

# **GT 4.2.0 Reliable File Transfer (RFT) Service: System Administrator's Guide**

DRAFT

# GT 4.2.0 Reliable File Transfer (RFT) Service: System Administrator's Guide

## Introduction

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

### **Important**

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in [Installing GT 4.2.0](#). 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.

DRAFT

# Table of Contents

How-to Index .....	5
1. Building and Installing .....	1
2. Configuring RFT .....	2
1. Configuration overview .....	2
2. Syntax of the interface .....	2
3. Optional configuration: configuring the PostgreSQL database .....	2
4. Optional configuration: configuring the MySQL database .....	3
5. RFT auto-registration with default WS MDS Index Service .....	4
6. Registering RFT manually with default WS MDS Index Service .....	5
3. Using MySQL .....	6
4. Deploying .....	7
1. Deploying into Tomcat .....	7
5. Testing .....	8
6. Security Considerations .....	9
1. Reliable Transfer Service (RFT) Security Considerations .....	9
7. Debugging .....	10
1. Logging in Java WS Core .....	10
2. Specifying verbose error messages .....	11
8. Troubleshooting .....	12
1. Errors .....	12
2. RFT fault-tolerance and recovery .....	12
9. Usage statistics collection by the Globus Alliance .....	13
1. Usage statistics sent by RFT .....	13
Index .....	14

---

## List of Tables

8.1. Reliable File Transfer (RFT) Errors ..... 12

DRAFT

# How-to Index

## C

- configuration interface,
  - auto-registration with WS MDS Index,
  - overview,
  - resource properties,
  - settings,
- configuring,
  - auto-registration to WS MDS Index Service,
  - MySQL database,
  - PostgreSQL database,
  - registering manually to WS MDS Index Service,
  - resource properties,

## D

- debugging
  - logging,
- deploying,
  - into Tomcat,

## E

- errors,

## F

- fault tolerance,

## I

- installing,
  - latest code from CVS (advanced users only),

## L

- logging
  - CEDPS-compliant,
  - debugging,

## M

- MySQL
  - using,

## R

- recovery,

## S

- security considerations,

## T

- testing,
- troubleshooting,

- installation,

## U

- usage statistics,

# Chapter 1. Building and Installing

RFT is built and installed as part of a default [GT 4.2.0 installation](#). 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
```

# Chapter 2. Configuring RFT

## 1. Configuration overview

RFT has the following prerequisites:

- [Java WS Core](#) - This is built and installed in a [Installing GT 4.2.0](#).
- A host certificate (see [Installing GT 4.2.0](#)).
- [GridFTP](#) - GridFTP performs the actual file transfer and is built and installed in a [Installing GT 4.2.0](#).
- 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, see below. .

## 2. Syntax of the interface

The security of the service can be configured by modifying the [security descriptor](#). 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 descriptor](#). The file to be altered is `$GLOBUS_LOCATION/etc/globus_wsrf_rft/security-config.xml`.

## 3. Optional configuration: configuring the PostgreSQL database

PostgreSQL (version 7.1 or greater) can be used with RFT but is no longer a requirement. 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](#)<sup>1</sup>.
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](#)<sup>2</sup> 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`

<sup>1</sup> <http://www.postgresql.org/docs/manuals/>

<sup>2</sup> <http://www.postgresql.org/docs/7.4/static/postmaster-start.html>

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 $GLOBUS_LOCATION/share/globus_wsrft_rft/rft_schema.sql`. Now that you have created a database to store RFT's state, the following steps configure RFT to find the database:
7. Open `$GLOBUS_LOCATION/etc/globus_wsrft_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 seconds 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.

## 4. Optional configuration: configuring the MySQL database

If you prefer MySQL to Postgres or derby you can use it with RFT instead. Instructions on how to this can be found at [here](#).<sup>3</sup>

---

<sup>3</sup> <http://www.globus.org/toolkit/docs/4.2/4.2.0/data/rft/admin/rft-admin-mysql.html>

## 5. RFT auto-registration with default WS MDS Index Service

With a default GT 4.2.0 installation, the RFT service is automatically registered with the default WS MDS Index Service running in the same container for monitoring and discovery purposes.

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

```
<resource name="mdsConfiguration"
  type="org.globus.wsrfl.impl.servicegroup.client.MDSConfiguration">
  <resourceParams>
    <parameter>
      <name>reg</name>
      <value>>true</value>
    </parameter>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrfl.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 4.2.0.
- `false` turns off auto-registration.

### 5.1. 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.
- `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\_wsrft\_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>
```

## 6. 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](#) in the WS MDS Aggregator Framework documentation.

# Chapter 3. Using MySQL

RFT in 4.2.0 works with MySQL database. A MySQL schema file is provided at `$GLOBUS_LOCATION/share/globus_wsrft_rft/rft_schema_mysql.sql`. You will need to download MySQL drivers (MySQL connector/J 3.1 and not 3.0) from [here](#)<sup>1</sup> and copy the driver jar to `$GLOBUS_LOCATION/lib`. You will also need to make following changes :

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_wsrft_rft/rft_schema_mysql.sql
```



## Note

If you are using older ( earlier than 4.1 ) versions of MySQL, you will need to use the schema `$GLOBUS_LOCATION/share/globus_wsrft_rft/rft_schema_mysql_pre4.0.sql` to make RFT work. See [Bug 3633](#)<sup>2</sup> for more details.

2. Edit `$GLOBUS_LOCATION/etc/globus_wsrft_rft/jndi-config.xml` and change the following values:
  - for *connectionString*, change `jdbc:postgresql://host/rftDatabase` to `jdbc:mysql:///rftDatabase`
  - for *driverName*, change `org.postgresql.Driver` to `com.mysql.jdbc.Driver`
  - and change the *userName* and *password* to whatever was set when users were created for MySQL.

---

<sup>1</sup> <http://dev.mysql.com/downloads/connector/j/3.1.html>

<sup>2</sup> [http://bugzilla.globus.org/globus/show\\_bug.cgi?id=3633](http://bugzilla.globus.org/globus/show_bug.cgi?id=3633)

# Chapter 4. Deploying

RFT is deployed as part of a [standard toolkit installation](#).

## 1. Deploying into Tomcat

RFT has been tested to work without any additional setup when deployed into Tomcat. Please follow these [basic instructions](#) 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.

DRAFT

# Chapter 5. 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 `authzValue` 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 security by default) need to be changed, depending on how you started 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_
```

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

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

# Chapter 6. Security Considerations

## 1. Reliable Transfer Service (RFT) Security Considerations

### 1.1. 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.

### 1.2. 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.

### 1.3. 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.

# Chapter 7. Debugging

The following information is about sys admin logging in Java WS Core (on which RFT is built).

## 1. Logging in Java WS Core

The following information applies to Java WS Core and all services built on Java WS Core.

Java WS Core server side has two types of loggers. One logger is used for development logging and by default writes to standard out. The other logger includes system administration information and is [CEDPs best practices](#)<sup>1</sup> compliant.

On client side, only developer logging is available and is configured using `log4j.properties`.

### 1.1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)<sup>2</sup> API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)<sup>3</sup> as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)<sup>4</sup>.

#### 1.1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)<sup>5</sup>. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

#### 1.1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

---

<sup>1</sup> <http://cedps.net/index.php/LoggingBestPractices>

<sup>2</sup> <http://jakarta.apache.org/commons/logging/>

<sup>3</sup> <http://logging.apache.org/log4j/>

<sup>4</sup> [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

<sup>5</sup> <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

## 1.2. Configuring system administration logs

The specific logger to edit will be `log4j.logger.sysadmin` in `$GLOBUS_LOCATION/container-log4j.properties`. There you can configure the following properties:

```
log4j.appender.infoCategory=org.apache.log4j.RollingFileAppender
log4j.appender.infoCategory.Threshold=INFO
log4j.appender.infoCategory.File=var/containerLog
log4j.appender.infoCategory.MaxFileSize=10MB
log4j.appender.infoCategory.MaxBackupIndex=2
```

Above implies the logging file is rolling with each file size limited to 10MB and the logging information is stored in `$GLOBUS_LOCATION/var/containerLog`.

## 1.3. Sample log file

The [sample log file](#)<sup>6</sup> contains many log entries for various scenarios in the Java WS container.

## 2. Specifying verbose error messages

Edit `$GLOBUS_LOCATION/container-log4j.properties` and add the following line to it:

```
log4j.category.org.globus.transfer=DEBUG
```

For even more verbosity add

```
log4j.category.org.globus.ftp=DEBUG
```

, which will also print out GridFTP messages.

---

<sup>6</sup> <http://www.globus.org/toolkit/docs/4.2/4.2.0/common/javawscore/sample-container-log.txt>

# Chapter 8. Troubleshooting

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

## 1. Errors

**Table 8.1. Reliable File Transfer (RFT) Errors**

Error Code	Definition	Possible Solutions
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 will fail	This occurs when you start the container if RFT is not configured properly to talk to a PostgreSQL database.	The usual cause is that Postmaster is not accepting TCP connections, which means that you must restart Postmaster with the <code>-i</code> option (see <a href="#">Configuring RFT</a> ).

## 2. RFT fault-tolerance and recovery

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

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

### 2.1. Failure modes that are not addressed:

- Running out of disk space for the database.

# Chapter 9. Usage statistics collection by the Globus Alliance

## 1. Usage statistics sent by RFT

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

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

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the RFT component. Nevertheless, if you wish to disable this feature, please see the "Usage Statistics Configuration" section of [Configuring Java WS Core](#) for instructions.

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

---

<sup>1</sup> [../../Usage\\_Stats.html](#)

# Index

## C

- configuration interface, 2
  - auto-registration with WS MDS Index, 4
  - overview, 2
  - resource properties, 4
  - settings, 2
- configuring, 2
  - auto-registration to WS MDS Index Service, 4
  - MySQL database, 3
  - PostgreSQL database, 2
  - registering manually to WS MDS Index Service, 5
  - resource properties, 4

## D

- debugging
  - logging, 10
- deploying, 7
  - into Tomcat, 7

## E

- errors, 12

## F

- fault tolerance, 12

## I

- installing, 1
  - latest code from CVS (advanced users only), 1

## L

- logging
  - CEDPS-compliant, 10
  - debugging, 10

## M

- MySQL
  - using, 6

## R

- recovery, 12

## S

- security considerations, 9

## T

- testing, 8
- troubleshooting, 12

installation, 12

## U

- usage statistics, 13