <authorization mechanism can be self or host. default self>
-file <file to write EPR of created Reliable File Transfer Resource]>
-f <path to the file that contains list of transfers>
```

This is a sample transfer file that the command-line client will be able to parse. It can also be found in **\$GLOBUS_LOCATION/share/globus_wsrf_rft_client/** along with other samples for directory transfers and deletes (lines starting with # are comments):

```
This option when it is set to true means to perform transfer in binary
form, if it is set to false transfer is done in ASCII. Default is binary.
true
```

```
#Block size in bytes that is transferred. Default is 16000 bytes.
16000
```

```
#TCP Buffer size in bytes
```

```
#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. Use the
bandwidth-delay product as your buffer size.
```

```
16000
```

```
#Notpt (No thirdPartyTransfer): turns third-party transfers off is this option
is set to false (on if set to true).
```

```
Site firewall and/or software configuration may prevent a connection
between the two servers (a third party transfer). If this is the case,
RFT 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.
```

```
false
```

```
#Number of parallel streams: Specifies the number of parallel data connections
that should be used.
```

1

#Data Channel Authentication (DCAU): Turns off data channel authentication for FTP transfers is set to false.(the default is true to authenticate the data channel).

true

Concurrency of the request: Number of files that you want to transfer at any given point. Default is set to one.

1

#Grid Subject name of the source gridftp server. This is used for Authorization purposes. If the source gridftp server is running with host credentials you can specify "n /DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710

#Grid Subject name of the destination gridftp server. This is used for Authorization purposes. If the destination gridftp server is running with host credentials you can specify "null" here. By default Host authorization is done. /DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710

#Transfer all or none of the transfers: This option if set to true will make RFT to clean up (delete) all the transfers that have been done already if one of the transfers fails.

false

#Maximum number of retries: This is number of times RFT retries a transfer failed with a non-zero exit code.

10

#Source/Dest URL Pairs: gsiftp urls of source followed by destination.

If directory is to be recursively transferred the source gsiftp url and destination gsiftp url should end with "/". Currently RFT supports Directory - Directory, File - Directory, File - File transfers. There can be more URL pairs and all of them use the same options as above for performing the transfer.

gsiftp://localhost:5678/tmp/rftTest.tmp

gsiftp://localhost:5678/tmp/rftTest_Done.tmp

Limitations

This command line client is very simple and does not do any intelligent parsing of various command line options or of the options in the sample transfer file. It works fine if used in the way documented here. For more information on all these options please refer to the [documentation of globus-url-copy](#). Also, please note that the maximum number of transfers the command-line client can process before running out of memory is ~21K with the default JVM heap size, which was 64M in our tests. Please look at [Performance Reports](#)¹ for more details.

¹ ../rft_scalability_3_9_4.doc

Name

globus-crft -- Command-line client to transfer files using RFT

globus-crft

Tool description

This distribution contains a client to the RFT service written in C. RFT is the reliable transfer server. It allows clients to submit URL transfer requests to a persistent service which will perform the transfers on behalf of the client.

Options

- | | |
|---|--|
| <code>-a --all-or-none <on off></code> | Enable all or none transfer: default off. |
| <code>-con --concurrent <int></code> | The number of simultaneous transfers. |
| <code>-C --cancel</code> | Cancel a transfer. |
| <code>-c --create</code> | Create a new RFT service. |
| <code>-del --delete</code> | Delete a URL. |
| <code>-ds --destination-subject <subject></code> | The expected domain name of the destination GridFTP server. |
| <code>-d --destroy</code> | Destroy the server. If used with <code>-monitor</code> , wait until completion and then destroy. |
| <code>-D --done</code> | Return the current status of the transfer in the exit code: <ul style="list-style-type: none">• 0=Done• 1=Active• 2=Pending• 3=Cancelled• 4=Failed |
| <code>-ef --epr-file <path></code> | Path to the EPR file. If used with <code>--create</code> the EPR is written to this location. In all other cases the EPR is read from this location. |
| <code>-ez --easy</code> | Create, submit, and wait for the transfer to complete. The job is started with some standard options. |
| <code>-e --factory <contact></code> | The endpoint to contact when creating a server. Used with <code>--create</code> . |
| <code>-f --transfer-file <path></code> | A path to a file that contains the source destination URL pairs. |
| <code>-gS --getStatusSet <int> <int></code> | Get the status of all the transfer requests in the range. |
| <code>-g --getStatus <source url></code> | Get the status of the given source url. |

-h | --help Print usage information.

FIXME - finish converting to variable list:

-ms | --message-security <[sig] | [conv] | [trans]>
 Security mechanism. 'msg' for secure message,
 'conv' for secure conversation, 'trans' for
 transport. The default is trans.

-m | --monitor Wait for the service to complete, and receive
 status updates.

-os | --getOverallStatus Get the overall status.

-p | --protection <[sig] | [enc]>
 Protection type. 'sig' for signature, 'enc' for
 encryption. The default is 'sig'.

-P | --parallel <int> The number of parallel sockets to use with each
 transfer.

-q | --quiet Write no output.

-rs | --getRequestStatus Get the request status.

-r | --retries Number of retries

-S | --subject <subject> The expected domain name of both the source and
 destination GridFTP servers.

-ss | --source-subject <subject>
 The expected domain name of the source GridFTP
 server.

-s | --submit Start the RFT service

-tb | --tcp-bs <int> The TCP buffer size to use with each transfer.

-ttl | --termination-time <int>
 Set the lifetime of the service.

-v | --version Print version information.

-vb | --verbose Display much more output.

-xi | --xml-input <path> Read the request description from the given xml
 description.

-xo | --xml-output <path> Write the request description to the given file
 location in xml format.

-z | --authz <[self] | [host] | [id <subject>]>
 Authorization. 'self', 'host', or 'id <DN>'.

Limitations

No limitations exist with this command line tool.

Name

rft-delete -- Command-line client to delete files using RFT

rft-delete

Tool description

This command-line tool is used to submit a list of files to be deleted.

Command and options

```
rft-delete [-h <host-ip of the container default localhost>
-r <port, defaults to 8080>
-l <lifetime for the resource default 60mins>
-m <security mechanism. 'msg' for secure message or 'conv' for
  secure conversation and 'trans' for transport. Defaults to
  secure transport.>
-p <protection type, 'sig' signature and 'enc' encryption,
  defaults to signature >
-z <authorization mechanism can be self or host. default self>
-file <file to write EPR of created Reliable File Transfer Resource]>
-f <path to the file that contains list of transfers>
```

This is a sample file that the command line client will be able to parse, and it can also be found in **\$GLOBUS_LOCATION/share/globus_wsrf_rft_client/** along with other samples for directory transfers and deletes (lines starting with # are comments):

```
# Subject name (defaults to host subject)
  /DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
  gsiftp://localhost:5678/tmp/rftTest_Done.tmp
  gsiftp://localhost:5678/tmp/rftTest_Done1.tmp
```

Limitations

No limitations exist with this command line tool.

Chapter 2. Troubleshooting

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

1. Troubleshooting tips

- Always have a valid proxy before using command line RFT clients.
- Make sure to provide suitable options to the client, and especially for the Termination time, so that the resource does not get destroyed before finishing the transfers.

2. RFT fault-tolerance and recovery

RFT uses PostgreSQL to check-point transfer state in the form of restart markers and recover from transient transfer failures, using retry mechanism with exponential backoff, during a transfer. RFT has been tested to recover from source and/or destination server crashes during a transfer, network failures, container failures (when the machine running the container goes down), file system failures, etc. RFT Resource is implemented as a PersistentResource, so ReliableFileTransferHome gets initialized every time a container gets restarted. Please find a more detailed description of fault-tolerance and recovery in RFT below:

- **Source and/or destination GridFTP failures:** In this case RFT retries the transfer for a configurable number of maximum attempts with exponential backoff for each retry (the backoff time period is configurable also). If a failure happens in the midst of a transfer, RFT uses the last restart marker that is stored in the database for that transfer and uses it to resume the transfer from the point where it failed, instead of restarting the whole file. This failure is treated as a container-wide backoff for the server in question. What this means is that all other transfers going to/from that server, across all the requests in a container, will be backed off and retried. This is done in order to prevent further failures of the transfers by using knowledge available in the database.
- **Network failures:** Sometimes this happens due to heavy load on a network or for any other reason packets are lost or connections get timed out. This failure is considered a transient failure and RFT retries the transfer with exponential backoff for that particular transfer (and not the whole container, as with the source and/or destination GridFTP failures).
- **Container failures:** These type of failures occur when the machine running the container goes down or if the container is restarted with active transfers. When the container is restarted, it restarts ReliableTransferHome, which looks at the database for any active RFT resources and restarts them.

2.1. Failure modes that are not addressed:

- Running out of disk space for the database.

Chapter 3. Usage statistics collection by the Globus Alliance

1. Usage statistics sent by RFT

The following usage statistics are sent by default in a UDP packet at the end of life time of each RFT Resource (or when a RFT resource is destroyed).

- Total number of files transferred by RFT since RFT was installed
- Total number of bytes transferred by RFT since RFT was installed
- Total number of files transferred in this RFT Resource
- Total number of bytes transferred in this RFT Resource
- Creation time of this RFT Resource
- Factory Start Time

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 the RFT component. Nevertheless, if you wish to disable this feature, please see the "Usage Statistics Configuration" section of [Configuring Java WS Core](#) for instructions.

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ [../../../../Usage_Stats.html](#)

Index

C

commands

- globus-crft, 6

- rft, 4

- rft-delete, 8

D

deleting files (from a list), 8

F

fault tolerance, 9

R

recovery, 9

S

submitting a third party GridFTP transfer, 4

submitting transfers (from a list)

- C client, 6

T

troubleshooting

- (for end-users), 9

- (for users), 9

U

usage statistics, 10

using RFT, 1