

GT 4.0 Reliable File Transfer (RFT) Service

GT 4.0 Reliable File Transfer (RFT) Service

Table of Contents

1. Key Concepts	1
1. Overview of Data Management in GT4	1
2. Data movement	1
3. Data replication	2
4. Higher level data services	4
2. Admin Guide	5
1. Introduction	5
2. Building and Installing	5
3. Configuring	6
4. Using MySQL	9
5. Deploying	10
6. Testing	10
7. Security Considerations	11
8. Troubleshooting	11
9. Usage statistics collection by the Globus Alliance	12
3. User's Guide	14
1. Introduction	14
2. Command-line tools	14
3. Graphical user interfaces	14
4. Troubleshooting	14
5. Usage statistics collection by the Globus Alliance	15
4. Developer's Guide	16
1. Introduction	16
2. Before you begin	16
3. Architecture and design overview	18
4. Public Interface	19
5. Usage scenarios	19
6. Tutorials	23
7. Debugging	23
8. Troubleshooting	23
9. Related Documentation	24
5. Fact Sheet	25
1. Brief component overview	25
2. Summary of features	25
3. Usability summary	26
4. Backward compatibility summary	26
5. Technology dependencies	26
6. Tested platforms	27
7. Associated standards	27
8. For More Information	27
6. Public Interface Guide	28
1. Semantics and syntax of APIs	28
2. Semantics and syntax of the WSDL	28
3. Command-line tools	31
4. Overview of Graphical User Interface	31
5. Semantics and syntax of domain-specific interface	31
6. Configuration interface	34
7. Environment variable interface	34
7. Quality Profile	35
1. Test coverage reports	35
2. Code analysis reports	35

GT 4.0 Reliable File Transfer (RFT)
Service

3. Outstanding Issues	35
4. Bug Fixes	35
5. Performance reports	36
8. Migrating Guide	37
1. Migrating from GT2	37
2. Migrating from GT3	37
I. GT 4.0 RFT Command Reference	38
rft	39
rft-delete	41
9. 4.0.8 Release Notes	42
1. Introduction	42
2. Changes Summary	42
3. Bug Fixes	42
4. Known Problems	42
5. For More Information	42
10. 4.0.7 Release Notes	43
1. Introduction	43
2. Changes Summary	43
3. Bug Fixes	43
4. Known Problems	43
5. For More Information	43
11. 4.0.6 Release Notes	44
1. Introduction	44
2. Changes Summary	44
3. Bug Fixes	44
4. Known Problems	44
5. For More Information	44
12. 4.0.5 Release Notes	45
1. Introduction	45
2. Changes Summary	45
3. Bug Fixes	45
4. Known Problems	45
5. For More Information	45
13. 4.0.4 Release Notes	46
1. Introduction	46
2. Changes Summary	46
3. Bug Fixes	46
4. Known Problems	46
5. For More Information	46
14. 4.0.3 Release Notes	47
1. Introduction	47
2. Changes Summary	47
3. Bug Fixes	47
4. Known Problems	47
5. For More Information	47
15. 4.0.2 Release Notes	48
1. Introduction	48
2. Changes Summary	48
3. Bug Fixes	48
4. Known Problems	49
5. For More Information	49
16. 4.0.1 Release Notes	50
1. Introduction	50
2. Changes Summary	50

GT 4.0 Reliable File Transfer (RFT)
Service

3. Bug Fixes	50
4. Known Problems	50
5. For More Information	50
17. 4.0.0 Release Notes	51
1. Component Overview	51
2. Feature Summary	51
3. Bug Fixes	52
4. Known Problems	53
5. Technology Dependencies	53
6. Tested Platforms	54
7. Backward Compatibility Summary	54
8. For More Information	55

Chapter 1. Data Management: Key Concepts

1. Overview of Data Management in GT4

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

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

2. Data movement

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

2.1. GridFTP

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

The Globus Toolkit provides:

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

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

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

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

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

¹ [../gridftp/user-index.html](#)

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

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

For more information about GridFTP, see:

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

2.2. Reliable File Transfer (RFT) Service

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

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

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

3. Data replication

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

3.1. Replica Location Service (RLS)

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

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

² ../gridftp/

³ http://www.globus.org/alliance/publications/papers/gridftp_final.pdf

⁴ ../rft/

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

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

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

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

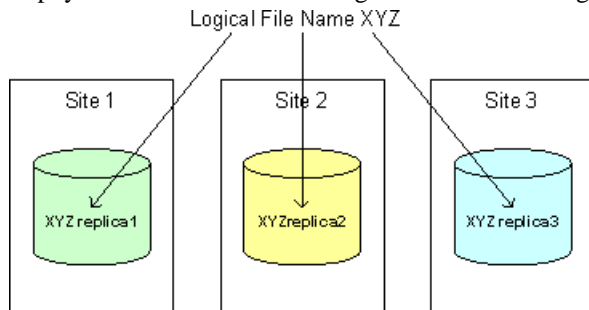


Figure 1. Example of the associations between a logical file name and three replicas on different storage sites.

3.2. Using RLS: An Example

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

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

3.3. For more information

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

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

⁵ rls.html

⁶ ../rls/

4. Higher level data services

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

4.1. Data Replication Service (DRS)

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

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

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

4.1.1. For more information

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

⁷ ../../techpreview/datarep/

Chapter 2. GT 4.0 Reliable File Transfer (RFT) Service: System Administrator's Guide

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.

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.

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](#)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 the 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
```

¹ <http://www.globus.org/toolkit/docs/4.0/admin/docbook/>

² <http://www.globus.org/toolkit/docs/4.0/admin/docbook/>

3. Configuring

3.1. Configuration overview

RFT has the following prerequisites:

- [Java WS Core](#)³ - This is built and installed in a [default GT 4.0 installation](#)⁴.
- A host certificate (see [Required Configuration](#)⁵).
- [GridFTP](#)⁶ Server - GridFTP performs the actual file transfer and is built and installed in a [default GT 4.0 installation](#)⁷.
- 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 others have been tested. For instructions on configuring the PostgreSQL database for RFT, click [here](#)⁸.

3.2. Syntax of the interface

The security of the service can be configured by modifying the [security descriptor](#)⁹. 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 the container are used. If they aren't specified, default credentials are used.
- GSI Secure conversation authentication is enforced for all methods.

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

To alter the security descriptor configuration, refer to [Security Descriptors](#)¹⁰. The file to be altered is `$GLOBUS_LOCATION/etc/globus_wsrft_rft/security-config.xml`.

3.3. 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](#)¹¹.

³ [../common/javawscore/](#)

⁴ [../admin/docbook/](#)

⁵ [../admin/docbook/](#)

⁶ [../data/gridftp/](#)

⁷ [../admin/docbook/](#)

⁸ [../data/rft/admin-index.html#s-rft-admin-postgresql](#)

⁹ [../security/authzframe/security_descriptor.html](#)

¹⁰ [../security/authzframe/security_descriptor.html](#)

¹¹ <http://www.postgresql.org/docs/manuals/>

2. Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding the "-o -i" switch to the postmaster script (This is either the init.d script found in /etc/init.d/postgresql or /var/lib/, depending on how you installed PostgreSQL). Follow the instructions [here](#)¹² to start the postmaster with the -i option.
3. You will now need to create a PostgreSQL user that will 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 either your postmaster is not started with the -i option or you didn't restart the postmaster after the 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 md5
```

 Note: use crypt instead of md5 if you are using PostgreSQL 7.3 or earlier.

```
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 the 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 the `ReliableFileTransferService` `<service>` section.
9. Change the `connectionString` to point to the machine on which you installed PostgreSQL and to the name of the database you used in step 2. If you installed PostgreSQL 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 the 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 milliseconds you want RFT to backoff before a failed transfer is retried by RFT. The default should work fine for now.
 - `maxActiveAllowed`: This is the number of transfers the container can do at given point. The default should be fine for now.

¹² <http://www.postgresql.org/docs/7.4/static/postmaster-start.html>

3.4. RFT auto-registration with default WS MDS Index Service

With a default GT 4.0.1 installation, the RFT service is automatically registered with the default WS MDS Index Service¹³ running in the same container for monitoring and discovery purposes.



Note

If you are using GT 4.0.0, we strongly recommend upgrading to 4.0.1 to take advantage of this capability.

However, if must use GT 4.0.0, or if this registration was turned off and you want to turn it back on, this is how it is configured:

There is a jndi resource defined in `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml` as follows :

```
<resource name="mdsConfiguration"
  type="org.globus.wsrf.impl.servicegroup.client.MDSConfiguration">
  <resourceParams>
    <parameter>
      <name>reg</name>
      <value>true</value>
    </parameter>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrf.jndi.BeanFactory</value>
    </parameter>
  </resourceParams>
</resource>
```

To configure the automatic registration of RFT to the default WS MDS Index Service, change the value of the parameter `<reg>` as follows:

- `true` turns on auto-registration; this is the default in GT 4.0.1.
- `false` turns off auto-registration; this is the default in GT 4.0.0.

3.4.1. Configuring resource properties

By default, the following resource properties (from the RFT Factory Resource) are sent to the default Index Service:

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

¹³ <http://www.globus.org/toolkit/docs/4.0/info/index/>

- `TotalNumberOfBytesTransferred`: A dynamic resource property of the 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.

You can configure which resource properties are sent in RFT's `registration.xml` file, `$GLOBUS_LOCATION/etc/globus_wsrf_rft/registration.xml`. The following is the relevant section of the file:

```
<Content xsi:type="agg:AggregatorContent"
  xmlns:agg="http://mds.globus.org/aggregator/types">

  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">

    <agg:GetMultipleResourcePropertiesPollType
      xmlns:rft="http://www.globus.org/namespaces/2004/10/rft">
      <!-- Specifies that the index should refresh information
        every 60000 milliseconds (once per minute) -->
      <agg:PollIntervalMillis>60000</agg:PollIntervalMillis>

      <!-- specifies that all Resource Properties should be
        collected from the RFT factory -->

      <agg:ResourcePropertyNames>rft:TotalNumberOfBytesTransferred</agg:ResourcePropertyNames>
      <agg:ResourcePropertyNames>rft:TotalNumberOfActiveTransfers</agg:ResourcePropertyNames>
      <agg:ResourcePropertyNames>rft:RFTFactoryStartTime</agg:ResourcePropertyNames>
      <agg:ResourcePropertyNames>rft:ActiveResourceInstances</agg:ResourcePropertyNames>

      <agg:ResourcePropertyNames>rft:TotalNumberOfTransfers</agg:ResourcePropertyNames>

    </agg:GetMultipleResourcePropertiesPollType>
  </agg:AggregatorConfig>
  <agg:AggregatorData/>
</Content>
```

3.5. Registering RFT manually with default WS MDS Index Service

If a third party needs to register an RFT service manually, see [Registering with mds-servicegroup-add](#)¹⁴ in the WS MDS Aggregator Framework documentation.

4. Using MySQL

RFT in 4.0.1 works with MySQL database. A MySQL schema file is provided at `$GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema_mysql.sql`. You will need to download MySQL drivers (MySQL connector/J) from [here](#)¹⁵ and copy the driver jar to `$GLOBUS_LOCATION/lib`. You will also need to make following changes :

¹⁴ <http://www.globus.org/toolkit/docs/4.0/info/aggregator/re01.html#mds-servicegroup-add-registering>

¹⁵ <http://dev.mysql.com/downloads/connector/j/>

1. Create a RFT Database and populate it with mysql schema.

```
mysqladmin -h hostname create rftDatabase -p

mysql -h hostname -D rftDatabase

source share/globus_wsrf_rft/rft_schema_mysql.sql
```

Note: If you are using older (earlier than 4.1) versions of mysql you will need to use `$GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema_mysql_pre4.0.sql` schema to make RFT work. See [Bug 3633](#)¹⁶ for more details. Older versions of the connector/J may also cause problems. See [Bug 5509](#)¹⁷ for more details.

2. Edit `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml` and change values of `connectionString` to `jdbc:mysql://rftDatabase` from `jdbc:postgresql://host/rftDatabase` and `driverName` to `com.mysql.jdbc.Driver` from `org.postgresql.Driver` and `userName` and `password` to whatever was set during creation of users for mysql.

5. Deploying

RFT is deployed as part of a standard toolkit installation. Please refer to the [System Administrator's Guide](#)¹⁸ for details.

5.1. Deploying into Tomcat

RFT has been tested to work without any additional setup when deployed into Tomcat. Please follow these [basic instructions](#)¹⁹ to deploy GT4 services into Tomcat. Note: you need to configure the GT4 install with the needed RFT configuration (like database configuration, etc) before you deploy into Tomcat.

6. Testing

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 the 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 the container is listening), `sourceHost` and `destinationsHost` (hostnames of gridftp servers). The default values will work fine if you are running the tests with a standard stand-alone container started with user credentials (self authorization is done in this case). If the container is started using host credentials, change `authzVal` to `host`. If the gridftp servers you are using for your testing are started as user, you need to supply subject names of the users in `sourceSubject` and `destinationSubject` for authorization with gridftp servers. If both the source and destination servers are started as one user, you can just fill in the user's subject in the `subject` field of `test.properties`. *If you are getting Authentication/Authorization Failures because of mismatched subject names then your `authzVal` and `authType` (uses transport*

¹⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=3633

¹⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5509

¹⁸ `../admin/docbook/`

¹⁹ <http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying>

security by default) need to be changed, depending on how you start the container. If you started the container with the `-nosec` option then you need to change `authType` to `GSI_MESSAGE`, `PROTOCOL` to `http` and `PORT` to `8080`.

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_rft_test
```

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

```
ant -f share/globus_wsrf_rft_test/runtests.xml generateTestReport
```

7. Security Considerations

7.1. Permissions of service configuration files

The service configuration files such as `jndi-config.xml` and `server-config.wsdd` (located under `etc/<gar>/` directory) contain private information such as database passwords and usernames. 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` as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

7.2. Access of information stored in the database

RFT stores the transfer requests 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.

7.3. Permissions of persistent data

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

8. Troubleshooting

8.1. 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 require staging w

Solution: The usual cause is that Postmaster is not accepting TCP connections, which means that you must restart Postmaster with the `-i` option (see [Section 3.3, “Required configuration: configuring the PostgreSQL database”](#)).

8.2. More verbose error messages

Problem: Make RFT print more verbose error messages

Solution: Edit `$GLOBUS_LOCATION/container-log4j.properties` and add the 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.

8.3. 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 a more detailed description of fault-tolerance and recovery in RFT below:

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

8.3.1. Failure modes that are not addressed:

- Running out of disk space for the database.

9. 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 was installed
- Total number of bytes transferred by RFT since RFT was installed
- Total number of files transferred in this RFT Resource

- Total number of bytes transferred in this RFT Resource
- Creation time of this RFT Resource
- Factory Start Time

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the 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](#)²⁰ for instructions.

Also, please see our [policy statement](#)²¹ on the collection of usage statistics.

²⁰ http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTargets

²¹ http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

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

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.

2. Command-line tools

Please see the [RFT Command Reference](#).

3. Graphical user interfaces

There is no GUI for the RFT service in this release.

4. Troubleshooting

4.1. Troubleshooting tips

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

4.2. RFT fault-tolerance and recovery

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

- Source and/or destination GridFTP failures: In this case RFT retries the transfer for a configurable number of maximum attempts with exponential backoff for each retry (the backoff time period is configurable also). If a failure happens in the midst of a transfer, RFT uses the last restart marker that is stored in the database for that transfer and uses it to resume the transfer from the point where it failed, instead of restarting the whole file. This failure is treated as a container-wide backoff for the server in question. What this means is that all other transfers

going to/from that server, across all the requests in a container, will be backed off and retried. This is done in order to prevent further failures of the transfers by using knowledge available in the database.

- Network failures: Sometimes this happens due to heavy load on a network or for any other reason packets are lost or connections get timed out. This failure is considered a transient failure and RFT retries the transfer with exponential backoff for that particular transfer (and not the whole container, as with the source and/or destination GridFTP failures).
- Container failures: These type of failures occur when the machine running the container goes down or if the container is restarted with active transfers. When the container is restarted, it restarts ReliableTransferHome, which looks at the database for any active RFT resources and restarts them.

4.2.1. Failure modes that are not addressed:

- Running out of disk space for the database.

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

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the 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.

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

² http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

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

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.

2. Before you begin

2.1. Feature summary

Features new in GT 4.0

- No new features since GT 3.9.5.

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 transferred successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports the 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 transfers, file transfers and deletes.
- Data channel authentication (DCAU).
- NoTPT option.
- Different subject names for source and destination GridFTP servers for the 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

2.2. 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
- Mac OS X
 - Mac OS X 10.3, 10.4

Tested containers for RFT:

- Java WS Core container
- Tomcat 5.0.30

2.3. Backward compatibility summary

Protocol changes since GT 3.2

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

API changes since GT 3.2

- None

Exception changes since GT 3.2

- None

Schema changes since GT 3.2

- WSDL changes to work with the new Java WS Core

2.4. 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 or later. Not tested with 8.0 yet.

2.5. Security considerations

2.5.1. Permissions of service configuration files

The service configuration files such as `jndi-config.xml` and `server-config.wsdd` (located under `etc/<gar>/` directory) contain private information such as database passwords and usernames. 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` as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

2.5.2. Access of information stored in the database

RFT stores the transfer requests 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.

2.5.3. Permissions of persistent data

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

3. Architecture and design overview

A design doc can be found [here](#)¹.

¹ Protocol_overview.doc

4. Public Interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in the [Public Interface Guide](#)².

5. Usage scenarios

5.1. 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();
```

² RFT_Public_Interfaces.html

```
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 the RFT factory and create an 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 the 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());
```

5.2. 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 the RFT factory and create an 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 the 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());
```

6. Tutorials

There are no tutorials available at this point.

7. Debugging

A standard way to debug RFT is to make the container print out more verbose error messages. You can do this with 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.

8. Troubleshooting

8.1. Database configuration

Database configuration is the most complicated and important part of RFT setup. You can find more instructions on troubleshooting in [Section 8, “Troubleshooting”](#).

8.2. RFT fault-tolerance and recovery

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

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

8.2.1. Failure modes that are not addressed:

- Running out of disk space for the database.

9. Related Documentation

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

³ http://www-unix.mcs.anl.gov/%7Ekeahey/DBGS/DBGS_files/dbgs_papers/allcock.pdf

⁴ <http://www.doc.ic.ac.uk/%7Eesn5/GGF/GGF11/BGBS-Allcock.pdf>

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

1. Brief component overview

The Reliable Transfer Service (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 deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters of the GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consisting of a set of third-party gridftp transfers) to the 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.

2. Summary of features

Features new in GT 4.0

- No new features since GT 3.9.5.

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 transferred successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports the 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 transfers, file transfers and deletes.

- Data channel authentication (DCAU).
- NoTPT option.
- Different subject names for source and destination GridFTP servers for the 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

3. Usability summary

Usability improvements for RFT:

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

4. Backward compatibility summary

Protocol changes since GT 3.2

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

API changes since GT 3.2

- None

Exception changes since GT 3.2

- None

Schema changes since GT 3.2

- WSDL changes to work with the new Java WS Core

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 or later. Not tested with 8.0 yet.

6. 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
- Mac OS X
 - Mac OS X 10.3, 10.4

Tested containers for RFT:

- Java WS Core container
- Tomcat 5.0.30

7. Associated standards

Associated standards for RFT:

- [WSRF](#)¹
- [WS-Addressing](#)²
- [WS-Security](#)³

8. For More Information

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

¹ <http://docs.oasis-open.org/wsr/2004/06/wsr/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 6. GT 4.0 RFT Public Interface Guide

1. Semantics and syntax of APIs

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 GT 4.0 RFT can also perform file deletion and recursive directory deletion operations. It is also used by GRAM to perform all the staging operations and cleanup operations.

1.2. Component API

Some relevant APIs:

- [Service API](#)¹
- [Common API](#)²
- [Client API](#)³

2. Semantics and syntax of the WSDL

2.1. Protocol overview

The 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 an RFT resource by submitting a list of URL pairs of files that need to be transferred/deleted to the RFT Factory service. The user also specifies the time to live for the resource the user is creating to the 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 manages the transfers (the resource). The operations exposed by both the RFT factory and the 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.

2.2. Operations

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

2.2.1. RFT Factory Service

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

¹ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_service_java/

² http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_common_java/

³ http://www.globus.org/api/javadoc-4.0.0/globus_wsrf_rft_client_java/

- `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`.

2.2.2. 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` (the 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) and `int to`.
 - Output Parameters: None
 - Fault: `RFTDatabaseFault`

2.3. Resource Properties

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

2.3.1. RFT Factory Resource Properties

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

2.3.2. RFT Resource Properties

- `OverallStatus`: This is a complex type providing the overall status of an RFT resource by providing the number of transfers pending, active, finished, retrying, failed, and cancelled. Each of these 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 a complex type resource property providing the status of an RFT resource in the 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 an RFT resource (`Pending->Active->Done`, `Pending->Active->Failed`, `Pending->Active->Cancelled`, and `Pending->Cancelled`).
- `TotalBytes`: This provides the total number of bytes transferred by the resource.
- `TotalTime`: This provides the total time taken to transfer the above-mentioned total bytes.

2.4. Faults

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

2.4.1. 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 before the factory and are thrown to the user/client.

2.4.2. RFT Service

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

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

2.5. 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.

3. Command-line tools

Please see the [RFT Command Reference](#).

4. Overview of Graphical User Interface

There is no GUI for the RFT service in this release.

5. Semantics and syntax of domain-specific interface

5.1. Request Schema

Please go [here](#)⁷ to view the entire RFT transfer request schema documentation.

5.2. Request Options

5.2.1. General Options

These options are set in the [transferRequest](#)⁸ and [deleteRequest](#)⁹ elements and apply similarly for each.

- *concurrency*

This denotes number of files in the request that needs to be transferred at one time.

- *maxAttempts*

Maximum number of attempts after transient errors to execute the transfer or deletion before giving up and raising an error.

- *finishBy*

⁴ 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=globus_4_0_0&content-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_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup

⁶ <http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/>

⁷ [rft_job_description.html](#)

⁸ [rft_job_description.html#element_transferRequest](#)

⁹ [rft_job_description.html#element_deleteRequest](#)

(Not Implemented) In future versions of RFT this will be used to enforce time constraints on a transfer.

5.2.2. Transfer Options

These options are set in the `rftOptions` element (see [RFTOptionsType](#)¹⁰ for more details) and are specific to file transfers. They can be specified as defaults for all transfers under the `transferRequest`¹¹ element, and/or individually under each `transfer` element (see [TransferType](#)¹² for more details):

```
<transferRequest>
  <transfer>...</transfer>
  <rftOptions>
    <-- option elements here -->
  </rftOptions>
</transferRequest>
```

AND/OR

```
<transferRequest>
  <transfer>
    ...
    <sourceUrl>
    <destinationUrl>
    ...
    <rftOptions>
      <-- option elements here -->
    </rftOptions>
  </transfer>
</transferRequest>
```

- *binary*
Transfer as a binary file. Default is "true".
- *blockSize*
Specifies the size of the data blocks to use in the transfer.
- *tcpBufferSize*
Specifies the TCP buffer size used for the transfer.
- *notpt*
If set to "true", third-party transfer mode will not be use. Instead, a client thread will be started that will GET data from the source server and and PUT data to the destination server. Default is "false".
- *parallelStreams*

¹⁰ [rft_job_description.html#type_RFTOptionsType](#)

¹¹ [rft_job_description.html#element_transferRequest](#)

¹² [rft_job_description.html#type_TransferType](#)

Specifies the number of parallel streams to use during the transfer. Default is 1.

- *dcau*

Specifies whether or not to use data channel authentication. Default is true.

- *subjectName*

Specifies the credential subject to use for authenticating both the source and destination servers.

- *destinationSubjectName*

Specifies the credential subject to use for authenticating the destination server.

- *sourceSubjectName*

Specifies the credential subject to use for authenticating the source server.

- *userName*

Specifies the username to be used to perform the transfer which sometimes may not be the same as transfer requester.

5.2.3. Deletion Options

These options are set in the `deleteOptions` element (see [DeleteOptionsType](#)¹³ for more details), and are specific to file deletions. They can be specified as defaults for all deletions under the `deleteRequest`¹⁴ element, and/or individually under each `deletion` element (see [DeleteType](#)¹⁵ for more details):

```
<deleteRequest>
  <deletion>...</deletion>
  <deleteOptions>
    <!-- option elements here -->
  </deleteOptions>
</deleteRequest>
```

AND/OR

```
<deleteRequest>
  <deletion>
    ...
    <file>
      <deleteOptions>
        <!-- option elements here -->
      </deleteOptions>
    </deletion>
  </deleteRequest>
```

- *subjectName*

¹³ rft_job_description.html#type_DeleteOptionsType

¹⁴ rft_job_description.html#element_deleteRequest

¹⁵ rft_job_description.html#type_DeleteType

Specifies the credential subject to use for authenticating the target server.

- *userName*

Specifies the username to be used to perform the deletion.

6. Configuration interface

Please see the [Configuring RFT](#).

7. Environment variable interface

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

Chapter 7. GT 4.0 RFT: Quality Profile

1. Test coverage reports

Not available right now.

2. Code analysis reports

Not available right now.

3. Outstanding Issues

You can find list of outstanding bugs in RFT [here](#)¹.

4. 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

¹ http://bugzilla.globus.org/globus/buglist.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&email1=mad-duri%40mcs.anl.gov&emailtype1=exact&emailassigned_to1=1&emailreporter1=1

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

- [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

5. Performance reports

A recent throughput report of RFT can be found [here](#)³⁰.

A recent performance report can be found [here](#)³¹.

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 Comparison.xls

³¹ rft_scalability_3_9_4.doc

Chapter 8. GT 4.0 Reliable File Transfer (RFT) Service: Migration Guide

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

1. Migrating from GT2

This does not apply to RFT.

2. Migrating from GT3

The RFT implementations 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 for GT4, which can be found [here](http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html)¹.

¹ <http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch04.html>

GT 4.0 RFT Command Reference

Name

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

rft

Tool description

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

Command syntax and options

```
rft [-h <hostname or ip-address of the container Defaults to localhost>
-r <port on which the container is listening, Defaults to TCP port 8443>
-l <lifetime of the created resource in minutes Default to 60mins>
-m <security mechanism. Allowed values: 'msg' for secure message or 'conv' for
secure conversation and 'trans' for transport. Defaults to 'trans'.>
-p <protection type, Allowed values 'sig' signature and 'enc' encryption,
Defaults to 'sig' >
-z <authorization mechanism. Defaults to 'host' authorization. Allowed values: 'self' for
-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_wsrft_client/** along with other samples for directory transfers and deletes (lines starting with # are comments):

```
This option when it is set to true means to perform transfer in binary
form, if it is set to false transfer is done in ASCII. Default is binary.
true
#Block size in bytes that is transferred. Default is 16000 bytes.
16000
#TCP Buffer size in bytes
#Specifies the size (in bytes) of the TCP buffer to be used by the underlying
ftp data channels. This is critical to good performance over the WAN. Use the
bandwidth-delay product as your buffer size.

16000

#Notpt (No thirdPartyTransfer): turns third-party transfers off is this option
is set to false (on if set to true).
Site firewall and/or software configuration may prevent a connection
between the two servers (a third party transfer). If this is the case,
RFT will "relay" the data. It will do a GET from the source and a PUT to
the destination. This obviously causes a performance penalty, but will allow
you to complete a transfer you otherwise could not do.

false

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

1

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

true

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

1

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

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

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

false

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

10

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

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

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

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

Limitations

This command line client is very simple and does not do any intelligent parsing of various command line options or of the options in the sample transfer file. It works fine if used in the way documented here. For more information on all these options please refer to the [documentation of globus-url-copy](#)¹. Also, please note that the maximum number of transfers the command-line client can process before running out of memory is ~21K with the default JVM heap size, which was 64M in our tests. Please look at [Performance Reports](#)² for more details.

¹ ../gridftp/rn01re01.html

² rft_scalability_3_9_4.doc

Name

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

rft-delete

Tool description

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

Command and options

```
rft-delete [-h <hostname or ip-address of the container. Defaults to 'localhost'>
-r <port on which the container is listening, Defaults to TCP port 8443>
-l <lifetime for the created resource in minutes. Defaults to 60mins>
-m <security mechanism Allowed values: 'msg' for secure messages, 'conv' for
secure conversation and 'trans' for secure transport. Defaults to
'trans'.>
-p <protection type Allowed values: 'sig' for signature and 'enc' for encryption.Defaults
-z <authorization Defaults to 'host' authorization. Allowed values: 'self' for self autho
-file <filename to write EPR of created Reliable File Transfer Resource>
-f <path to the file that contains list of transfers>
```

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

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

Limitations

The format of the input file to the transfer clients should be exactly like the example above. The client code does not have tolerance for missed lines in the file. This *will be fixed in 4.2*

Chapter 9. GT 4.0.8 Incremental Release Notes: RFT

1. Introduction

These release notes are for the incremental release 4.0.8. It includes a summary of changes since 4.0.7, bug fixes since 4.0.8 and any known problems that still exist at the time of the 4.0.8 release. This page is in addition to the top-level 4.0.8 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.8>.

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

2. Changes Summary

Improved log messages.

3. Bug Fixes

- Fixed a possible deadlock bug by using a second threadpool for request handling (and not sharing threads with transfer handling)
- [Bug 6063](#):² Hash collision issue in RFT connection caching

4. Known Problems

No problems are known to exist at the time of this release

5. For More Information

Click [here](#)³ for more information about this component.

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=6063

³ [index.html](#)

Chapter 10. GT 4.0.7 Incremental Release Notes: RFT

1. Introduction

These release notes are for the incremental release 4.0.7. It includes a summary of changes since 4.0.6, bug fixes since 4.0.7 and any known problems that still exist at the time of the 4.0.7 release. This page is in addition to the top-level 4.0.7 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.7>.

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

2. Changes Summary

A configurable option to control the number of concurrent threads to handle transfer request has been added. This would help prevent the container being overloaded.

3. Bug Fixes

- [Bug 5910](#):²Possible corruption on cached data connections
- [Bug 5919](#):³Problem with user DN based authroization
- [Bug 3121](#):⁴Bug in enforcing maximum active transfers in a container limit
- Fixed problem with 'all or none' option in 4.0.6
- Fixed problem with the back off mechanism for transfer retries in 4.0.6

4. Known Problems

5. For More Information

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

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5910

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5919

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=3121

⁵ [index.html](#)

Chapter 11. GT 4.0.6 Incremental Release Notes: RFT

1. Introduction

These release notes are for the incremental release 4.0.6. It includes a summary of changes since 4.0.5, bug fixes since 4.0.6 and any known problems that still exist at the time of the 4.0.6 release. This page is in addition to the top-level 4.0.6 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.6>.

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

2. Changes Summary

GridFTP Connection Caching code has been added to RFT in 4.0.6. This feature speeds up frequently repeated use of GridFTP transfers by keeping the connections involved open for a period of time. The basic cache characteristics are configurable.

3. Bug Fixes

- [Bug 5642](#):²Merge GridFTP Connection Caching work to the release branch

4. Known Problems

- [Bug 3121](#):³ Bug in enforcing maximum active transfers in a container

5. For More Information

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

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5094

³ http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=3121

⁴ [index.html](#)

Chapter 12. GT 4.0.5 Incremental Release Notes: RFT

1. Introduction

These release notes are for the incremental release 4.0.5. It includes a summary of changes since 4.0.4, bug fixes since 4.0.5 and any known problems that still exist at the time of the 4.0.5 release. This page is in addition to the top-level 4.0.5 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.5>.

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

2. Changes Summary

No changes have been made since 4.0.4, except for bug fixes.

3. Bug Fixes

- [Bug 5094](#):² IgnoreFilePermErr does not work correctly
- [Bug 5146](#):³ RFT Postgres schema issues

4. Known Problems

- [Bug 3121](#):⁴ Bug in enforcing maximum active transfers in a container

5. For More Information

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

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5094

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5146

⁴ http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=3121

⁵ [index.html](#)

Chapter 13. GT 4.0.4 Incremental Release Notes: RFT

1. Introduction

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

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

2. Changes Summary

The only changes for RFT since GT 4.0.2 are bug fixes.

3. Bug Fixes

- [Bug 4299](#):² Error in RFT
- [Bug 4849](#):³ make wrft doesn't install needed dependencies
- [Bug 4953](#):⁴ Incorrect sql to create RFT indices for MySQL

4. Known Problems

- [Bug 3121](#):⁵ Bug in enforcing maximum active transfers in a container

5. For More Information

Click [here](#)⁶ for more information about this component.

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=4299

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=4849

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=4953

⁵ http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=3121

⁶ [index.html](#)

Chapter 14. GT 4.0.3 Incremental Release Notes: RFT

1. Introduction

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

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

2. Changes Summary

The only changes for RFT since GT 4.0.2 are bug fixes.

3. Bug Fixes

The following bugs have been fixed for RFT since GT 4.0.2:

- [Bug 4366](#):² Bad 4.0.2 tag in ws-transfer?
- [Bug 4503](#):³ Incomplete file list
- [Bug 4643](#):⁴ race condition for requests with 2 directory creation
- [Bug 4639](#):⁵ globus_4_0_2 tag, java-only installation

4. Known Problems

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

- [Bug 3121](#):⁶ Bug in enforcing maximum active transfers in a container

5. For More Information

Click [here](#)⁷ for more information about this component.

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=4366

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=4503

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=4643

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=4639

⁶ http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=3121

⁷ [index.html](#)

Chapter 15. GT 4.0.2 Incremental Release Notes: RFT

1. Introduction

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

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

2. Changes Summary

No significant changes occurred in RFT from last version

3. Bug Fixes

The following bugs were fixed for RFT:

- [Bug 2419](#):² Need for RFT Package
- [Bug 3326](#):³ Remove globus_common dependency from RFT setup
- [Bug 3719](#):⁴ RFT Client doesn't retries after a transient transfer error
- [Bug 3685](#):⁵ RFT loosing connection with MySQL
- [Bug 3633](#):⁶ Directory removal not workineth MySQL schema for RFT
- [Bug 3634](#):⁷ Problem with RFT IDs when using MySQL database
- [Bug 3640](#):⁸ RFT Unit Tests - System.err - Badly formatted numbers in ...
- [Bug 3644](#):⁹ MySQL Schema - duplicate primary key constraint name
- [Bug 3646](#):¹⁰ RFT Test - change default authzVal to "host" from "self"
- [Bug 3653](#):¹¹ patch to RFTTest.java - add log4j logging

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=2419

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=3326

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=3719

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=3685

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=3633

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=3634

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=3640

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=3644

¹⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=3646

¹¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=3653

- [Bug 3661](#).¹² rft comman - default port should be 8443 instead of 8080
- [Bug 3662](#).¹³ rft command - fails with NegativeArraySizeException
- [Bug 3717](#).¹⁴ RFT client hangs up if all the transfers fail
- [Bug 3745](#).¹⁵ AllOrNone Notifications
- [Bug 3833](#).¹⁶ Problems with rft-delete
- [Bug 3925](#).¹⁷ Misconfigured database + Index service registration == me...
- [Bug 3943](#).¹⁸ RFT -help text and on-line doc are out of sync and need i...
- [Bug 3944](#).¹⁹ RFT messages on missing input file
- [Bug 3959](#).²⁰ MT issues in RFT?
- [Bug 4225](#).²¹ error deleting a directory
- [Bug 3864](#).²² RFT errors impact GRAM stability

4. Known Problems

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

- [Bug 3121](#).²³ Bug in enforcing maximum active transfers in a container

5. For More Information

Click [here](#)²⁴ for more information about this component.

¹² http://bugzilla.globus.org/globus/show_bug.cgi?id=3661

¹³ http://bugzilla.globus.org/globus/show_bug.cgi?id=3662

¹⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=3717

¹⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=3745

¹⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=3833

¹⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=3925

¹⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=3943

¹⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=3944

²⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=3959

²¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=4225

²² http://bugzilla.globus.org/globus/show_bug.cgi?id=3864

²³ http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=3121

²⁴ [index.html](#)

Chapter 16. GT 4.0.1 Incremental Release Notes: RFT

1. Introduction

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

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

2. Changes Summary

The following change has occurred for RFT:

- RFT database schema has been updated to make it work with mySQL and other databases. All these changes are backwards compatible.

3. Bug Fixes

The following bugs were fixed for RFT:

- [Bug 3459](#):² RFT/GRAM cleanup failure
- [Bug 2764](#):³ Remove Postgresql specific queries from RFT
- [Bug 3288](#):⁴ Database Connection pooling optimization
- [Bug 3580](#):⁵ Directory removal not working

4. Known Problems

No problems are known to exist for RFT at the time of the 4.0.1 release.

5. For More Information

Click [here](#)⁶ for more information about this component.

¹ http://www.globus.org/toolkit/docs/4.0/data/rft/RFT_Release_Notes.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=3459

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=2764

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=3288

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=3580

⁶ [index.html](#)

Chapter 17. GT 4.0 Release Notes: Reliable File Transfer (RFT) Service

1. Component Overview

The Reliable Transfer Service (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 deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters of the GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consisting of a set of third-party gridftp transfers) to the 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.

2. Feature Summary

Features new in GT 4.0

- No new features since GT 3.9.5.

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 transferred successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports the 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 transfers, file transfers and deletes.

- Data channel authentication (DCAU).
- NoTPT option.
- Different subject names for source and destination GridFTP servers for the 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

3. 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

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

- [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

4. 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.

5. Technology Dependencies

RFT depends on the following GT components:

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

- 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 or later. Not tested with 8.0 yet.

6. 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
- Mac OS X
 - Mac OS X 10.3, 10.4

Tested containers for RFT:

- Java WS Core container
- Tomcat 5.0.30

7. Backward Compatibility Summary

Protocol changes since GT 3.2

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

API changes since GT 3.2

- None

Exception changes since GT 3.2

- None

Schema changes since GT 3.2

- WSDL changes to work with the new Java WS Core

8. For More Information

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

³⁰ <http://www.globus.org/toolkit/docs/4.0/data/rft/index.html>