Table of Contents

FTS Administration Toolpack for FTS 2.1

The FTS toolpack consists of a variety of scripts and database procedures to do useful things upon the FTS - either the daemons or the backend database.

This is where quick scripts will go until we've had a chance to integrated them nicely into the release. Much of it is not very pretty and much of it will do terrible damage to your database schema if you run it wrongly (requiring a complete re-install). You may wish to ask your DBA to run some of these tools for you.

Some of the tools display messages on the standard output. In order to get them please make sure your dbms_output package is enable.

```
execute dbms_output.enable;
```

A proper release?

The toolpack is built in gLite, however not integrated into the release.

It comes in the messy `glite-data-transfer-scripts` RPM.

We have merged the **history** package into FTS 2.2 (to be released), where it will be part of the core schema, so no additional packages will be necessary.

Current version

Current version is 2.1.0-1 (to match FTS 2.1 major version numbers).

Version	RPM	Notes
2.1.0-1	glite-data-transfer-scripts-2.1.0-1.noarch.rpm	Fixes for 3.2.0 schema

FTS admin package

This is a PL/SQL package that can be loaded into your database. It consists of two files:

```
/opt/glite/share/glite-data-transfer-scripts/plsql/fts_admin_pack.sql
/opt/glite/share/glite-data-transfer-scripts/plsql/fts_admin_pack_body.sql
```

Load the package into your database on the schema owner account (the account on which your ran the schema install) using the tool of your choice.

Changing the channel source or destination

To update the channel destination (for example if the GOCDB sitename has changed or if you want to make it upper-case, as recommended), run the `fts_admin_pack.CHANGE_CHANNEL_SOURCE` or `fts_admin_pack.CHANGE_CHANNEL_DEST` methods, passing the channel name and the new source or destination.

You do not need to stop the agents to make this change.

```
SQL> exec fts_admin_pack.CHANGE_CHANNEL_DEST('CERN-RANDOM', 'RANDOMT1_NEW_NAME');

PL/SQL procedure successfully completed.
```

Changing a channel name

The channel name may be changed using the `fts_admin_pack.RENAME_CHANNEL` passing the old and new channel names.

This is not a lightweight operation with the current schema.

YOU MUST STOP ALL VO AND TRANSFER AGENTS BEFORE YOU DO THIS

If you do not stop all agents, it will lock-up your schema (in which case, buy your DBA a coffee...)

```
SQL> exec fts_admin_pack.RENAME_CHANNEL('CERN-RANDOM', 'CERN-NEWRANDOM');

PL/SQL procedure successfully completed.
```

The channel name is used as a key in the job table and updating this may take a long time (like `several hours` if you have many jobs in the table).

FTS history package

* Check carefully the history package with schema 3.2.0! *

The purpose of the history package is to move old jobs (jobs in terminal state, X days old) to a history table.

This will prevent any part of the FTS from indexing over those jobs while maintaining the data for audit requirements. This is important at high transfer rates since making the database trawl through hundreds of thousands of jobs just to find the few ones which are currently running leads to peformance problems. N.B. AFTER A JOB HAS BEEN MOVED, YOU CANNOT QUERY ITS STATUS BY ANYTHING EXCEPT DIRECT SQL COMMANDS (THE FTS WEB-SERVICE WILL NOT SEE IT).

N.B. This is (still) a work-around. A better solution is being investigated making use of Oracle partioning. The work is in progress for this - in particular, the 3.1.0 schema now has a partitioning key (timestamp) which is put onto each entry by a trigger when the associated job enters a terminal state.

Loading the schema

Load the history schema into the database. The schema is essentially the same as the `t_job`, `t_file` and `t_transfer` tables. This should be loaded onto the schema owner account.

From SQL/Plus:

```
SQL> @/opt/glite/share/glite-data-transfer-scripts/plsql/create_fts_history_tables.sql
```

Loading the package

This describes how to load PL/SQL packages into the database using the SQL/Plus tool.

You should ask your DBA if you have any problems with this; they may have an alternative mechanism for using user packages. You may need some permissions to be set on your FTS account.

A package generally has two components, the header and the body. The header should be loaded first. Load directly using SQL/Plus:

```
SQL> @/opt/glite/share/glite-data-transfer-scripts/plsql/fts_history_pack.sql

SQL> @/opt/glite/share/glite-data-transfer-scripts/plsql/fts_history_pack_body.sql
```

The package may be queried:

```
SQL> describe fts_admin_pack
PROCEDURE CHANGE_CHANNEL_DEST
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 V_CHANNEL_NAME                 VARCHAR2(32)            IN
 V_CHANNEL_DEST                 VARCHAR2(100)           IN
PROCEDURE CHANGE_CHANNEL_SOURCE
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 V_CHANNEL_NAME                 VARCHAR2(32)            IN
 V_CHANNEL_SOURCE               VARCHAR2(100)           IN
PROCEDURE RENAME_CHANNEL
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 V_CHANNEL_NAME_OLD             VARCHAR2(32)            IN
 V_CHANNEL_NAME_NEW             VARCHAR2(32)            IN
```

and procedure methods may be called:

```
SQL> exec fts_admin_pack.CHANGE_CHANNEL_DEST('RANDOM-RANDOM', 'T1SITE_NEWNAME');

PL/SQL procedure successfully completed.
```

Start the DBMS job

You need to have DB permissions on the FTS account to run DBMS jobs (essentially a database cron task).

It will move jobs in a terminal state that have a submit_time over 7 days old. The job runs, by default every 10 minutes, moving 100 jobs and their contents at a time.

REMEMBER - AFTER A JOB HAS BEEN MOVED, YOU CANNOT QUERY ITS STATUS BY ANYTHING EXCEPT DIRECT SQL COMMANDS (THE FTS WEB-SERVICE WILL NOT SEE IT).

```
SQL> exec fts_history.submit_job;

PL/SQL procedure successfully completed.
```

Check the job is there using your user ID:

```
SQL> select job, next_date, next_sec from user_jobs where user='LCG_FTS_STRESSTEST2';

       JOB NEXT_DATE NEXT_SEC
---------- --------- --------
        82 04-JUL-06 16:00:24
```

Start the job for the first time (this is often needed) and check that there is no error code reported in the logging table. Use the job id you got back from hte user_jobs table:

```
SQL> exec dbms_job.run(82);

PL/SQL procedure successfully completed.

SQL> select ERRCODE, JOBS, FILES from T_HISTORY_LOG;

   ERRCODE       JOBS      FILES
```

```
---------- ---------- ----------
                  100        217
```

The numbers refer to the number of jobs and files that were moved to the history table on this job run. There will be an entry each time the job runs. Any error codes are reported here.

To stop the DBMS job, run:

```
SQL> exec fts_history.STOP_JOB;

PL/SQL procedure successfully completed.

SQL> select * from user_jobs where job = 82;

no rows selected
```

Check the status

Every time the job runs, the number of entries in `t_job` should reduce by approx. 100, provided you have that many entries that are more than 7 days old in a final job state. The number of entries in `t_job_history` should increase by the corresponding number.

The same can be said for `t_file` / `t_file_history` and `t_transfer` / `t_transfer_history`, although the number of entries moved will depend on the content and performance of the jobs.

```
SQL> select count(*) from t_job;

  COUNT(*)
----------
     12223


SQL> select count(*) from t_job_history;

  COUNT(*)
----------
       100
```

... wait a while (> 10 minutes) ...

```
SQL> select count(*) from t_job;

  COUNT(*)
----------
     12123


SQL> select count(*) from t_job_history;

  COUNT(*)
----------
       200
```

Oracle block fragmentation

Fragmentation has been noted on the DB blocks when using the history package.

Impact

- Standard service operation is not affected (since FTS uses indices to find the blocks).
- The table takes up more space than it should (since the fragmented blocks have a large unused portion).
- Schema upgrades (in particular new index builds) take much longer than they should, since they require a full table scan which requires reading all the blocks into the DB buffer cache.

Resolution

- Before schema upgrade, the tables can be defragmented. The actual cause of the fragmentation is being understood with Oracle support.

Notice

Please do this in collaboration with your DBA. It is likely that most of these operations will require DBA priviliges.

Please do this in collaboration with the WLCG 3D project. Any question should be sumitted to `grid-service-databases@cern.ch`.

How to check for fragmentation

Check if any table is highly fragmented (>60%)

```
select a.owner, table_name, mb_used, mb_allocated,
round(100-((mb_used*100)/mb_allocated)) PCT_FRAGMENTED from
  (select round((num_rows * avg_row_len)/1024/1024) MB_used, table_name, owner from dba_tables) a
  (select round(sum(bytes)/1024/1024) MB_allocated, segment_name,
     owner from dba_segments group by segment_name, owner) b
   where a.table_name=b.segment_name
      and a.owner=b.owner and mb_allocated>100 and a.owner not in 'SYS'
   order by 5 desc, 4 desc;
```

the likely candidates for high levels of fragmentation are `t_file`, `t_job` and `t_transfer`.

How to defragment

Check with `grid-service-databases@cern.ch` first!

Notes:

1. Stop all daemons from accessing the database
2. The SHRINK procedure requires ASSM tablespaces
3. You will need to drop the timestamp function-based indices and recreate them afterwards

Basic procedure per table you want to fragment:

1. Activate row movement on this table: `alter table OWNER.TABLE_NAME enable row movement;`
2. Shrink it: `alter table OWNER.TABLE_NAME shrink space;`
3. Re-gather stats: `exec dbms_stats.gather_table_stats('OWNER','TABLE_NAME');`

See FtsRelease20TableFragmentationCern for CERN procedure.

Show writer account package Hide writer account package

Writer account preparation

This step is for the CERN Oracle RAC databases and is only relevant for CERN's writer account setup.

See your DBA about your site's DB account setup.

THIS NEEDS TO BE DONE ON EVERY SCHEMA CHANGE

Load the package into the schema owner account

Load the writer account package from sqlplus into the schema OWNER.

```
sqlplus username/passwd@"connectstring" < /opt/glite/share/glite-data-transfer-scripts/plsql/fts_

sqlplus username/passwd@"connectstring" </opt/glite/share/glite-data-transfer-scripts/plsql/fts_w
```

Make the grants in the owner account

Run this from sqlplus on the schema OWNER account. It will make the necessary grants to the specified writer account.

```
exec fts_writer_account.make_grants('lcg_fts_prod_w');
```

Make the synonyms in the writer account

Run this from sqlplus on the schema WRITER account. It will make the necessary synonyms to the schema objects in the specified owner account. Note you must prefix the package call with the name of the owner account (since this is who the package whose function your are calling belongs to).

```
exec lcg_fts_prod.fts_writer_account.make_synonyms('lcg_fts_prod');
```

Last edit: UnknownUser on 2010-02-25 - 17:45
Number of topics: 1

This topic: LCG > FtsAdminTools21
Topic revision: r5 - 2010-02-25 - unknown