Table of Contents

FTS Server administration version 2.x

This page describes the basic usage and adminstration of the FTS for version >2.0, FtsRelease20, and the format, location and contents of the logfiles. It refers to both the FTS web-service and the FTA agent daemons.

For a more detailed look at:

- Channel administration FtsChannelAdmin20.
- VO administration functions, look at FtsVOAdmin20.

SysV configuration

FTS web-service

The FTS web-service runs inside the Tomcat 5 J2EE container. This is controlled by the `/etc/init.d/tomcat5` SysV script. It starts a single Java process daemon with user `tomcat4:tomcar4`. (sic: Tomcat 5 running under user `tomcat4`.) It is ckconfig'd to start on runlevels 2,3,4,5.

FTA agent daemons

The FTA daemons are C++ daemons. They are all controlled by the `/etc/init.d/transfer-agents` SysV script. By default, this script will apply the given action to all the configured FTA daemons on the server.

Each daemon is prefix with the name `glite-transfer-` and runs with user `edguser:edguser`.

All daemons are ckconfig'd to start on runlevels 2,3,4,5.

FTS Web-service Portal

Starting the FTS web-service

To start the service:

```
service tomcat5 start
```

this will start the Java Tomcat 5 daemon under userid `tomcat4:tomcat4`. The web-service itself takes a few seconds to start up within the servlet container. Check for success of failure in `/var/log/tomcat4/catalina.out`.

Stopping the Tomcat web-service

To stop the service:

```
service tomcat5 stop
```

Reconfiguring the Tomcat web-service

If a configuration change must be made to the webservice, edit the `site-info.def` file, rerun the YAIM configuration script to rebuild the Tomcat config files, and then restart the daemon.

1. `/opt/glite/yaim/scripts/configure_node site-info.def FTS2`
2. `service tomcat5 restart`

Advanced: Tomcat DB connection pool tuning

The Tomcat container maintains a connection pool to the database. The defaults are tuned for reasonable-load production.

By default, Tomcat will maintain a single open connection. It will open more, as necessary, up to a maximum of 50 concurrent DB connections. It will allow no more of than 30 of these to be idle at any one time (closing connections if there are too many idle). The DB transaction time is short compared to the overall web-service operation, so 50 DB connections can serve many concurrent web-service clients without any queuing on the server.

The parameters can be changed in the file:

```
/etc/tomcat5/Catalina/localhost/glite-data-transfer-fts.xml
```

- `maxActive` sets the maximum number of open connections (default is 50). This maximum should be less than the maximum number of sessions allowed on the database account.
- `maxIdle` sets the maximum number of connections to keep open (default is 30)

Any changes require a server restart:

```
service tomcat5 restart
```

Note that these tunings will be lost if you rerun the YAIM configuration script (these advanced tunings will be moved into YAIM in a future version).

Updating CA certificates

FTS caches the CA certificates, and it does it once, at service (re)start. Therefore, if you updated the CA certificates, the service must be restarted!

FTA Agents

Starting the FTA agent daemons

To start all the agents:

```
service transfer-agents start
```

this will start all the configured agent daemons, one-by-one.

To start just an individual instance:

```
service transfer-agents start --instance glite-transfer-channel-agent-urlcopy-CERN-CERN
```

where `glite-transfer-channel-agent-urlcopy-CERN-CERN` is the name of the channel agent to start.

Stopping the agent daemon

To stop all the agents:

```
service transfer-agents stop
```

this will stop all the configured agent instances one-by-one.

To stop just an individual instance:

```
service transfer-agents stop --instance glite-transfer-channel-agent-urlcopy-CERN-CERN
```

where `glite-transfer-channel-agent-urlcopy-CERN-CERN` is the name of the channel agent to stop.

Reconfiguring the FTA agents

If a configuration change must be made to the service, edit the `site-info.def`. file, rerun the YAIM configuration script to rebuild the config files, and restart the agent or agent(s).

```
/opt/glite/yaim/scripts/configure_node site-info.def FTA2
```

The configuration script will report which agent instances have changed configuration. You should restart each of them individually:

```
service transfer-agents restart --instance glite-transfer-channel-agent-urlcopy-CERN-CERN
```

or, if the majority have been changed, you can restart all of them simply with:

```
service transfer-agents restart
```

Logfiles

The FTS service consists of two independent daemon types:

- the FTS web-service portal running inside the Tomcat application server
- the FTA transfer agents (channel agents and VO agents), which run as a set of normal unix daemons.

All the logfiles are independent. Additionally, when the channel agent daemon needs to perform a transfer, it forks, and then starts logging the results of that transfer into a distinct logfile.

FTS webservice logfiles

The webservice logfiles can be found in `/var/log/tomcat4/`.

1. `catalina.out`. This is the least useful logfile and logs only critical container events. Generally no application logging goes here.
2. `glite-security-trustmanager.log`. This contains the authentication result of every call to the web-service. It is best avoided since the information is duplicated in other logs.
3. `org.glite.data`. This is the primary logfile for the FTS webservice and will be very verbose since it runs in `DEBUG` mode. This is the first place to look if the webservice or the client commandline tools start to misbehave. DB errors and service startup errors are usually flagged as `ERROR` or `FATAL` catagory and are always accompanied by a Java stack trace in the log to indicate why and where they occurred. In particular, if a client command line produces the message `"Internal server error"` then this is the logfile to look into.
4. `org.glite.data.transfer.fts-calls`. This logs, one line per call, normal user-level calls to the webservice (those made to the File Transfer port-type). The method and its paramters are logged, together with the hostname and DN of the calling client.
5. `org.glite.data.transfer.channeladmin-calls`. This logs, one line per call, channel management calls to the webservice (those made to the Channel Management port-type). The method and its paramters are logged, together with the hostname and DN of the calling client.

All the FTS logfiles are rotated daily provided the service is running. The rotation is done by the log4j tool writing the logfile - there is no explicit logrotate script. The rotated logs are subsequently gzipped by a `root`

cron job installed in `/etc/cron.d/`, since the log4j software does not support zipping of rotated logs.

FTA agent daemon logfiles

The FTA agent daemons log into `/var/log/glite/` with a logname:

```
glite-transfer-channel-agent-urlcopy-INSTANCENAME.log
glite-transfer-channel-agent-srmcopy-INSTANCENAME.log
glite-transfer-vo-agent-INSTANCENAME.log
```

depending on the agent type, where `INSTANCENAME` is what you specified in the FTS server configuration file. There is one logfile per agent instance.

Using the default configuration, the daemons logs at `INFO` level which means individual actions (starting and stopping a transfer) as well as errors and warning will be displayed. Normal startup configuration parameters will also be logged. If the agent starts to develop problems, your FTS support may ask you to up the logging to `DEBUG`. To do this, edit the `site-info.def` file adding the line below for the instance in question:

```
FTA_CERN_BNL_LOG_PRIORITY=DEBUG
```

where `CERN-BNL` is the instance name in this example. You should then reconfigure the agent as described above.

All the agent logs are rotated by the root logrotate daily cron job from the script `/etc/logrotate.d/glite-data-transfer-agents`. All agents daemons are restrated by the postrotate script.

FTA transfer logfiles

By far the largest amount of logging will be produced from the individual transfer logfiles, since in the recommended configuration, this should be run at `DEBUG` level. These are all put under the directory `/var/tmp/glite-url-copy-edguser/`.

Active logfiles (those still that are being written to by an active transfer process) are directly in:

```
/var/tmp/glite-url-copy-edguser/.
```

Logfiles of completed jobs get put in:

```
/var/tmp/glite-url-copy-edguser/CHANNELNAMEcompleted/
```

while logfiles of failed jobs get put in:

```
/var/tmp/glite-url-copy-edguser/CHANNELNAMEfailed/
```

There is one pair of directories (failed and completed) for every `CHANNELNAME` that has run a job on the server.

The is one logfile for every transfer attempted. Once a file transfer status has been determined (either failed or completed) it's logfile is moved to the relevant directory and no longer written to). The logfile names consist of the channel name plus a datestamp plus a mktemp hash; there too many to give sensible names. For example, a failed transfer log on the `CERN-RAL` channel could look like:

```
/var/tmp/glite-url-copy-edguser/CERN-RALfailed/CERN-RAL__2006-05-03-1023_JHqepc
```

The contents of the logfile for a single 3rd party copy log all the steps that the transfer went through:

- SRM get on the source file
- SRM put on the destination file
- gridFTP transfer of file
- SRM set status done on source file
- SRM set status done on the destination file

If any steps need to be retried, you may see multiple versions of the same step. If possible, errors returned from gridFTP and SRM are logged. There are timestamps on all of the entries.

NOTE! The timestamps are in UTC (GMT), NOT the current timezone!

The final entry should always be the status of the transfer together with any error that may have been returned if the transfer failed. This final status and the error is put in the database. Note that if any step after the SRM get or put fails, the process will always attempt to do and SRM `setStatus(Done)` on the source and destination, to cleanup the state on the SRMs, so you will see this happenning in the logfile. It will also attempt an `advisoryDelete` on the destination file to clean it up.

The FTS transfer logfiles are not currently cleaned up by default, so the contents of these directories will grow.

FTA transfer logfiles: lost transfers

You may notice a new directory in FTS 2.0 per-channel:

```
/var/tmp/glite-url-copy-edguser/CHANNELNAMElost/
```

This is where it puts any transfers that did not complete gracefully. There is a known issue in the Globus VDT 1.2 client we use where upon certain non-protocol compliant errors from one of the gridFTP servers, the Globus code helpfully executes `abort(3)` inside the library. The `SIGABORT` isn't reliably caught, so usually the process ends [dis]-gracefully.

In these cases, the failed logfiles are put in the 'lost' directories. You should check for them periodically.

The abort issue should be solved in the VDT 1.6 client that we are slowly migrating towards - this will mean that these bad gridFTP transfers are cleanly failed by the FTS.

The solution to stopping the bad transfers in the first place is usually to restart the (dcache) gridFTP doors on the involved SRM.

Cleaning up FTS transfer logfiles

For large numbers of transfers at high transfer rates, the directory `/var/tmp/glite-url-copy-edguser/CHANNELNAMEcompleted/` will end up containing a very large number of small logfiles. There is a script tool `/opt/glite/bin/glite-url-copy-cleanlog` to clean up these directories and archive the results in a tarfile.

```
/opt/glite/bin/glite-url-copy-cleanlog -h
```

shows the help. For example, to cleanup completed logfiles created by the agent user `edguser` on channel `CERN-RAL`, cleaning only logfiles older than half a day ago, and archiving the results to `/var/glite/transfer-logfile-backup/`, run as root:

```
/opt/glite/bin/glite-url-copy-cleanlog edguser CERN-RAL
/root/glite/transfer-logfile-backup 0 0.5
```

The tool is intended to be run regularly from cron. It is suggested to archive the completed logfile daily for all channels on the system. Failed logfiles can be archived as necessary - and possibly you may wish to inspect them first.

FTS service admimistration and channel management

A more detailed look at channel administration, along with common administrative tasks is described in FtsChannelAdmin20.

Details of the processes running

Doing a `ps aux` as root will show the running processes.

If the webserver is configured and running on the node you should see:

- a Java process owned by `tomcat4:tomcat4`. This is the main Tomcat process that runs the FTS webservice application. Running `ps auxm` will show a number of pooled service threads.

If any agents are configured and running on the node you should see:

- For every configured instance, an agent process called `glite-transfer-channel-agent-urlcopy-CHANNELNAME` or `glite-transfer-channel-agent-srmcopy-CHANNELNAME` (for channel agents) or `glite-transfer-vo-agent-VONAME` owned by `edguser:edguser`. These are the main agent processes. Running `ps auxm` should show two threads per instance.

When a channel agent wants to start a new transfer job, it creates a memory mapped file describing the job. It then double forks and calls `execlp` on `glite-url-copy-exec` passing the memory mapped file as an argument (the result of which will be called the *transfer process*). Note this is different from FTS 1.5 (which just double forked away from the parent).

The majority of the transfer process' memory usage exists in shared memory since it consists of Oracle shared libraries; this should be remembered when looking at memory usage with tools such as `top`. Running `ps auxm` should show three threads once the transfer process has established itself. After forking, the process is name like:

```
glite-url-copy-exec CERN-SARA__2007-07-19-1026_oaqzmc
```

which matches the name of the active logfile in `/var/tmp/glite-url-copy-edguser/`. In that directory, there is also the memory-mapped file, named like `/var/tmp/glite-url-copy-username/CERN-GRIDKA__2006-05-03-1059_9o9ie2.mem`. This is used updated by the transfer process and read by the original transfer agent daemon to check the current status of the job.

For debugging purposes, the current status of a running transfer process may be retrieved by running the command line passing the relevant memory mapped file (as the daemon user):

```
glite-url-copy-print-status GRIDKA-CERN__2007-07-19-1018_v87dX0.mem
```

or simply `cat` the active logfile
`/var/tmp/glite-url-copy-edguser/GRIDKA-CERN__2007-07-19-1018_v87dX0.log`

Advanced: debugging the agent daemon or transfer proccesses

The transfer daemon does a `chdir` to `/tmp/` upon forking. Provided the core file size has not be limited by the daemon user or by the system-wide limit configuration (usually set in `/etc/security/limits.conf`) then any core files from SEGV failures or similar should be in `/tmp/`.

To debug the process with `gdb` or similar (either process-attached or offline core file), the originating executable is:

```
/opt/glite/libexec/glite-url-copy-exec
```

the debugger should follow through the shared libraries to the current execution point. The libraries are built with debugging symbols.

Advanced: debugging the web-service Java process

To do live debugging on the Java Tomcat web-service process, edit the file:

```
/etc/init.d/tomcat5
```

and change the `daemon` and `su` call in the start method from:

```
daemon --user $TOMCAT_USER $TOMCAT_SCRIPT start
```

to:

```
daemon --user $TOMCAT_USER $TOMCAT_SCRIPT jpda start
```

and restart tomcat:

```
service tomcat5 restart
```

This will open a JPDA debugging port on tcp/5000 that you can connect to with a Java debugger. FTS/FTA FAQ

Troubleshooting

Troubleshooting information and FAQ can be found at DMFtsSupport

Maintainer: GavinMcCance

This topic: LCG > FtsServerAdmin20
Topic revision: r5 - 2010-03-17 - unknown