GT 4.0 OGSA-DAI

GT 4.0 OGSA-DAI

Table of Contents

List of Tables

Chapter 1. GT 4.0 Development Release Notes for RFT

1.1. 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 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 3.9.5 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.

1.2. Feature Summary

Features new in release 3.9.5

- 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.
- 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.

Other Supported Features

- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- third-party transfers

• Data channel authentication (DCAU)

Deprecated Features

• None

1.3. Bug Fixes

• Bug 2352 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2352)
• Bug 2313 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2313)
• Bug 2479 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2479)
• Bug 2554 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2554)
• Bug 2572 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2572)
• Bug 2622 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2622)
• Bug 2544 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2544)
• Bug 1719 (http://bugzilla.globus.org/globus/show_bug.cgi?id=1719)
• Bug 2237 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2237)
• Bug 2717 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2717)
• Bug 2724 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2724)

1.4. Known Problems

Does not compile with JDK 1.3.1

1.5. 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.

1.6. Tested Platforms

- Linux

1.7. 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

1.8. For More Information

Click here (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/index.html) for more information about this component.

Chapter 2. GT 4.0 RFT : System Administrator's Guide

2.1. 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.

This information is in addition to the basic installation instructions in the GT 4.0 System Administrator's Guide (../../../admin/docbook/).

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. So 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.

2.2. 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 (../../../admin/docbook/). 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:

Table 2-1. TITLE

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`

2.3. Configuring

2.3.1. 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 (http://www.postgresql.org/docs/manuals/).

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 (http://www.postgresql.org/docs/7.4/static/postmaster-start.html) to start the postmaster with -i option.

3. 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 "user name "ip address" 255.255.255.255 trust sudo /etc/init.d/postgresql restart`

4. 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

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.

2.4. Deploying

[information about deploying the component into various containers/environments]

2.5. 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:

Table 2-2. TITLE

1	Run: `cvs co ws-transfer/reliable/service/test`
2	Start a gridftp server on the machine you are running the tests on; use port 5678. This can be done by running: `globus-gridftp-server -s  -p 5678 &`
3	Start the container with RFT deployed in it.

4	`cd ws-transfer/reliable/service/test/unit` Edit test.properties. Put in appropriate values for all the properties.
5	Generate a report of the tests that ran in the current working directory: `ant test`

2.6. Security Considerations

2.6.1. 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.

2.6.2. 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.

2.6.3. 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.

2.6.4. Permissions of user's delegated proxy credential file

User's delegated proxy is stored in a file in /tmp directory (on the host running the container) and appropriate permissions are set on it so it can be accessed only by user running the container. This file is removed by the container after the expiration of RFT resource time to live. If it is removed manually before the expiration of resource lifetime all the transfers will fail. The transfers will also fail to start if there is not enough free space in /tmp dir to store the delegated credential.

2.7. Troubleshooting

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 require staging wi
```

Solution: Usual mistake is Postmaster is not accepting TCP connections which means that you must restart Postmaster with -i option (see Section 2.3.1).

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.

2.8. 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 Adminstrator guide section on Usage Statistics
Configuration (http://www-unix.globus.org/toolkit/docs/development/4.0-
drafts/common/javawscore/admin/#usageStatisticsTargets) for
instructions.

Also, please see our policy statement
(http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/Usage_Stats.html) on the collection of
usage statistics.

Chapter 3. GT 4.0 RFT : User's Guide

3.1. Introduction

RFT Service implementation in 3.9.5 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 3.9.5 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.

3.2. Command-line tools

Reference I, *Commandline Reference*

3.3. Graphical user interfaces

There is no GUI for RFT service in this release.

3.4. Troubleshooting

- 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.

3.5. 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 Adminstrator guide section on Usage Statistics Configuration (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/admin/#usageStatisticsTargets) for instructions.

Also, please see our policy statement (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/Usage_Stats.html) on the collection of usage statistics.

Chapter 4. GT 4.0 RFT : Developer's Guide

4.1. 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.

4.2. Architecture and design overview

A design doc can be found here (../Protocol_overview.doc).

4.3. 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 (../RFT_Public_Interfaces.html).

4.4. Usage scenarios

- Scenario 1: You have a large dataset and you want to make sure all the files are transferred. RFT is used to reliably transfer the data.
- Scenario 2: RFT is also used to perform staging operations in GRAM and to remove files and directories on a gridftp server.

4.5. Tutorials

There are no tutorials available at this point.

4.6. Feature summary

Features new in release 3.9.5

- 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.

- 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.

Other Supported Features

- Recursive directory transfers and deletes.

- Parallel streams

- TCP Buffer Size

- third-party transfers

- Data channel authentication (DCAU)

Deprecated Features

- None

4.7. Tested platforms

Tested platforms for RFT:

- Linux

RFT is currently being tested on:

- Windows

- Mac OS X

4.8. 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

4.9. 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.

4.10. Security considerations

4.10.1. 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.

4.10.2. 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.

4.10.3. 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.

4.10.4. Permissions of user's delegated proxy credential file

User's delegated proxy is stored in a file in /tmp directory (on the host running the container) and appropriate permissions are set on it so it can be accessed only by user running the container. This file is removed by the container after the expiration of RFT resource time to live. If it is removed manually before the expiration of resource lifetime all the transfers will fail. The transfers will also fail to start if there is not enough free space in /tmp dir to store the delegated credential.

4.11. 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.

4.12. Troubleshooting

Database configuration is the most complicated and important part of RFT setup. You can find more instructions on troubleshooting at Section 2.7.

4.13. Related Documentation

- Lessons learned producing an OGSI compliant Reliable File Transfer Service (pdf)
  (http://www-unix.mcs.anl.gov/%7Ekeahey/DBGS/DBGS_files/dbgs_papers/allcock.pdf)

- Reliable Data Transport: A Critical Service for the Grid (pdf)
  (http://www.doc.ic.ac.uk/%7Esjn5/GGF/GGF11/BGBS-Allcock.pdf)

Chapter 5. GT 4.0 Component Fact Sheet: Reliable File Transfer (RFT)

5.1. 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 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 3.9.5 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.

5.2. Summary of features

Features new in release 3.9.5

- 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.
- 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.

Other Supported Features

- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- third-party transfers

- Data channel authentication (DCAU)

Deprecated Features

- None

5.3. Usability summary

Usability improvements for RFT:

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

5.4. 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

5.5. 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.

5.6. Tested platforms

Tested platforms for RFT:

- Linux

RFT is currently being tested on:

- Windows
- Mac OS X

5.7. For More Information

Click here (index.html) for more information about this component.

Chapter 6.

6.1. Semantics and syntax of APIs

6.1.1. 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 3.9.5 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.

6.1.2. Component API

Some relevant API :

- Service API (http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_service_java/)
- Common API (http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_common_java/)
- Client API (http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_client_java/)

6.2. Semantics and syntax of the WSDL

6.2.1. Protocol overview

RFT Service implementation in 3.9.5 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 3.9.5 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.

6.2.2. Operations

...

6.2.3. 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: 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:

6.2.4. 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 Parameters: None

  - Output Parameters: None

  - Fault: RepeatedlyStartedFault:

- `getStatus:` To get the status of a particular file.

  - Input Parameters: 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: int from (the relative position of the transfer in the request) and int offset (Number of files queried)

  - Output Parameters: An array of `TransferStatusType`

  - Fault: RFTDatabaseFault

- cancel: 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

6.2.5. Resource properties

...

6.2.6. 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.

6.2.7. RFT Service

- `RequestStatusProperty` : represents the current state of the resource (Active, Pending, Failed, Finished). It also includes the last fault message encountered while executing the request.

- `OverallStatusProperty: ` provides current state of the transfer by providing number of Transfers done, pending, active, failed, cancelled and retrying. It also contains the fault message, if any, raised during a transfer.

- `TotalBytes`: provides the total number of bytes transferred by the resource

- `TotalTime`: provides the total time taken to transfer the above mentioned total bytes.

6.2.8. Faults

...

6.2.9. 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.

6.2.10. 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.

6.2.11. WSDL and Schema Definition

- Reliable Transfer Factory Port Type (http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_factory_port_type.wsdl?rev=1.15&only_with_tag=g type=text/vnd.viewcvs-markup)
- Reliable Transfer Port Type (http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_port_type.wsdl?rev=1.14&only_with_tag=globus_3_ type=text/vnd.viewcvs-markup)

You can find links to all the RFT schemas here. (http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/)

6.3. Command-line tools

Reference I, *Commandline Reference*

6.4. Overview of Graphical User Interface

There is no GUI for RFT service in this release.

6.5. Semantics and syntax of domain-specific interface

Please look here (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/rft_job_description.html) for information on how RFT schemas look like.

6.6. Configuration interface

6.6.1. Configuration overview

RFT has the following prerequisites:

• Java WS Core (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/)
  - this is built and installed in a default GT 4.0 installation
  (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/docbook/).

• a host certificate (see Required Configuration
  (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/#requiredconfig).)

• GridFTP (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/gridftp/) Server - GridFTP
  perfoms the actual file transfer and is built and installed in a default GT 4.0 installation
  (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/docbook/).

• 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
  (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/admin/#postgresql).

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
(http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/info/index/admin/registering-rft.html).

6.6.2. Syntax of the interface

The security of the service can be configured by modifying the security descriptor (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/authzframe/security_descriptor.html). 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 (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/authzframe/security_descriptor.html). The
file to be altered is `$GLOBUS_LOCATION/etc/globus_wsrf_rft/security-config.xml`

6.7. 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.

Chapter 7. "GT 4.0 RFT: Quality Profile

7.1. Test coverage reports

Not available right now.

7.2. Code analysis reports

Not available right now.

7.3.

You can find list of outstanding bugs in RFT by following this: link
(http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=RFT&component=R
0-0=noop&type0-0-0=noop&value0-0-0=)

7.4. Bug Fixes

- Bug 2352 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2352)
- Bug 2313 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2313)
- Bug 2479 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2479)
- Bug 2554 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2554)
- Bug 2572 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2572)
- Bug 2622 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2622)
- Bug 2544 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2544)
- Bug 1719 (http://bugzilla.globus.org/globus/show_bug.cgi?id=1719)
- Bug 2237 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2237)
- Bug 2717 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2717)
- Bug 2724 (http://bugzilla.globus.org/globus/show_bug.cgi?id=2724)

7.5. Performance reports

A recent performance report can be found here (rft_scalability_3_9_4.doc).

I. Commandline Reference

rft

Name

`rft` — PURPOSE

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>
-port <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 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 "nu
/DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
#Grid Subject name of the destination gridftp server. This is used for Authorization purpos
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 transfer

false
#Maximum number of retries: This is number of times RFT retries a transfer failed with a no

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 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

(http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/gridftp/user/#commandline). 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 (http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/rft_scalability_3_9_4.doc) for more details

rftdelete

Name

`rftdelete` — PURPOSE

Synopsis

rftdelete

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>
-port <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.