Updating the L1 calibration database

Sometimes the instrument specialists give us new calibrations, usually just a dead strip list for TKR or three at a time for CAL (asym, MeV/DAC, peds). These need to be put in a standard place, registered in the calibration database, and set active. This requires picking a time when validity switches from the previous set to the new one. We want this to happen between runs. The gaps between non-SAA runs are short, and CALDB doesn't do leap seconds, so you should put the transitions near the middle of an SAA passage - the first one after the last run that we have data for.

The original update procedure used the rdbGUI and required various information to be input by hand. Since the rdbGUI will not work on >RHEL6 systems, a new Python update script was written in 2022 to automate the procedure.

In general, it is probably better to do the updates early in the week rather than later in case there are any problems.

Database Update Scripts

There are three Python scripts created for the database updates:

- **update_l1_calibration_database.py** is the script that actually does the database update. Technically, it's the only one you really need. You give it the UTC start time for the calibration to take effect and the names of the new calibration file(s). It adds a new row in the database table for each file and changes the end time of the previous calibration file to the start time of the new file.
- **show_l1_selection.py** executes the exact same three database queries the L1 pipeline does when getting the each type of calibration file. The first query gets the "ser_no", which is basically the row number in the table. The second gets the "data_ident" (file name) and a few other parameters for the ser_no, and the third prints out the start and end times ("vstart" and "vend") for using that calibration file. By default, the script will show it for the current time (in UTC) but you can set it to another time. This is useful for checking what the pipeline will choose before and after the start time you give it.
- **view_l1_calibration_database.py** more generally prints the contents of the database table. By default, it will print out the last two rows for each of the four calibration types we update (which should be the previous and current calibrations). The script also has various options to change the rows and columns output.

You can see the range of options provided by each script by just giving it the -h (or --help) option.

The update script performs various checks and will exit with an error if any of these conditions fail:

- The calibration files exist where they are supposed to.
- The time you give it is valid. It should have the correct format (YYYY-MM-DD HH:MM:SS) and cannot be in the past or more than 24 hours in the future. The 24 hours is an arbitrary limit just to make sure you're got giving it a very wrong date by accident.
- The calibration file is not already in the database (based on the file name), so you can't accidentally enter a duplicate.
- For tracker files, the version number in the file name is higher than the current version.
- For non-tracker files, the file name numbers are 6 greater than for the previous calibration file values, e.g., pedavr_658m_662m.xml should be followed by pedavr_664m_668m.xml and the two numbers in the file name are 4 apart. Each file is meant to cover 6 megaseconds and are created from three 2 Ms files, so 240m-244m is made of the 240m, 242m, and 244m files. These two checks are taken from information on the file name on the page Working Group on Calibration Issues. They can be relaxed if they end up being too strict.

The scripts can be accessed in the conda environment called "calibrator" that is available on both the rhel6 and centos7 machines. This environment is also available on the S3DF (with a slightly different activation command), but since the calibration files aren't (currently) up-to-date on S3DF, the update script will give an error. Also, the pipeline wouldn't know to look in the S3DF directory. The two database viewing scripts will work on S3DF, though.

The scripts only recognize the four types of calibration files we currently update: CAL_Asym, CAL_MevPerDac, CAL_Ped, and TKR_DeadChan. Other types can be added easily if needed in the future.

First Time Setup

These are things that you will need to do before you can run the calibration database update script.

1. Create a MySQL options file in your home directory called `~/.my_cal.cnf`.
    a. This file will contain the database access parameters the Python scripts will use.
    b. You can use a different file with the -o parameter. It can be useful to have different files for the calib and calib_test databases. Be sure to include the full path (e.g., ~/) for the file.
    c. Change the permissions so that you are the only one who can read it (`chmod 600`).
2. Add the following lines to the file:

    cnf file

    ```
    [client]
    host=glastCalibDB.slac.stanford.edu
    user=<your database user name>
    password=<your database password>
    database=calib
    ```

    You can use the calib_test database instead of the calib database for testing. You may need to enclose your password in quotes if it contains certain special characters (like #).

If you don't have an individual account, you can now use the 'calibrator' account as was done with the rdbGUI. This was not possible when the scripts were written because the Python module would not work with the old password format used by MySQL 5.5, but since the upgrading the MySQL 8 (and resetting the password), it works. The password is the same as the user name but with an "8" in place of the second "a".

Updating the Database

These are the steps that you will have to do to update the database for a new calibration file(s).

1. If there is not already an existing issue for the update, start one in the JIRA PII tracker. Here is an example that can be used as a template:

   PII-454 - Jira project doesn't exist or you don't have permission to view it. based off the ones they use for on-board hot strip masking updates (e.g., OBCONF-204 - Jira project doesn't exist or you don't have permission to view it.).
2. Figure out the transition time:
   a. Go to the data processing page.
      i. Find the most recent run. It will be at the top, unless deliveries have arrived out of order.
      ii. Click on the run number.
      iii. Note the start time.
   b. Go to the mission planning timeline
      i. Find the first SAA passage that is still in the future ~~processing-wise~~.
         1. NOTE: use an SAA passage that is in the future in UTC, not just processing-wise. The update script will reject an update attempt applied to the UTC past.
      ii. Pick a time that is in the middle of the SAA passage. That is when we will start the new calibration.
      iii. You may also want to note the start time of the next run after the SAA. You will want to send that information to the data monitors list (see below) when you inform them of the new calibration.
3. Setup your environment:
   a. On RHEL6 or CentOS:
      i. Log into a SLAC machine.
      ii. If one is not already sourced in your .bashrc or .cshrc then:
         1. bash: `source /afs/slac.stanford.edu/g/glast/ground/scripts/group.sh`
         2. (t)csh: `source /afs/slac.stanford.edu/g/glast/ground/scripts/group.cshrc`
      iii. This will set environment variables like $LATCalibRoot where the calibration files are stored.
      iv. Make sure that you have write permissions to the `$LATCalibRoot` directory. If not, ask Tom Glanzman.
   b. On S3DF:
      i. Log into S3DF and then the fermi-devl node.
      ii. If you have .profile.d files sourced in your .bash_profile as recommened in the S3DF Cheatsheet, then `$LATCalibRoot` will be set for you.
      iii. Make sure that you have write permissions to the `$LATCalibRoot` directory. If not, ask Tom Glanzman.
4. Copy files to the "normal" place (if the calibration file creators haven't already):
   a. TKR: `$LATCalibRoot/TKR`
   b. CAL: `$LATCalibRoot/CAL/p7repro`
5. Start the **calibrator** conda environment. Note that this uses the Fermitools/Conda Shared Installation at SLAC, which only works with the bash shell.
   a. S3DF: `source /sdf/group/fermi/sw/conda/bin/activate calibrator`
   b. RHEL6: `source /nfs/farm/g/glast/software/conda/bin/activate calibrator`
6. Run the `update_l1_calibration_database.py` script with the time in UTC for the new calibration(s) to take effect and the names of the one or more calibration files. It will update the database and print out the last two rows for each calibration type, so you can see the new file and the updated vend (stop) time of the previous calibration. Here's an example that updates all the usual CAL files:

Example run

```
(calibrator) [horner@rhel6-64l] ~ % update_l1_calibration_database.py "2022-04-21 22:30:00"
fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml fit_proton_calib_664m_668m_bigsum.calMPD.xml
pedavr_664m_668m.xml

Processing fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml.
Rows changed/added in database for CAL_Asym:
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
| ser_no |                                   data_ident                                  |
vstart       |         vend        |     update_time     |
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
|  1294  | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_658m_662m_bigsum.gcr_asym_hist.xml | 2022-02-17
23:00:00 | 2022-04-21 22:30:00 | 2022-04-21 14:12:07 |
|  1295  | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml | 2022-04-21
22:30:00 | 2037-01-01 00:00:00 | 2022-04-21 14:12:07 |
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+

Processing fit_proton_calib_664m_668m_bigsum.calMPD.xml.
Rows changed/added in database for CAL_MevPerDac:
+--------+-----------------------------------------------------------------------
+---------------------+---------------------+---------------------+
| ser_no |                                 data_ident                              |
vstart       |         vend        |     update_time     |
+--------+-----------------------------------------------------------------------
+---------------------+---------------------+---------------------+
|  1293  | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_658m_662m_bigsum.calMPD.xml | 2022-02-17 23:00:
00 | 2022-04-21 22:30:00 | 2022-04-21 14:12:07 |
|  1296  | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_664m_668m_bigsum.calMPD.xml | 2022-04-21 22:30:
00 | 2037-01-01 00:00:00 | 2022-04-21 14:12:07 |
+--------+-----------------------------------------------------------------------
+---------------------+---------------------+---------------------+

Processing pedavr_664m_668m.xml.
Rows changed/added in database for CAL_Ped:
+--------+-------------------------------------------------+---------------------+---------------------
+---------------------+
| ser_no |                    data_ident                   |        vstart       |         vend
|     update_time     |
+--------+-------------------------------------------------+---------------------+---------------------
+---------------------+
|  1292  | $(LATCalibRoot)/CAL/p7repro/pedavr_658m_662m.xml | 2022-02-17 23:00:00 | 2022-04-21 22:30:00
| 2022-04-21 14:12:07 |
|  1297  | $(LATCalibRoot)/CAL/p7repro/pedavr_664m_668m.xml | 2022-04-21 22:30:00 | 2037-01-01 00:00:00
| 2022-04-21 14:12:07 |
+--------+-------------------------------------------------+---------------------+---------------------
+---------------------+

Calibration database successfully updated.
```

If there are any problems, the script will print out an error message and stop, e.g.,:

Error Examples

```
(calibrator) [horner@rhel6-64l] ~ % update_l1_calibration_database.py "2022-04-22 18:30:00"
fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml fit_proton_calib_664m_668m_bigsum.calMPD.xml
pedavr_664m_668m.xml

Processing fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml.
Error: The table has 1 row(s) with same calibration file.
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
| ser_no |                                   data_ident                                  |
vstart       |         vend        |     update_time     |
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
|  1295  | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml | 2022-04-21
22:30:00 | 2037-01-01 00:00:00 | 2022-04-21 14:12:07 |
+--------+-------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
Exiting script.

(calibrator) [horner@rhel6-64l] ~ % update_l1_calibration_database.py "2022-04-21 18:30:00"
fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml fit_proton_calib_664m_668m_bigsum.calMPD.xml
pedavr_664m_668m.xml

Processing fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml.
Error: Input date is before current UTC time.
```

The script also has a noaction (aka dry-run) option that will show you what it would have done but not actually update the database, e.g.,

Example of dry-run

```
(calibrator) [horner@cent7a] ~ % update_l1_calibration_database.py "2022-04-26 23:00:00"
LAT_BadStrips_64.xml -n

Processing LAT_BadStrips_64.xml.
No action set.  Would have run SQL command:
insert into metadata_v2r1 (instrument, calib_type, flavor, data_fmt, data_size,vstart, vend, locale,
fmt_version, completion, proc_level, creator, uid, data_ident, enter_time) values (%s,%s,%s,%s,%s,%s,%s,%
s,%s,%s,%s,%s,%s,%s,%s)
with values:
('LAT', 'TKR_DeadChan', 'L1current', 'XML', 66985, '2022-04-26 23:00:00', '2037-01-01 00:00:00', 'SLAC',
'v2r1', 'OK', 'PROD', 'update_l1_calibration_database.py', 'horner', '$(LATCalibRoot)/TKR
/LAT_BadStrips_64.xml', datetime.datetime(2022, 4, 26, 20, 43, 44, 121595, tzinfo=datetime.timezone.utc))
No action set.  Would have run command:
update metadata_v2r1 set vend=%s where ser_no = %s
with values:
('2022-04-26 23:00:00', 1287)
No action set. No rows changed.  Last entry is:
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+
| ser_no |                data_ident                |        vstart       |         vend        |
update_time     |
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+
|  1287  | $(LATCalibRoot)/TKR/LAT_BadStrips_63.xml | 2021-11-02 13:40:00 | 2037-01-01 00:00:00 | 2021-
11-02 16:32:38 |
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+

Calibration database successfully updated.
```

You can then check the database with the show_l1_selection.py script. For example, after you run the update script but before the calibration start time, it will show the previous calibration file being selected, but you can see the vend time is set to your start time.

Example of show_l1_selection.py

```
(calibrator) [horner@rhel6-64l] ~ % show_l1_selection.py
Doing database lookups for CAL_Asym
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1294  |
+--------+
Second query to get file name returns:
+----------+-------------+-----------------------------------------------------------------------------------+
| data_fmt | fmt_version |                                     data_ident                                    |
+----------+-------------+-----------------------------------------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_658m_662m_bigsum.gcr_asym_hist.xml |
+----------+-------------+-----------------------------------------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-02-17 23:00:00 | 2022-04-21 22:30:00 |
+---------------------+---------------------+


Doing database lookups for CAL_MevPerDac
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1293  |
+--------+
Second query to get file name returns:
+----------+-------------+-------------------------------------------------------------------------+
| data_fmt | fmt_version |                                data_ident                               |
+----------+-------------+-------------------------------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_658m_662m_bigsum.calMPD.xml |
+----------+-------------+-------------------------------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-02-17 23:00:00 | 2022-04-21 22:30:00 |
+---------------------+---------------------+


Doing database lookups for CAL_Ped
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1292  |
+--------+
Second query to get file name returns:
+----------+-------------+---------------------------------------------------+
| data_fmt | fmt_version |                    data_ident                     |
+----------+-------------+---------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/pedavr_658m_662m.xml |
+----------+-------------+---------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-02-17 23:00:00 | 2022-04-21 22:30:00 |
+---------------------+---------------------+


Doing database lookups for TKR_DeadChan
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1291  |
+--------+
```

```
Second query to get file name returns:
+----------+-------------+-----------------------------------------+
| data_fmt | fmt_version |                data_ident               |
+----------+-------------+-----------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/TKR/LAT_BadStrips_64.xml |
+----------+-------------+-----------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-02-17 23:00:00 | 2037-01-01 00:00:00 |
+---------------------+---------------------+
```

If you set the time you're asking about (with the -s or --start option) to after the start time you set, then it will show the new calibration file as the one the pipeline will choose (note in this example the tracker file was not updated).

After update start

```
(calibrator) [horner@rhel6-64l] ~ % show_l1_selection.py -s "2022-04-22 00:00:00"
Doing database lookups for CAL_Asym
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1295  |
+--------+
Second query to get file name returns:
+----------+-------------+-------------------------------------------------------------------------------------+
| data_fmt | fmt_version |                                     data_ident                                      |
+----------+-------------+-------------------------------------------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml |
+----------+-------------+-------------------------------------------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-04-21 22:30:00 | 2037-01-01 00:00:00 |
+---------------------+---------------------+


Doing database lookups for CAL_MevPerDac
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1296  |
+--------+
Second query to get file name returns:
+----------+-------------+-------------------------------------------------------------------------+
| data_fmt | fmt_version |                                data_ident                               |
+----------+-------------+-------------------------------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_664m_668m_bigsum.calMPD.xml |
+----------+-------------+-------------------------------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-04-21 22:30:00 | 2037-01-01 00:00:00 |
+---------------------+---------------------+


Doing database lookups for CAL_Ped
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
```

```
|  1297  |
+--------+
Second query to get file name returns:
+----------+-------------+-------------------------------------------------+
| data_fmt | fmt_version |                    data_ident                   |
+----------+-------------+-------------------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/CAL/p7repro/pedavr_664m_668m.xml |
+----------+-------------+-------------------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-04-21 22:30:00 | 2037-01-01 00:00:00 |
+---------------------+---------------------+


Doing database lookups for TKR_DeadChan
First query to get ser_no returns:
+--------+
| ser_no |
+--------+
|  1291  |
+--------+
Second query to get file name returns:
+----------+-------------+-----------------------------------------+
| data_fmt | fmt_version |                data_ident               |
+----------+-------------+-----------------------------------------+
|   XML    |     v2r1    | $(LATCalibRoot)/TKR/LAT_BadStrips_64.xml |
+----------+-------------+-----------------------------------------+
Third query to get vstart and vend returns:
+---------------------+---------------------+
|        vstart       |         vend        |
+---------------------+---------------------+
| 2022-02-17 23:00:00 | 2037-01-01 00:00:00 |
+---------------------+---------------------+
```

You can also use the view_l1_calibration_database.py to see the status of the database:

Example of view_l1_calibration_database.py

```
(calibrator) [horner@rhel6-64l] ~ % view_l1_calibration_database.py
Latest entries in database for CAL_Asym:
+--------+-----------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
| ser_no |                                     data_ident                                    |
vstart       |         vend        |     update_time     |
+--------+-----------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
|  1294  | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_658m_662m_bigsum.gcr_asym_hist.xml | 2022-02-17 23:00:
00 | 2022-04-21 22:30:00 | 2022-04-21 14:12:07 |
|  1295  | $(LATCalibRoot)/CAL/p7repro/fit_gcrhists_lkhd_664m_668m_bigsum.gcr_asym_hist.xml | 2022-04-21 22:30:
00 | 2037-01-01 00:00:00 | 2022-04-21 14:12:07 |
+--------+-----------------------------------------------------------------------------------
+---------------------+---------------------+---------------------+
Latest entries in database for CAL_MevPerDac:
+--------+-------------------------------------------------------------------------+---------------------
+---------------------+---------------------+
| ser_no |                                data_ident                                |        vstart
|         vend        |     update_time     |
+--------+-------------------------------------------------------------------------+---------------------
+---------------------+---------------------+
|  1293  | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_658m_662m_bigsum.calMPD.xml | 2022-02-17 23:00:00 |
2022-04-21 22:30:00 | 2022-04-21 14:12:07 |
|  1296  | $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_664m_668m_bigsum.calMPD.xml | 2022-04-21 22:30:00 |
2037-01-01 00:00:00 | 2022-04-21 14:12:07 |
+--------+-------------------------------------------------------------------------+---------------------
+---------------------+---------------------+
Latest entries in database for CAL_Ped:
+--------+---------------------------------------------------+---------------------+---------------------
+---------------------+
| ser_no |                   data_ident                      |        vstart       |         vend        |
update_time     |
+--------+---------------------------------------------------+---------------------+---------------------
+---------------------+
|  1292  | $(LATCalibRoot)/CAL/p7repro/pedavr_658m_662m.xml | 2022-02-17 23:00:00 | 2022-04-21 22:30:00 | 2022-
04-21 14:12:07 |
|  1297  | $(LATCalibRoot)/CAL/p7repro/pedavr_664m_668m.xml | 2022-04-21 22:30:00 | 2037-01-01 00:00:00 | 2022-
04-21 14:12:07 |
+--------+---------------------------------------------------+---------------------+---------------------
+---------------------+
Latest entries in database for TKR_DeadChan:
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+
| ser_no |               data_ident                |        vstart       |         vend        |
update_time     |
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+
|  1287  | $(LATCalibRoot)/TKR/LAT_BadStrips_63.xml | 2021-11-02 13:40:00 | 2022-02-17 23:00:00 | 2022-02-17 11:
47:00 |
|  1291  | $(LATCalibRoot)/TKR/LAT_BadStrips_64.xml | 2022-02-17 23:00:00 | 2037-01-01 00:00:00 | 2022-02-17 11:
47:00 |
+--------+-----------------------------------------+---------------------+---------------------
+---------------------+
```

Post Update Steps

These are the steps you should do after running a successful update.

1. Send an email to whoever created the calibration files and cc the data monitor list (datamonlist@glast4.stanford.edu) to let them know when the calibration should take effect.
    a. Get the ID of the next run by using XTime or similar tool to convert start time of the next run after the SAA pass into MET.
    b. Include the start time and first run in the email, e.g., "vstart is 2018-09-28 03:43. First run should be 559799706."
2. Update the JIRA issue with the same information.
3. **Later** check that the new calibration has been used.
    a. Go to data processing page and click on the L1Proc bar for the run after the change should have happened.
    b. Go to the "Substreams" table in the middle of the page and click on one of the streams for the "doChunk" task.

    c. Go down to the "Substreams" table and click on any one of the "doCrumb" streams.
    d. Click on the log file link for the "recon" process.
    e. Search in the log for appropriate XML file(s).
4. Assuming everything is good, update and close the JIRA issue.

Troubleshooting

This section is to help troubleshoot any errors.

L1Proc failing

The new script should prevent errors like this, but if the L1Proc deliveries start to fail after the calibration change takes effect, look at the logs mentioned above and search for any errors. If you see something like:

> *XmlBaseCnv FATAL Unable to parse document $(LATCalibRoot)/CAL/p7repro/fit_proton_calib_634m_638m_bigsum.calMPD.xml*
> *aka /afs/slac/g/glast/ground/releases/calibrations//CAL/p7repro/fit_proton_calib_634m_638m_bigsum.calMPD.xml*

then that calibration file was not actually copied to the TKR or CAL directories described above. Copy it there and then either contact someone to rollback the process or do it yourself following the instructions in Things to know while on-call for Data Processing. In this case, a command line rollback of delivery 210427007:

> */afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD rollbackStream --minimum 'L1Proc[210427007]'*

was all that was needed to successfully process the delivery.

Further troubleshooting instructions will be added as issues arise.