GT 4.0 Reliable File Transfer (RFT) Service

GT 4.0 Reliable File Transfer (RFT) Service

Table of Contents

Chapter 1. Data Management: Key Concepts

Overview of Data Management in GT4

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

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

Data movement

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

GridFTP

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

The Globus Toolkit provides

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

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

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

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

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

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

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

For more information about GridFTP, see the documentation[2].

Reliable File Transfer (RFT) Service

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

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

For more information about RFT, see the documentation[3].

Data replication

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

Replica Location Service (RLS)

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

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

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

- A *logical file name* is a unique identifier for the contents of a file.

- A *physical file name* is the location of a copy of the file on a storage system.

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

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

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

Using RLS: An Example

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

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

For more information

For more detailed key concepts about RLS, click here[4].

For more information about RLS, see the documentation[5].

Higher level data services

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

Data Replication Service (DRS)

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

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

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

For more information

For more information about DRS, go to the Tech Preview documentation[6].

Notes

1. ../gridftp/user-index.html
2. ../gridftp/
3. ../rft/
4. rls.html
5. ../rls/
6. ../../techpreview/datarep/

Chapter 2. GT 4.0 Release Notes: Reliable File Transfer (RFT) Service

Component Overview

The Reliable Transfer Service (RFT) Service implementation in 4.0 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters over GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consists of a set of third-party gridftp transfers) to RFT Factory service. 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 resource the user created 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 WS-RF command line clients and other resource properties.

Feature Summary

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports MLST command.
- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option
- Different subject names for source and destination GridFTP servers for authorization mechanism.

- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Bug Fixes

- Bug 2749[1]
- Bug 2724[2]
- Bug 2683[3]
- Bug 2662[4]
- Bug 2703[5]
- Bug 2847[6]
- Bug 2826[7]
- Bug 2312[8]
- Bug 2879[9]
- Bug 2930[10]
- Bug 2935[11]
- Bug 2852[12]
- Bug 2986[13]
- Bug 3017[14]
- Bug 2984[15]
- Bug 2965[16]
- Bug 2666[17]
- Bug 2927[18]
- Bug 3072[19]
- Bug 2916[20]
- Bug 2721[21]
- Bug 2999[22]
- Bug 3110[23]
- Bug 3091[24]
- Bug 3130[25]
- Bug 2914[26]
- Bug 3115[27]
- Bug 2956[28]

Known Problems

Does not compile with JDK 1.3.1

The configured maximum allowed active transfers constraint is not enforced. For more details please look at the bug report.[29]

Technology Dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Delegation Service
- Service Groups
- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Tested Platforms

- Linux

Backward Compatibility Summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

For More Information

Click here[30] for more information about this component.

Notes

1. http://bugzilla.globus.org/globus/show_bug.cgi?id=2749
2. http://bugzilla.globus.org/globus/show_bug.cgi?id=2724
3. http://bugzilla.globus.org/globus/show_bug.cgi?id=2683
4. http://bugzilla.globus.org/globus/show_bug.cgi?id=2662
5. http://bugzilla.globus.org/globus/show_bug.cgi?id=2703
6. http://bugzilla.globus.org/globus/show_bug.cgi?id=2847
7. http://bugzilla.globus.org/globus/show_bug.cgi?id=2826
8. http://bugzilla.globus.org/globus/show_bug.cgi?id=2312
9. http://bugzilla.globus.org/globus/show_bug.cgi?id=2879
10. http://bugzilla.globus.org/globus/show_bug.cgi?id=2930
11. http://bugzilla.globus.org/globus/show_bug.cgi?id=2935
12. http://bugzilla.globus.org/globus/show_bug.cgi?id=2852
13. http://bugzilla.globus.org/globus/show_bug.cgi?id=2986
14. http://bugzilla.globus.org/globus/show_bug.cgi?id=3017
15. http://bugzilla.globus.org/globus/show_bug.cgi?id=2984
16. http://bugzilla.globus.org/globus/show_bug.cgi?id=2965
17. http://bugzilla.globus.org/globus/show_bug.cgi?id=2666
18. http://bugzilla.globus.org/globus/show_bug.cgi?id=2927
19. http://bugzilla.globus.org/globus/show_bug.cgi?id=3072
20. http://bugzilla.globus.org/globus/show_bug.cgi?id=2916
21. http://bugzilla.globus.org/globus/show_bug.cgi?id=2721
22. http://bugzilla.globus.org/globus/show_bug.cgi?id=2999
23. http://bugzilla.globus.org/globus/show_bug.cgi?id=3110
24. http://bugzilla.globus.org/globus/show_bug.cgi?id=3091
25. http://bugzilla.globus.org/globus/show_bug.cgi?id=3130
26. http://bugzilla.globus.org/globus/show_bug.cgi?id=2914
27. http://bugzilla.globus.org/globus/show_bug.cgi?id=3115
28. http://bugzilla.globus.org/globus/show_bug.cgi?id=2956
29. http://bugzilla.globus.org/globus/show_bug.cgi?id=3121
30. http://www-unix.globus.org/toolkit/docs/4.0/data/rft/index.html

Chapter 3. GT 4.0 Reliable File Transfer (RFT) Service: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with RFT. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

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

RFT is used to perform third-party transfers across GridFTP servers. It uses a database to store its state periodically so the transfers can be recovered from any failures. RFT uses standard grid security mechanisms for authorization and authentication of the users. In order to effectively use RFT you should have installed and configured a database with RFT database schemas and have the necessary security infrastructure in place to perform a 3rd party transfer.

Building and Installing

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

The following are specialized instructions for advanced developers who want to deploy latest code from CVS:

Build RFT from CVS:

1. Configure your CVSROOT to point to globus cvs location.
2. Run:
   ```
   cvs co ws-transfer
   ```
3. Run:
   ```
   cd ws-transfer/reliable
   ```
4. Set GLOBUS_LOCATION to point to your globus installation.
5. Run:
   ```
   ant deploy
   ```

Configuring

Required configuration: configuring the PostgreSQL database

PostgreSQL (Version 7.1 or greater) needs to be installed and configured for RFT to work. You can either use the packages which came with your operating system

(RPMs, DEBs, ...) or build from source. We used PostgreSQL version 7.3.2 for our testing and the following instructions are good for the same.

1. Install Postgresql. Instructions on how to install/configure postgresql can be found here[3].
2. Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding -o "-i" switch to postmaster script (This can be init.d script found in /etc/init.d/postgresql or /var/lib/ depending on how you installed postgresql). Follow the intstructions here [4] to start the postmaster with -i option.
3. You will now need to create a postgresql user that would connect to the database. This is usually the account under which the container is running. You can create a postgresql user by running the following command: `su postgres createuser globus` . If you get the following error : `psql: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket "/tmp/.s.PGSQL.5432"?` This generally means that 1. either your postmaster is not started with -i option or 2. you did'nt restart the postmaster after above mentioned step
4. Now you need to set security on the database you are about to create. You can do it by following the steps below:

   `sudo vi /var/lib/pgsql/data/pg_hba.conf` and append the following line to the file:

   ```
   host rftDatabase "username" "host-ip" 255.255.255.255
   "password"
   ```

   ```
   sudo /etc/init.d/postgresql restart
   ```

5. To create the database that is used for RFT, run (as user globus): `createdb rftDatabase`
6. To populate the RFT database with appropriate schemas, run: `psql -d rftDatabase -f $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql` Now that you have created a database to store RFT's state, the following steps configure RFT to find the database:
7. Open `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml`
8. Find the `dbConfiguration` section under `ReliableFileTransferService` `<service>` section.
9. Change the `connectionString` to point to the machine on which you installed Postgres and name of the database you used in step 2. If you installed Postgres on the same machine as your Globus install, the default should work fine for you.
10. Change the `userName` to the name of the user who owns/created the database and do the same for the password. (It also depends on how you configured your database.)
11. Don't worry about the other parameters in that section. The defaults should work fine for now.
12. Edit the configuration section under `ReliableFileTransferService`. There are two values that can be edited in this section.
13.
    - `backOff` : Time in seconds you want RFT to backoff before a failed transfer is retried by RFT. Default should work fine for now.

- `maxActiveAllowed`: This is the number of transfers the container can do at given point. Default should be fine for now.

Deploying

RFT is deployed as part of a standard toolkit installation. Please refer to System Adminstrator's Guide[5] for details.

Testing

You need to checkout the tests from CVS because RFT tests are not included in the installer. Please follow these steps to run RFT unit tests:

RFT Testing

1. set `$GLOBUS_LOCATION` to point to your Globus install.
2. Start a gridftp server on the machine you are running the tests on default port. This can be done by running:

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

3. Start the container with RFT deployed in it.
4. Edit $GLOBUS_LOCATION/share/globus_wsrf_rft_test/test.properties. Put in appropriate values for properties like authzValue(self or host), HOST - host ip of container, PORT- port on which container is listening, sourceHost,destinationsHost - hostnames of gridftp servers. The default values will work fine if you are running the tests with a standard stand-alone container.
5. The *.xfr files in $GLOBUS_LOCATION/share/globus_wsrf_rft_test/ are the transfer files that will be used in the tests. Again the default values work fine if you followed the instructions so far.
6. Run the following command which will run all the rft unit tests:

   ```
   ant -Dtests.jar=$GLOBUS_LOCATION/lib/globus_wsrf_rft_test.jar -f share/globus_wsrf
   ```

7. Run the following command to generate the test-reports in html form:

   ```
    ant -f  share/globus_wsrf_rft_test/runtests.xml generateTestRe-
   port
   ```

Security Considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `etc/<gar>/` directory) contains private information such as database passwords and username. Ensure that these configuration files are only readable by the user that is running the container.

The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Access of information stored in the database

RFT stores the transfer request in a database. Proper security measures need to be taken to protect the access of the data by granting/revoking appropriate permissions on tables that are created for RFT use and other steps that are appropriate and consistent with site specific security measures.

Permissions of persistent data

RFT uses subscription persistence API from GT4 core to store all of its subscription data under the `~/.globus/pe rsisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Troubleshooting

PostgreSQL not configured

Problem: If RFT is not configured properly to talk to a PostgreSQL database, you will see this message displayed on the console when you start the container :

```
"Error creating RFT Home: Failed to connect to database ...
Until this is corrected all RFT request will fail and all GRAM jobs that re-
quire staging will fail".
```

Solution: Usual mistake is Postmaster is not accepting TCP connections which means that you must restart Postmaster with -i option (see the Section called *Required configuration: configuring the PostgreSQL database*).

More verbose error messages

Problem: Make RFT print more verbose error messages:

Solution: Edit $GLOBUS_LOCATION/container-log4j.properties and add following line to it: `log4j.category.org.globus.transfer=DEBUG` . For more verbosity add `log4j.category.org.globus.ftp=DEBUG` which will print out Gridftp messages too.

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 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 again. This failure is treated as a container-wide backoff for the server in question. What that 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.

Failure modes that are not addressed:

- Running out of disk space for the database.

Usage statistics collection by the Globus Alliance

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 is installed
- Total number of bytes transferred by RFT since RFT is 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 GRAM component. Nevertheless, if you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration[6] for instructions.

Also, please see our policy statement [7] on the collection of usage statistics.

Notes

1. ../../admin/docbook/
2. ../../admin/docbook/
3. http://www.postgresql.org/docs/manuals/
4. http://www.postgresql.org/docs/7.4/static/postmaster-start.html
5. ../../admin/docbook/

6. http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets
7. http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 4. GT 4.0 Reliable File Transfer (RFT) Service: User's Guide

Introduction

RFT Service implementation in GT 4.0 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 4.0 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.

Command-line tools

Please see the RFT Command Reference.

Graphical user interfaces

There is no GUI for RFT service in this release.

Troubleshooting

Troubleshooting tips

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

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 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 again. This failure is treated as a container-wide backoff for the server in question. What that 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.

Failure modes that are not addressed:

- Running out of disk space for the database.

Usage statistics collection by the Globus Alliance

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 is installed
- Total number of bytes transferred by RFT since RFT is 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 GRAM component. Nevertheless, if you wish to disable this feature, please see the Java WS Core System Administrator's Guide section on Usage Statistics Configuration[1] for instructions.

Also, please see our policy statement [2] on the collection of usage statistics.

Notes

1. http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets
2. http://www-unix.globus.org/toolkit/docs/4.0/Usage_Stats.html

Chapter 5. GT 4.0 Reliable File Transfer (RFT) Service: Developer's Guide

Introduction

RFT Service implementation in GT 4.0 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 4.0 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.

Before you begin

Feature summary

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports MLST command.
- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option

- Different subject names for source and destination GridFTP servers for authorization mechanism.
- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Tested platforms

Tested platforms for RFT:

- Linux

- Fedora Core 1 i686
- Fedora Core 3 i686
- RedHat 7.3 i686
- RedHat 9 x86
- Debian Sarge x86
- Debian 3.1 i686

Backward compatibility summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

Technology dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Delegation Service
- Service Groups

- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Security considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `etc/<gar>/` directory) contains private information such as database passwords and username. Ensure that these configuration files are only readable by the user that is running the container.

The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Access of information stored in the database

RFT stores the transfer request in a database. Proper security measures need to be taken to protect the access of the data by granting/revoking appropriate permissions on tables that are created for RFT use and other steps that are appropriate and consistent with site specific security measures.

Permissions of persistent data

RFT uses subscription persistence API from GT4 core to store all of its subscription data under the `~/.globus/pe rsisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Architecture and design overview

A design doc can be found here[1].

Public Interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in the public interface guide[2].

Usage scenarios

Transferring large datasets using GridFTP

RFT is primarily used to reliably transfer large datasets using GridFTP. If you are a developer and would like to use RFT, the following steps would help you to do that.

- Contact the Delegation Factory Service and get an EPR for the Delegation Resource that contains your delegated credential.

```
public static EndpointReferenceType
       delegateCredential(String host, String port) throws Exception {
       ClientSecurityDescriptor desc = new ClientSecurityDescriptor();
       // Credential to sign with, assuming default credential
       GlobusCredential credential = GlobusCredential.getDefaultCredential();
       desc.setGSITransport(Constants.GSI_TRANSPORT)
       Util.registerTransport();
       desc.setAuthz('host');

       String factoryUrl = PROTOCOL + "://" + host + ":"
                       + port + SERVICE_URL_ROOT
                       + DelegationConstants.FACTORY_PATH;

       // lifetime in seconds
       int lifetime = TERM_TIME * 60;

       // Get the public key to delegate on.
       EndpointReferenceType delegEpr = AddressingUtils
               .createEndpointReference(factoryUrl, null);
       X509Certificate[] certsToDelegateOn = DelegationUtil
               .getCertificateChainRP(delegEpr, desc);
       X509Certificate certToSign = certsToDelegateOn[0];
       return DelegationUtil.delegate(factoryUrl,
                   credential, certToSign, lifetime, false,
                   desc);
   }
```

- Now construct a TransferRequestType Object :

```
TransferType[] transferArray = new TransferType[1];
transferArray[0] = new TransferType();
transferArray[0].setSourceUrl("gsiftp://foo/bar");
transferArray[0].setDestinationUrl("gsiftp://blah/");
RFTOptionsType rftOptions = new RFTOptionsType();
rftOptions.setBinary(true);
// You can set more options like parallel streams, buffer sizes etc
// Refer to Public Interface guide of RFT for more details
TransferRequestType request = new TransferRequestType();
request.setRftOptions(rftOptions);
request.setTransfer(transferArray);
request.setTransferCredentialEndpoint(delegateCredential(host,port));
```

- Now contact RFT factory and create RFT resource

```
  public static EndpointReferenceType createRFT(String rftFactoryAddress,
        BaseRequestType request)
throws Exception {
    endpoint = new URL(rftFactoryAddress);
    factoryPort = rftFactoryLocator
            .getReliableFileTransferFactoryPortTypePort(endpoint);
    CreateReliableFileTransferInputType input =
        new CreateReliableFileTransferInputType();
    //input.setTransferJob(transferType);
    if(request instanceof TransferRequestType) {
        input.setTransferRequest((TransferRequestType)request);
    } else {
        input.setDeleteRequest((DeleteRequestType)request);
    }
    Calendar termTime = Calendar.getInstance();
    termTime.add(Calendar.HOUR, 1);
    input.setInitialTerminationTime(termTime);
    setSecurity((Stub)factoryPort);
```

```
        CreateReliableFileTransferOutputType response = factoryPort
                .createReliableFileTransfer(input);

        return response.getReliableTransferEPR();
    }
```

- Now contact RFT service Implementation and call start to actually start the transfer:

```
ReliableFileTransferPortType rft = rftLocator
                .getReliableFileTransferPortTypePort(rftepr);
        setSecurity((Stub)rft);

        //For secure notifications
        subscribe(rft);
        System.out.println("Subscribed for overall status");
        //End subscription code
        Calendar termTime = Calendar.getInstance();
        termTime.add(Calendar.MINUTE, TERM_TIME);
        SetTerminationTime reqTermTime = new SetTerminationTime();
        reqTermTime.setRequestedTerminationTime(termTime);
        System.out.println("Termination time to set: " + TERM_TIME
                + " minutes");
        SetTerminationTimeResponse termRes = rft
                .setTerminationTime(reqTermTime);
        StartOutputType startresp = rft.start(new Start());
```

Deleting a set of files and directories using GridFTP

RFT can also be used to delete a set of files and directories using GridFTP server. The following steps depict how to:

- Contact the Delegation Factory Service and get an EPR for the Delegation Resource that contains your delegated credential.

```
 public static EndpointReferenceType
        delegateCredential(String host, String port) throws Exception {
        ClientSecurityDescriptor desc = new ClientSecurityDescriptor();
        // Credential to sign with, assuming default credential
        GlobusCredential credential = GlobusCredential.getDefaultCredential();
        desc.setGSITransport(Constants.GSI_TRANSPORT)
        Util.registerTransport();
        desc.setAuthz('host');

        String factoryUrl = PROTOCOL + "://" + host + ":"
                        + port + SERVICE_URL_ROOT
                        + DelegationConstants.FACTORY_PATH;

        // lifetime in seconds
        int lifetime = TERM_TIME * 60;

        // Get the public key to delegate on.
        EndpointReferenceType delegEpr = AddressingUtils
                .createEndpointReference(factoryUrl, null);
        X509Certificate[] certsToDelegateOn = DelegationUtil
                .getCertificateChainRP(delegEpr, desc);
        X509Certificate certToSign = certsToDelegateOn[0];
        return DelegationUtil.delegate(factoryUrl,
                    credential, certToSign, lifetime, false,
                    desc);
    }
```

- Now construct a DeleteRequestType Object :

```
DeleteType[] deleteArray = new DeleteType[1];
deleteArray[0] = new DeleteType();
deleteArray[0].setFile("gsiftp://foo/bar");
DeleteOptionsType deleteOptions = new DeleteOptionsType();
deleteOptions.setSubjectName("SUBJECT-NAME");
DeleteRequestType request = new DeleteRequestType();
request.setDeleteOptions(deleteOptions);
request.setDeletion(deleteArray);
request.setTransferCredentialEndpoint(delegateCredential(host,port));
```

- Now contact RFT factory and create RFT resource

```
  public static EndpointReferenceType createRFT(String rftFactoryAddress,
        BaseRequestType request)
throws Exception {
    endpoint = new URL(rftFactoryAddress);
    factoryPort = rftFactoryLocator
            .getReliableFileTransferFactoryPortTypePort(endpoint);
    CreateReliableFileTransferInputType input =
        new CreateReliableFileTransferInputType();
    //input.setTransferJob(transferType);
    if(request instanceof TransferRequestType) {
        input.setTransferRequest((TransferRequestType)request);
    } else {
        input.setDeleteRequest((DeleteRequestType)request);
    }
    Calendar termTime = Calendar.getInstance();
    termTime.add(Calendar.HOUR, 1);
    input.setInitialTerminationTime(termTime);
    setSecurity((Stub)factoryPort);
    CreateReliableFileTransferOutputType response = factoryPort
            .createReliableFileTransfer(input);

    return response.getReliableTransferEPR();
}
```

- Now contact RFT service Implementation and call start to actually start the transfer:

```
ReliableFileTransferPortType rft = rftLocator
                .getReliableFileTransferPortTypePort(rftepr);
        setSecurity((Stub)rft);

        //For secure notifications
        subscribe(rft);
        System.out.println("Subscribed for overall status");
        //End subscription code
        Calendar termTime = Calendar.getInstance();
        termTime.add(Calendar.MINUTE, TERM_TIME);
        SetTerminationTime reqTermTime = new SetTerminationTime();
        reqTermTime.setRequestedTerminationTime(termTime);
        System.out.println("Termination time to set: " + TERM_TIME
                + " minutes");
        SetTerminationTimeResponse termRes = rft
                .setTerminationTime(reqTermTime);
        StartOutputType startresp = rft.start(new Start());
```

Tutorials

There are no tutorials available at this point.

Debugging

A standard way to debug RFT is to make container print out more verbose error messages. You can do that by doing the following steps:

Edit $GLOBUS_LOCATION/container-log4j.properties and add following line to it: `log4j.category.org.globus.transfer=DEBUG` . For more verbosity add `log4j.category.org.globus.ftp=DEBUG` which will print out Gridftp messages too.

Troubleshooting

Database configuration

Database configuration is the most complicated and important part of RFT setup. You can find more instructions on troubleshooting at the Section called *Troubleshooting* in Chapter 3.

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 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 again. This failure is treated as a container-wide backoff for the server in question. What that 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.

Failure modes that are not addressed:

- Running out of disk space for the database.

Related Documentation

- Lessons learned producing an OGSI compliant Reliable File Transfer Service (pdf)[3]
- Reliable Data Transport: A Critical Service for the Grid (pdf)[4]

Notes

1. Protocol_overview.doc
2. RFT_Public_Interfaces.html
3. http://www-unix.mcs.anl.gov/%7Ekeahey/DBGS/DBGS_files/dbgs_papers/allcock.pdf
4. http://www.doc.ic.ac.uk/%7Esjn5/GGF/GGF11/BGBS-Allcock.pdf

Chapter 6. GT 4.0 Component Fact Sheet: Reliable File Transfer (RFT) Service

Brief component overview

The Reliable Transfer Service (RFT) Service implementation in 4.0 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters over GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consists of a set of third-party gridftp transfers) to RFT Factory service. 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 resource the user created 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 WS-RF command line clients and other resource properties.

Summary of features

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports MLST command.
- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option
- Different subject names for source and destination GridFTP servers for authorization mechanism.

- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Usability summary

Usability improvements for RFT:

- The command-line RFT clients print out more detailed and accurate fault information for all commonly occuring errors.

Backward compatibility summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

Technology dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Delegation Service
- Service Groups
- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Tested platforms

Tested platforms for RFT:

- Linux

- Fedora Core 1 i686
- Fedora Core 3 i686
- RedHat 7.3 i686
- RedHat 9 x86
- Debian Sarge x86
- Debian 3.1 i686

Associated standards

Associated standards for RFT:

- WSRF[1]
- WS-Addressing[2]
- WS-Security[3]

For More Information

Click here[4] for more information about this component.

Notes

1. http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ServiceGroup-1.2-draft-02.pdf
2. http://msdn.microsoft.com/ws/2004/03/ws-addressing
3. http://msdn.microsoft.com/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnglobspec/html/wssecurspecindex.asp
4. index.html

Chapter 7. GT 4.0 RFT Public Interface Guide

Semantics and syntax of APIs

Programming Model Overview

The Reliable Transfer Service (RFT) is a WSRF based service that provides interfaces for controlling and monitoring third party file transfers using GridFTP servers. The client controlling the transfers (in this case RFT) is hosted inside of a Grid service so it can be managed using the soft state model. It is essentially a reliable and recoverable version of the GT2 `globus-url-copy` tool and more. In 4.0 RFT can also perform file deletion, recursive directory deletion operations. It is also used by GRAM to perform all the staging operations and cleanup operations.

Component API

Some relevant API :

- Service API[1]
- Common API[2]
- Client API[3]

Semantics and syntax of the WSDL

Protocol overview

RFT Service implementation in 4.0 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 4.0 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.

Operations

Please find below operations of both RFT Factory and RFT Service Implementation.

RFT Factory Service

Used to create a Reliable File Transfer resource. The operations exposed by the factory are as follows:

- `createReliableFileTransfer`: Creates a Reliable File Transfer Resource.

  - Input Parameters: Initial Termination time , Transfer Request or Delete Request
  - Output parameters:&nbsp; Termination time, Current time, Endpoint reference of the Resource created. (This should be stored by the user as it is needed to query the status of the resource and to perform any further operations on the resource.
  - Fault: createReliableFileTransferFault:

RFT Service

Used to manage the Resource created using the RFT Factory Service. The operations exposed by the service are as follows:

- `start:` Starts executing the transfers/deletes
  - Input&nbsp; Parameters: None
  - Output Parameters: None
  - Fault: RepeatedlyStartedFault:
- `getStatus:` To get the status of a particular file.
  - Input Parameters:&nbsp; A source URL of the file that is part of the request.
  - Output Parameters: `Transfer Status Type`
  - Fault: RFTDatabaseFault
- `getStatusSet` : To get the status of a set of files in a request
  - Input Parameters:&nbsp; int&nbsp; from (the relative position of the transfer in the request)&nbsp; and int offset (Number of files queried)
  - Output Parameters:&nbsp; An array of `TransferStatusType`
  - Fault: RFTDatabaseFault
- cancel:&nbsp; To cancel a transfer that is part of a resource.
  - Input Parameters : int from (the relative position of the transfer in the request) int to
  - Output Parameters: None
  - Fault: RFTDatabaseFault

Resource Properties:

Resource Properties of RFT Factory (which acts both as a resource and a service at the same time) and RFT Resource are found below:

RFT Factory Service

- `ActiveResourceInstances`: A dynamic resource property of total number of active rft resources in the container at a given point of time.
- `TotalNumberOfTransfers`: A dynamic resource property of total number of transfers/deletes performed since the RFT service is deployed in this container

- `TotalNumberOfActiveTransfers`: A dynamic resource property of number of active transfers across all rft resources in a container at a given point of time.
- `TotalNumberOfBytesTransferred`: A dynamic resource property of total number of bytes transferred by all RFT resources created since the deployment of the service.
- `RFTFactoryStartTime`: Time when the service was deployed in the container. Used to calculate uptime.
- `DelegationServiceEPR`: The end point reference of the Delegation resource that holds the delegated credential used in executing the resource.

RFT Service

- `OverallStatus`: This is a complex type providing overall status of a RFT resource by providing number of transfers pending, active, finished, retrying, failed, cancelled. Each of those values can be obtained by invoking getTransfers(Finished/Active/Failed/Restarted/Pending/Cancelled) on OverallStatus Resource Property.Note that this Resource Property gets updated every time one of the transfers changes state, so there can be and will be more than one update, in the life time of a RFT resource, if you subscribe to this RP. This Resource Property also includes the last fault (if thrown) from a transfer and can be accessed by invoking getFault on OverallStatus. This will indicate why a transfer has failed.
- `RequestStatus`: This is complex type resource property providing status of a RFT resource in form of Pending/Active/Done/Failed. The status can be obtained from RequestStatusType by invoking getRequestStatus(). This will result in one of four status strings(Pending/Active/Done/Failed/Cancelled). This RP also contains a fault that denotes the last fault in a RFT resource and can be accessed by invoking getFault(). If a client is subscribed to this RP there will be only be 2 updates in the life time of a rft resource (Pending->Active->Done, Pending->Active->Failed, Pending->Active->Cancelled, Pending->Cancelled).
- `TotalBytes`: provides the total number of bytes transferred by the resource
- `TotalTime`: provides the total time taken to transfer the above mentioned total bytes.

Faults

Faults from RFT Factory Service and RFT service can be found below:

RFT Factory Service

- `createReliableFileTransferFault`: All the errors encountered during the creation of the RFT resource are mapped to this fault. Any security related errors are caught even before the factory and are thrown to the user/client.

RFT Service

- `RepeatedlyStartedFault`: This is raised if a client calls start more than once on a resource.

- `RFTDatabaseFault`: Thrown when the service is unable to find the resource the user/client is querying for.

WSDL and Schema Definition

- Reliable Transfer Factory Port Type[4]
- Reliable Transfer Port Type[5]

You can find links to all the RFT schemas here.[6]

Command-line tools

Please see the RFT Command Reference.

Overview of Graphical User Interface

There is no GUI for RFT service in this release.

Semantics and syntax of domain-specific interface

Please look here[7] for information on how RFT schemas look like.

Configuration interface

Configuration overview

RFT has the following prerequisites:

- Java WS Core[8] - this is built and installed in a default GT 4.0 installation[9].
- a host certificate (see Required Configuration[10].)
- GridFTP[11] Server - GridFTP perfoms the actual file transfer and is built and installed in a default GT 4.0 installation[12].
- PostgreSQL - PostgreSQL is used to store the state of the transfer to allow for restart after failures. The interface to PostgreSQL is JDBC so any DBMS that supports JDBC can be used, although no other has been tested. For instructions on configuring the PostgreSQL database for RFT, click here[13].

RFT can be registered to an MDS index service to facilitiate monitoring and discovery. The MDS documentation contains a note on registering RFT to an Index Service[14].

Syntax of the interface

The security of the service can be configured by modifying the security descriptor[15]. It allows for configuring the credentials that will be used by the service, type of authentication and authorization that needs to be enforced. By default, the following security configuration is installed:

- Credentials set for use by container is used. If that is not specified, default credentials are used.
- GSI Secure conversation authentication is enforced for all methods.

Note: Changing required authentication and authorization method will require suitable changes to the clients that contact this service.

To alter security descriptor configuration, refer to Security Descriptors[16]. The file to be altered is `$GLOBUS_LOCATION/etc/globus_wsrf_rft/security-config.xml`

Environment variable interface

The only Env variable that needs to be set for RFT is GLOBUS_LOCATION in order to run the command line clients, which should be set to globus installation.

Notes

1. http://www-unix.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_service_java/
2. http://www-unix.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_common_java/
3. http://www-unix.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_client_java/
4. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_factory_port_type.wsdl?rev=1 type=text/vnd.viewcvs-markup
5. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_port_type.wsdl?rev=1.14&only type=text/vnd.viewcvs-markup
6. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/
7. http://www-unix.globus.org/toolkit/docs/4.0/data/rft/rft_job_description.html
8. http://www-unix.globus.org/toolkit/docs/4.0/common/javawscore/
9. http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/
10. http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/
11. http://www-unix.globus.org/toolkit/docs/4.0/data/gridftp/
12. http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/
13. http://www-unix.globus.org/toolkit/docs/4.0/data/rft/admin-index.html#s-rft-admin-postgresql
14. http://www-unix.globus.org/toolkit/docs/4.0/info/index/admin/registering-rft.html
15. http://www-unix.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html
16. http://www-unix.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html

Chapter 8. GT 4.0 RFT: Quality Profile

Test coverage reports

Not available right now.

Code analysis reports

Not available right now.

Outstanding Issues

You can find list of outstanding bugs in RFT by following this: link[1]

Bug Fixes

- Bug 2749[2]
- Bug 2724[3]
- Bug 2683[4]
- Bug 2662[5]
- Bug 2703[6]
- Bug 2847[7]
- Bug 2826[8]
- Bug 2312[9]
- Bug 2879[10]
- Bug 2930[11]
- Bug 2935[12]
- Bug 2852[13]
- Bug 2986[14]
- Bug 3017[15]
- Bug 2984[16]
- Bug 2965[17]
- Bug 2666[18]
- Bug 2927[19]
- Bug 3072[20]
- Bug 2916[21]
- Bug 2721[22]
- Bug 2999[23]
- Bug 3110[24]
- Bug 3091[25]
- Bug 3130[26]
- Bug 2914[27]

- Bug 3115[28]
- Bug 2956[29]

Performance reports

A recent performance report can be found here[30].

Notes

1. http://bugzilla.globus.org/globus/buglist.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_statu
2. http://bugzilla.globus.org/globus/show_bug.cgi?id=2749
3. http://bugzilla.globus.org/globus/show_bug.cgi?id=2724
4. http://bugzilla.globus.org/globus/show_bug.cgi?id=2683
5. http://bugzilla.globus.org/globus/show_bug.cgi?id=2662
6. http://bugzilla.globus.org/globus/show_bug.cgi?id=2703
7. http://bugzilla.globus.org/globus/show_bug.cgi?id=2847
8. http://bugzilla.globus.org/globus/show_bug.cgi?id=2826
9. http://bugzilla.globus.org/globus/show_bug.cgi?id=2312
10. http://bugzilla.globus.org/globus/show_bug.cgi?id=2879
11. http://bugzilla.globus.org/globus/show_bug.cgi?id=2930
12. http://bugzilla.globus.org/globus/show_bug.cgi?id=2935
13. http://bugzilla.globus.org/globus/show_bug.cgi?id=2852
14. http://bugzilla.globus.org/globus/show_bug.cgi?id=2986
15. http://bugzilla.globus.org/globus/show_bug.cgi?id=3017
16. http://bugzilla.globus.org/globus/show_bug.cgi?id=2984
17. http://bugzilla.globus.org/globus/show_bug.cgi?id=2965
18. http://bugzilla.globus.org/globus/show_bug.cgi?id=2666
19. http://bugzilla.globus.org/globus/show_bug.cgi?id=2927
20. http://bugzilla.globus.org/globus/show_bug.cgi?id=3072
21. http://bugzilla.globus.org/globus/show_bug.cgi?id=2916
22. http://bugzilla.globus.org/globus/show_bug.cgi?id=2721
23. http://bugzilla.globus.org/globus/show_bug.cgi?id=2999
24. http://bugzilla.globus.org/globus/show_bug.cgi?id=3110
25. http://bugzilla.globus.org/globus/show_bug.cgi?id=3091
26. http://bugzilla.globus.org/globus/show_bug.cgi?id=3130
27. http://bugzilla.globus.org/globus/show_bug.cgi?id=2914
28. http://bugzilla.globus.org/globus/show_bug.cgi?id=3115
29. http://bugzilla.globus.org/globus/show_bug.cgi?id=2956
30. rft_scalability_3_9_4.doc

Chapter 9. GT 4.0 Reliable File Transfer (RFT) Service: Migration Guide

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

Migrating from GT2

This does not apply to RFT.

Migrating from GT3

The RFT implementation in GT4 and GT3 are not interoperable as they are built on different GT core implementations. In order to migrate to GT4 RFT, you should follow the installation instructions of GT4 RFT which can be found here[1].

Notes

1. http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook/

I. GT 4.0 RFT Command Reference

rft

Name

`rft` — Submit and monitor a 3rd party GridFTP transfer

Synopsis

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 transferr 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 un-
derlying
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 op-
tion
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 al-
low
you to  complete a transfer you otherwise could not do.

false
```

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

1

#Data Channel Authentication (DCAU): Turns off data channel authenti-
cation 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 trans-
fer 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 spec-
ify "null" here. By default host authorization is performed
/DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
#Grid Subject name of the destination gridftp server. This is used for Au-
thorization 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 tran
fers fails. Used in GRAM when staging.

false
#Maximum number of retries: This is number of times RFT retries a trans-
fer failed with a non-fatal error.

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 Di-
rectory -
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 trans-
fer.

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 and 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 documentation of globus-url-copy here[1]. Also please note that maximum number of transfers command-line client can process before running out of memory is ~21K with default JVM heap size which is 64M in our tests. Please look at Performance reports[2] for more details

Notes

1. http://www-unix.globus.org/toolkit/docs/4.0/data/gridftp/rn01re01.html
2. http://www-unix.globus.org/toolkit/docs/4.0/data/rft/rft_scalability_3_9_4.doc

rft-delete

Name

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

Synopsis

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, 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 with this commandline tool.