Protocol Archive Test Plan for the Swift Mission

HEASARC

Laboratory for High Energy Astrophysics, NASA/GSFC, Code 662, Greenbelt, MD 20771

24 October 2002

1 Introduction

This document describes the planned tests to exercise the end to end archive protocol. This includes : transferring data, moving data to the final archive location, initiating the database ingest. A different test plan was developed to specifically test the DTS transfer protocol.

While the software to transfer data, DTS, is common to all sites, the software that archives the data and ingests databases may use procedures already in place at the different sites. This depends on how the hardware is locally configured with respect to the staging area and the type of database system in use.

The HEASARC has developed a package named Data Archive System (DAS) to move data from the staging area to the final archive location. It is configurable in terms of disk name and can be re-used by different sites providing that the disk space for the final archive location is accessible from the DTS machine by a single mount point. Similar to DTS, it has the capability to spawn scripts to initiate other processing. Other sites can either install the DAS as it is or re-use parts.

In distribution there is also a package that contains scripts to ingest databases into the EXOSAT DBMS browse systems that may be re-usable with other sites.

All sites participating to this test should have already installed DTS and have procedures in place to archive the data and to ingest databases. They can join the test even with a partial procedure in place and the same test will be repeated.

The software DTS, DAS and database scripts are available at the following web address:

http://legacy.gsfc.nasa.gov/dts/dts.html

If questions arise concerning the usage of the listed software or a bug is discovered in the course of testing, please contact *micah@milkyway.gsfc.nasa.gov* . Provide a detailed description and associated log files if applicable.

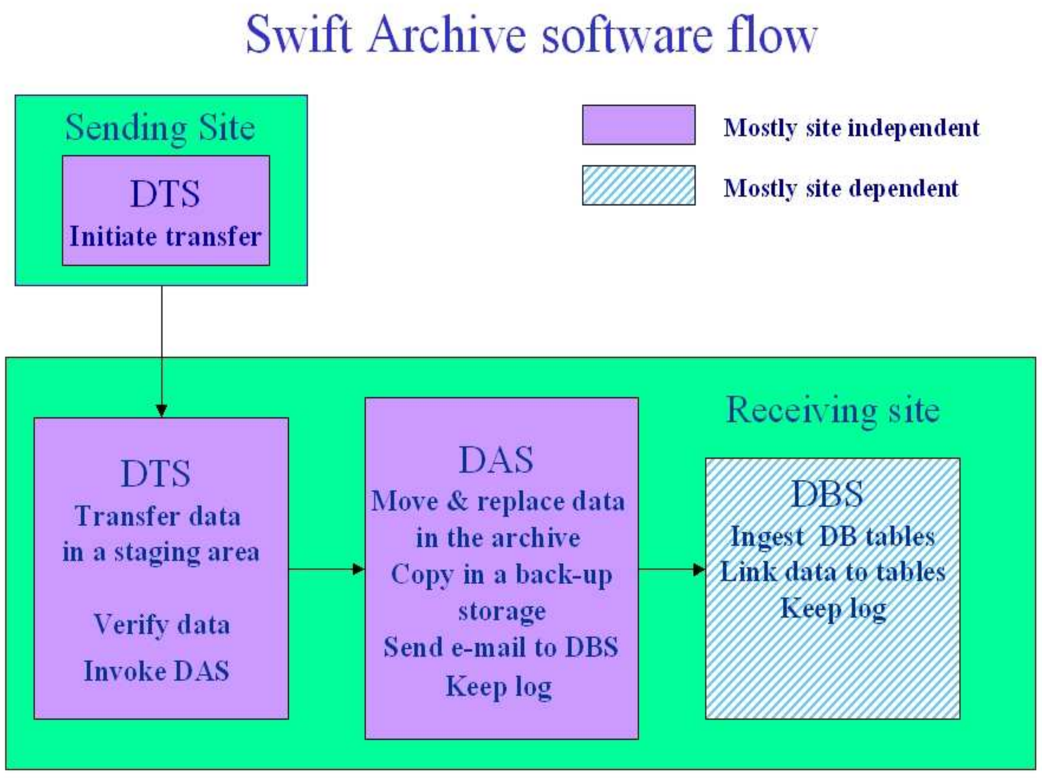
2 Overall Archive Procedure

Below is the description of the overall archive procedure as currently implemented at HEASARC. This consists of three steps.

- The first step is the data transfer. This is achieved via DTS that transfers data from the sending site to the receiving site. The data arrives in a staging area of the receiving site. The DTS software is common to all sites.

- The second step is to move the data from the staging area into the final archive location. This is achieved via DAS, which is initiated within DTS via the DTS flag 'type'. Depending on the 'type' value, DAS moves data into the appropriate directory for that 'type'. If the incoming data set is a newer revision of the same data set already present in the archive, it replaces the latter with the newer version. The older version is moved into a timestamped directory in the backup location. DAS keeps logs of all these activities. DAS, like DTS, uses the 'type' to initiate other operations. The disks for the final archive location of the data needs to be accessible from the DTS machine by a single mount point.

- The third step is to ingest the database tables in the database system and to connect the data to the entries in the database tables. This is done via the database system ingest (DBSI). This is initiated by a DAS-spawned script that copies the database table into a working directory and sends an email to an address monitored by procmail. The working directory is assumed to be on disks mounted on the DTS machine. The email initiates the database ingest procedure. This creates tables suitable for the HEASARC database systems Sybase and EXOSAT DBMS (Browse), ingests the tables, saves the incoming tables to a backup directory and keeps logs of the overall operation. A second email initiated by a different DAS-spawned script to an address monitored by procmail starts the indexing of the data file in the archive that are then associated with the entries in the database table. The latter procedure is necessary to allow the data retrieval via the web browser. Most of these procedures depend on the database system and how each site has organized the ingest procedure locally.



3 Test Summary & What to download

3.1 Test Summary

There are several types of tests to verify the Swift archive procedure. These are :

- Ingest of different type of data sets and database tables. This exercises the usage of the 'type' flag in DTS and in the subsequent archive procedures.
- Ingest of different sizes of datasets to exercise the time necessary to complete the archive of different size data sets. In some tests, data are sent almost simultaneously, while other tests data are sent in intervals separated by several hours.
- Ingest of a data set processed with a newer software version that replaces an existing data set. This exercises the replacement procedure as well as the ability to correctly keep records of incoming and backed up data.
- Ingest a PGP encrypted data set to ensure that encryption does not cause unforeseen problems with the archive procedure.
- Simulate failures in the archive procedure or database ingest due to hardware momentaneously unavailable.

3.2 What to download

To exercise the entire archive procedure, the receiving site should have in place mechanisms to transfer the data, move data in the final archive location, to ingest the database tables and to link the data to tables. To transfer the data all sites should have already installed DTS. If the receiving site adopts DAS for their archival operation, this software should be downloaded and set-up as described in the following sections. If the receiving site uses the EXOSAT DBMS as database system, there is available software to ingest tables. The table ingest software installation and set-up it is described in a separate document. The software (DTS, DAS & the database ingest) is available at

http : //legacy.gsfc.nasa.gov/dts/dts.html

There is no software available to link the data to the database tables, since it is specific to how the archive data center provides data to the users via their local interface.

The sending site on the other hand does not need any additional software besides DTS.

All sites (receiving & sending) need to download the test bed.

4 Test Bed

To carry out these tests the following data sets are available:

- Databases
- Production data (varying size and revision)
- TDRSS messages

The simulated database tables have as content the fields described in the HEASARC-SDC ICD. Date, positions and some other quantities are arbitrary, however the numbering of the observations follows the observation number scheme adopted for Swift.

The simulated Swift observing log, known as as-flown timeline and described in the MOC-SDC ICD, is a test file generated by Omitrom. Each delivery of this table is just an update, always unique, to previous deliveries. To simulate the append and replace procedures for the database ingest two additional files are available.

The datasets (production data and TDRSS messages) are make-overs of existing files taken from the HEASARC archive where the root of the filename has been changed accordingly to the root name in use for Swift. The content of each data set is arbitrary and it is made up of files to reach a specific size.

The data sets and database tables are available together with this test plan at :
http://legacy.gsfc.nasa.gov/dts/dts.html

Retrieve the archive test bed from the website:

http://legacy.gsfc.nasa.gov/dts/protect/archtest.tar.gz

Make a test bed directory and unpack the files into that directory:

```
gunzip -c archtest.tar.gz | tar xvf -
```

The top level of the test bed directory should contain the following:

```
config/
db/
obs/
tdrss/
trend/
```

The test bed contains data, database tables, scripts and associated file lists for use in the testing procedure.

5 Software Setup and Installation

5.1 DAS Setup & Installation

Download the DAS software, and untar it in the home directory of the DTS operator account (e.g. dtsops). The das.config file must then be edited to reflect the specifics of the local site. In particular, LOGDIR, DESTDIR, BAKDIR, and INVBAKDIR must be modified to reflect valid pathnames on the host machine, and it is recommended that MAILERR be set to the DTS operator account so all error messages go to the same account. The remaining variables may be left as their default values.

For example, if the path to the main disk is */local/swift*, the following structure is suggested.

```
DESTDIR="/local/swift/data"
BAKDIR="/local/swift/bak"
INVBAKDIR="/local/swift/bak/inv"
```

Note, these directories must be created before executing the DAS.

The recommended location for LOGDIR depends on the setting for DTS_IN and DTS_OUT in dts.config. Preferably these log directories should be contained within a common directory. For example,

dts.config:

```
DTS_IN="/local/swift/log/in"
DTS_OUT="/local/swift/log/out"
```

das.config:

```
LOGDIR="/local/swift/log/das"
```

Replace the dts.scripts file in your DTS installation with the one provided in the 'config' directory of the test bed.

The general process is to transfer data with DTS, and based on the 'type' trigger the script set in the dts.scripts file. In this case, the triggered script will always be das.pl with suitable flags. After DAS archives the file to the appropriate directory, the types in the das.scripts file trigger other scripts which are included in the scripts directory found in the DAS distribution or your own scripts patterned after the included example.pl script. Which scripts you use are based primarily on the requirements of your local database.

In the 'config' directory of the test bed, there are two type/script definition files for DAS. The das.scripts file contains the standard setup for HEASARC where dbnotify.pl copies the database file to an incoming directory based on the database name and emails an account which is monitored with procmail, triggering the ingest. For types that don't involve the database, startdpl.pl is used, which emails the location of the recently archived data for action by another monitoring process. If your site has a similar setup, modify dbnotify.pl and startdpl.pl to reflect the appropriate email and directory structure in the section at the top of the scripts labelled "Local configuration." Then, replace the installed das.scripts file in the das directory with the das.scripts.db file from the test bed.

If, however, your database ingest has different requirements, for the purpose of this test, use the das.scripts.mailonly file to replace the standard das.scripts file. For all types, a script called simplemail.pl mails the arguments passed on by das.pl. Edit the email in simplemail.pl to reflect the email of the tester.

If the DAS has been installed in a place other than the recommended location (i.e. the DTS operator home directory), the das.scripts file must be modified to reflect the correct path.

DAS Setup Checklist:

- Untar DAS package into ~`/das`
- Replace existing `dts.scripts` with version from test bed
- Edit ~`/das/das.config`
- If using the HEASARC db mechanism:
  - Replace ~`/das/das.scripts` with `das.scripts.db`
  - Edit ~`/das/scripts/dbnotify.pl`
  - Edit ~`/das/scripts/startdpl.pl`

- Otherwise if using the mailonly method:
  - Replace `~/das/das.scripts` with `das.scripts.mailonly`
  - Edit `~/das/scripts/simplemail.pl`

6 Testing

Each test will use the following general steps:

```
FROM SITE1 send to SITE2: dts -se SITE2 -t TYPE [-l files|-f filelist]
SITE2 RECEIVING MODE    : dts -g
```

SITE1 may run "dts -g" at any time to clean the staging area of successfully transferred files.

The intent is to test the automated sequence of transfer, archive, and ingest from end to end. Each site must have a contact person so that tests may be coordinated between sites. After each test is sent, an email must be sent to the receiving site's contact person so "dts -g" may be run. The email message must contain the identifier for the test that is being run, listed as "Test: TEST_NAME" in the procedures below. Once the files have been retrieved, archived and ingested, the test participant at the receiving site must check the MAILERR account for error messages and verify that the data was properly archived or databases properly ingested.

After all testing is completed, the staging areas may be cleaned out to save disk space, however, we request that you retain the contents of the log directories, as they contain valuable information on transfer rate, etc.

Note that the test procedures are written primarily from the perspective of the sending site. Instructions for to the receiving site are preceded by "Receiving site:".

6.1 Different 'type'

This suite of send operations tests the ability to trigger based on different types.

Change (cd) to the 'db' directory in the test bed.

Test: DBASE_ALL

The following DTS commands must be run before contacting the receiving site. The intent is to test the receiving site's ability to receive multiple unique database tables at once and ingest them despite the concurrency of their arrival.

```
dts -se SITE2 -t Proddatadb -l tb_swiftobs.tdat
dts -se SITE2 -t Tdrssdb -l tb_swifttdrss.tdat
dts -se SITE2 -t Xrtconfdb -l tb_swiftxrlog.tdat
dts -se SITE2 -t Uvotconfdb -l tb_swiftuvlog.tdat
dts -se SITE2 -t Batconfdb -l tb_swiftbalog.tdat
dts -se SITE2 -t Timelinedb -l tb1_swiftlog.tdat
```

A simple shell script, dball.sh has been provided in the 'db' directory to simplify the execution of this test. Usage: `./dball.sh SITE2`

Receiving site: The first task is to ensure that the archive directories and the database are completely empty. It may also be helpful to clean DTS's in and out directories as well the DAS log directory. After running "dts -g" verify that all databases have been ingested properly. If the simplemail.pl method is being used, verify that there are 6 new mail messages corresponding to each database file. An example of the kind of message to expect follows:

```
To: tester@lab.somewhere
Subject: Data Archived

TYPE: Tdrssdb
CMMT: swifttdrss
DEST: /local/swift/data/dbtmp
FILES:
./tb_swifttdrss.tdat
```

Test: DBASE_ADD

The following test is only meaningful if a database ingest mechanism is actually present at the receiving site, rather than the mail-only method. Most databases associated with this project are ingested as a replacement for the existing database, however, the timeline database appends each ingest to the existing database. In order to test this append process, run:

```
dts -se SITE2 -t Timelinedb -l tb2_swiftlog.tdat
```

Receiving site: Run "dts -g" and verify that the data has been appended to the previously ingested table from tb1_swiftlog.tdat. The database should contain 16 rows total, 8 from DBASE_ALL and 8 from the current test.

Test: DBASE_RPL

Once again, the following test is only meaningful if a database ingest mechanism is actually present at the receiving site. It exercises the case where the incoming table is intended to completely replace the current database table.

```
dts -se SITE2 -t Batconfdb -l tb2_swiftbalog.tdat
```

Receiving site: Run "dts -g" and verify that the data has been replaced in the swiftbalog table. The database should contain 7 rows total. The original tb_swiftbalog.tdat file from DBASE_ALL contains 10 rows. The current table, tb2_swiftbalog.tdat is smaller by 3 rows, all the rows with a start time on 2002-04-06 are removed. An inspection of the database should verify 7 rows with none having a start time on that day.

Test: TDRSS_MSG

This test concerns only data, TDRSS messages. Change to the 'tdrss' directory in the test bed.

```
dts -se SITE2 -t Tdrss -f list.txt
```

Receiving site: Run "dts -g" and verify that the data has been copied to the archive in the tdrss/00100001 directory. View the dasinv_Tdrss.log file in the DAS log directory, and verify that an "IN" entry for 00100001 has been written. In general, a line in the inventory log contains entries for IN or OUT, a timestamp, an identifier, the revision, and the full path to the new data.

An email should also be sent from startdpl.pl or simplemail.pl depending on the das.scripts file used.

6.2 Different sizes

This test uses data sets of varying size to determine the scalability of the system. The observation number 00100001001 is 1 GB in size, while 00100001002 is 0.5 GB and 00100001003 is 0.1 GB.

Change to the 'obs' directory in the test bed.

Test: PROD_001

```
dts -se SITE2 -t Proddata -f list001.txt
```

Receiving site: Run "dts -g" and verify that the data has been copied to the obs/00100001001 directory. Also, check the dasinv_Proddata.log file for a new IN entry and that a corresponding email has been delivered.

Test: PROD_002

```
dts -se SITE2 -t Proddata -f list002.txt
```

Receiving site: Run "dts -g" and verify that the data has been copied to the obs/00100001002 directory. Also, check the dasinv_Proddata.log file for a new IN entry and that a corresponding email has been delivered.

Test: PROD_003

In order to prepare the version for the test after this one, run the following shell script from the 'obs' directory:

```
./oldrev.sh
```

Then, perform the dts send:

```
dts -se SITE2 -t Proddata -f list003.txt
```

Receiving site: Run "dts -g" and verify that the data has been copied to the obs/00100001003 directory. Also, check the dasinv_Proddata.log file for a new IN entry and that a corresponding email has been delivered.

6.3 Reprocessed data set

The following tests exercise the revision checking mechanism of the DAS. The Proddata and Tdrss types have a revision file pattern associated with them ('sw[0-9]+_tapeċat'). A file name matching that pattern will be checked for the revision keys, PROCVER and SEQPNUM. PROCVER is of the form X.Y.Z and SEQPNUM is an integer padded with zeroes to make three digits. If the revision of the archive data is earlier than the revision of the incoming data, the existing data set is moved into a backup directory, and the new one is moved into the proper location. If, however, the archive data is newer or equivalent to the incoming data, an error occurs.

Test: PROD_UPD

This test exercises the usual case where a new data set replaces the old. In its original state, the test bed uses the revision: PROCVER=1.8.0, SEQPNUM=005. There is another version of the tape.cat file which has the revision: PROCVER=2.1.5, SEQPNUM=001.

Change to the test bed's 'obs' directory. Run the shell script:

```
./newrev.sh
```

Alternatively, NEWsw00100001003_tape.cat.gz may be manually copied over the existing sw00100001003_tape.cat.gz file in the 00100001003/dir1 directory.

```
dts -se SITE2 -t Proddata -f list003.txt
```

Receiving site: Run "dts -g" and verify that the new data has been copied to the obs/00100001003 directory and the old data is backed up in a timestamped directory in the BAKDIR. Also, check the dasinv_Proddata.log file for a new IN entry for the new data and an OUT entry for old data. Also, verify that the corresponding email is delivered.

Test: PROD_OLD

This test exercises the case when an incoming data set has instead an older revision of what currently present in the archive. This event should not normally occur. If the test is successful the archive should not change.

Run the shell script:

```
./oldrev.sh
```

Alternatively, OLDsw00100001003_tape.cat.gz may be manually copied over the existing sw00100001003_tape.cat.gz file in the 00100001003/dir1 directory.

From the test bed's 'obs' directory, run the command:

```
dts -se SITE2 -t Proddata -f list003.txt
```

Receiving site: Run "dts -g" and verify the error message email. No change will be made to the archive.

6.4 PGP data set

Test: PROD_PGP

This test exercises the requirement to keep data pgp-encrypted for a period early in the SWIFT mission. The test is to verify that the encryption does not break any of the steps of the archival software.

Change to the 'obs/pgp' directory in the test bed and run the following command:

```
dts -se SITE2 -t Proddata -f list003.txt
```

Receiving site: Run "dts -g" and verify that the new data has been copied to the obs/00100001003 directory and the old data is backed up in a timestamped directory in the BAKDIR. Also, check the dasinv_Proddata.log file for a new IN entry for the new data and an OUT entry for old data. Note: The new pgp data has the revision: PROCVER=3.0.0, SEQPNUM=001 A corresponding email should also be received.

6.5 Simulate failure

Test: DAS_FAIL

If the hardware to archive the data is temporarily not available, this causes a failure in the archive procedure. To simulate this occurence the directory name location is arbitrarly changed to a non-existent location.

Run the shell script:

```
./badrev.sh
```

Alternatively, BADsw00100001003_tape.cat.gz may be manually copied over the existing sw00100001003_tape.cat.gz file in the 00100001003/dir1 directory.

From the test bed's 'obs' directory, run the command:

```
dts -se SITE2 -t Proddata -f list003.txt
```

Receiving site: DON'T run "dts -g" yet. First DESTDIR in das.config must be set to a bad directory. Since the DTS operator should not have permission to create a directory in the root directory, a suitable setting would be.

```
#DESTDIR="/local/swift/data"
DESTDIR="/baddir"
```

Note that the existing value is commented out, so one may revert back to the correct setting easily. Now, run "dts -g" and verify the error message email. It should give an error of the form: "ERROR: CHECK_DEST – Failed to create file: /baddir/.nfstest_[pid]". No change will be made to the archive.

To recover from the failure, restore DESTDIR to its correct setting. The log file mentioned in the email has a corresponding .in file in the same directory, which contains the das.pl command that must be re-run by using the syntax `source file.in`. The archive operation should now complete successfully.

Test: DBSTG_FAIL

This test applies only if the local site is using the dbnotify method. The dbnotify.pl script copies the database table to a directory, and then sends an email to an address monitored by procmail. When the mail is received a script is executed which ingests the table. In the standard case, the machine running the DAS is different from the machine running the ingest script. Therefore, a common directory must be mounted on both hosts. This procedure tests the case where the disk containing that common directory is unavailable to the machine running DAS.

From the 'db' directory in the test bed, run the command:

```
dts -se SITE2 -t Proddatadb -l tb_nohea_swiftobs.dat
```

Receiving site: DON'T run "dts -g" yet. First the $dbhold variable in dbnotify.pl must be set to a bad directory. Since the DTS operator should not have permission to create a directory in the root directory, a suitable setting would be.

```
#my($dbhold) = "/dba_dbase/work";
my($dbhold) = "/baddir";
```

Note that the existing value is commented out, so one may revert back to the correct setting easily. Now, run "dts -g" and verify the error message email. It should give an error indicating that dbnotify.pl failed to run. No change will be made to the database.

To recover from the failure, restore $dbhold to its correct setting. The log file mentioned in the email has a corresponding .in file in the same directory, which contains the das.pl command that must be re-run by using the syntax `source file.in`. The database ingest should now complete successfully.

A HEASARC DB Setup

All sites receiving databases for ingesting will have their own site-specific considerations based on the database system used. Therefore, the tests described in this document use general terms when discussing database ingesting. The HEASARC database ingest procedure is described below.

The HEASARC setup automates the database ingest of all planned Swift database tables. DTS transfers the database tables from the sending site, then hands off control to DAS with executes the dbnotify.pl script. This script places the newly delivered TDAT file into /dba_dbase/work/$TABLE/incoming/, then sends an email to an account which has the following procmail recipe associated with it:

```
MAILDIR =       $HOME/Mail      # Make sure directory exists
DEFAULT =       $MAILDIR/mbox
LOGFILE =       $MAILDIR/procmail.log
LOGABSTRACT =   all
VERBOSE =       on
SHELL =         /bin/sh
MY_ADDR =       "from@host.lab"
CRON_NOTIFY =   "dbops@host.lab"

SWIFT_NOTIFY =  "swiftops@host.lab"
TABLELIST = "(swifttdrss|swiftobs|swiftlog|swiftbalog|swiftuvlog|swiftxrlog)"
UCTABLELIST = "(SWIFTTDRSS|SWIFTOBS|SWIFTLOG|SWIFTBALOG|SWIFTUVLOG|SWIFTXRLOG)"

:0
* ^From(:.*| )(dbops|swiftops|dtsops)@.+\.host\.lab\>
* $ ^Subject:[  ]*\/$TABLELIST
* ! ^X-Loop:
* $ MATCH ?? ()\/$TABLELIST
* $ TABLELIST ?? ()\/$\MATCH
```

```
{
        TABLE=$MATCH

        :0
        * $ TABLE ?? ()\/$UCTABLELIST
        * $ UCTABLELIST ?? ()\/$\MATCH
        {
                UCTABLE=$MATCH

                :0 fbi
                | /dba_dbase/work/$TABLE/update-$TABLE.pl

                :0 fhwi
                | formail -I Apparently-To: \
                        -I Resent- \
                        -I"To: $CRON_NOTIFY, $SWIFT_NOTIFY" \
                        -I"From: $UCTABLE Automated Updater <$MY_ADDR>" \
                        -I"Subject: Re: $TABLE" \
                        -I"Reply-To: $CRON_NOTIFY" \
                        -I"X-Loop: $MY_ADDR"

                :0
                ! -t
        }
}
```

The procmail recipe executes /dba_dbase/work/$TABLE/update-$TABLE.pl (included with tdat2browse-and-perl-tools.tar.gz) which then handles the database ingest. If the database ingest cannot be successfully completed within one hour, e-mail is sent notifying the appropriate persons of the error.

With the exception of SWIFTLOG, if multiple TDAT files are present in the incoming directory when update-$TABLE.pl executes, update-$TABLE.pl will process the file with the most recent timestamp and archive any other TDAT files present in the old_files directory. This assumes that TDAT files that are delivered at a later time should supercede any previously delivered TDAT file. It's possible that this may not always be true if there are problems with the data processing or e-mail delivery, but this has been deemed to be an acceptable risk. In such a scenario, the correct TDAT file must be copied from the old_files directory to the incoming directory and the update-$TABLE.pl program must be run manually in order to correct the problem. Since TDAT deliveries are expected to occur only once per day, this is not expected to be a common problem.

SWIFTLOG is different because any newly delivered TDAT files are supposed to be appended to the database table instead of replacing the database table completely. With SWIFTLOG, update-swiftlog.pl processes the TDAT file with the most recent timestamp. Any other TDAT files located in the incoming directory are left there for additional invocations of update-swiftlog.pl. (There should be one e-mail and hence one invocation of update-swiftlog.pl per TDAT file that is copied to the swiftlog/incoming directory.)