

Pipeline errors and solutions

- [Can't open lockfile](#)
- [Handling Stuck Jobs](#)
- [Checking Host Status](#)
- [Running a command directly on a Host](#)
- [Failure to execute java script](#)
- [Failure in checkRun](#)
- [Job fails to start \(stays in PEND status\)](#)
- [Handling a host with lots of hung jobs](#)
- [Files missing from NFS file server](#)
- [Throttling the Pipeline](#)
- [Changing the Reaper Settings](#)
- [HalfPipe marked as Failed but L1 started successfully](#)
- [Files missing from NFS and xrootd servers](#)
- [Multiple Submissions of the same Job](#)
- [HalfPipe progress bar field has red question mark](#)
- [Unknown LSE_Keys error](#)
- [Set up a rollback when jobs still running](#)
- [cleanupCompleteRun doesn't start after last run processes](#)
- [MeritHist - Different number of events in Merit chain](#)
- [GlastCalibDB \(mysql-node03\) is down \(Halting a specific Task, like L1Proc\)](#)
- [getTimes Terminates](#)
- [MOOT Key Mismatch between on-board LAT and pipeline expectation](#)
- [Problems with Magic 7 File](#)

Can't open lockfile

This message was received:

```
Event type: L1Proc.unLockRun
Event timestamp: 2017-10-30 21:57:42.474089
Spacecraft ID: -1
Target: None
Trigger name: L1-processing
Cluster size: 1
Message text: Can't open lockfile cd /r0531060188.lock.
```

Status in data monitoring pages:

The process was listed as terminated and has this java error:

```
org.srs.jobcontrol.JobControlException: Remote Exception performing operation
    at org.srs.jobcontrol.JobControlClient.checkException(JobControlClient.java:219)
    at org.srs.jobcontrol.JobControlClient.submit(JobControlClient.java:77)
    at org.srs.pipeline.server.process.batch.BatchProcessExecutor.run(BatchProcessExecutor.java:202)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:745)
Caused by: java.net.SocketTimeoutException: Read timed out
    at java.net.SocketInputStream.socketRead0(Native Method)
    at java.net.SocketInputStream.read(SocketInputStream.java:152)
    at java.net.SocketInputStream.read(SocketInputStream.java:122)
    at java.io.BufferedInputStream.fill(BufferedInputStream.java:235)
    at java.io.BufferedInputStream.read(BufferedInputStream.java:254)
    at java.io.DataInputStream.readByte(DataInputStream.java:265)
    at sun.rmi.transport.StreamRemoteCall.executeCall(StreamRemoteCall.java:214)
    at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:161)
    at java.rmi.server.RemoteObjectInvocationHandler.invokeRemoteMethod(RemoteObjectInvocationHandler.java:
194)
    at java.rmi.server.RemoteObjectInvocationHandler.invoke(RemoteObjectInvocationHandler.java:148)
    at com.sun.proxy.$Proxy7.submit(Unknown Source)
    at org.srs.jobcontrol.JobControlClient.submit(JobControlClient.java:74)
    ... 4 more
```

Seems to be an NFS error. I can't write to that directory as myself or the glast account. Need to check on proper account/permissions.

Handling Stuck Jobs

A stuck job usually shows up in a bjobs list as having consumed no CPU or memory, and in an UNKWN state and has been sitting around for a long time (depends on job, need more data).

The procedure is to log on as glastraw and bkill them. As Warren explained:

You can't roll back a running job. You have to bkill them and wait for the reaper to notice that they're dead. The pipeline will automatically retry them once for L1 (0 is the default, you can ask for up to 3). If the jobs you killed were retries, then you'll have to roll them back. I've never seen a job in unknown state recover by itself.

Checking Host Status

Use `bhosts` to check the host status (although you might want to `grep` with the host name as the list is long).

Running a command directly on a Host

From Warren:

You can just immediately run a command on any batch host with `lsrun`. Obviously this should be used sparingly. `uptime` is a good cheap thing that will show you if the host is responsive and what the load is. Or check to see if a particular filesystem is causing trouble, or if some command works.

Failure to execute java script

When a script gets terminated and viewing messages includes the error message "failure when executing java...(Need to get exact text next time it happens)". This is typically an indication of a bad variable in the environment/database that the process is running it. This bad value is typically set by the upstream process having run in some strange manner and mangling the variable. In every case so far that I've seen, the upstream process ran twice simultaneously on two different hosts and so the two processes were overwriting each other. This is fairly easy to identify as the output log contains all the execution output (except possibly the environment output) twice and there are two different LSF summary blocks at the end of the log.

The solution is to simply roll back the upstream process so it gets a clean execution.

Failure in checkRun

Check run failed and I thought at first it was a case of bad data as the process ran simultaneously on two different hosts. However a simple rollback did not solve the problem as there was a lock left from running the `mergeCalChunks` upstream process. This lock was left because of an infrastructure glitch preventing proper access to the xroot data disks. Rerunning that process removed the lock file but `checkRun` still didn't work, complaining about the existence of a `dontCleanUp` file. This file was actually left from the bad `mergeCalChunks` run. Before deleting it we need to check that the files listed there actually exist. This command:

```
for ff in $(egrep '^root' <path to>/dontCleanUp) ; do xrd.pl --where stat $ff ; done
```

will poll the xroot server for information on the files listed in the `dontCleanUp` file.

If they are all there, we are good to go and can remove the `dontCleanUp` file and rollback the `checkRun` process. If not, we need to look further upstream in the data processing chain.

Note: the `xrd.pl` in the above command is found at `/afs/slac.stanford.edu/g/glast/applications/xrootd/PROD/bin/xrd.pl`

Job fails to start (stays in PEND status)

The particular event that triggered this scenario was the two deliveries of run 543796948. The first delivery (180315009) had several recon jobs that ended up on a dead host and had to be killed. While they were hung, the rest of the events in the run arrived in delivery 180315010 and the `findChunks` task was in a pending state waiting for its pre-exec command (`/afs/slac/g/glast/isoc/flightOps/rhel6/gcc44/ISOC_PROD/bin/isoc run /afs/slac.stanford.edu/g/glast/ground/PipelineConfig/Level1/5.6/lockFile.py`) to give it the go ahead. After the hung recon tasks were killed, the reaper properly restarted them and the run completed and properly cleaned up after itself. However, the `findChunks` task never started up. LSF normally retries the pre-exec command regularly (every 2.5 minutes in this case - **how to we check?**) but had not run it in over 2 hours (**again how to we check? is there a max retries?**).

When this happens:

1. First try stopping and restarting the job (`bstop` and `bresume`). This will often kickstart things and get them going again
2. If that doesn't work, just *bkill* the process and let the reaper resubmit it (Which is what we had to do in this case).

Handling a host with lots of hung jobs

This is only a partial description of handling this. This comes up when there are a large number of jobs in an UNKWN state all on the same host. Often, looking at the status of the host (`bhosts <hostname>`) will tell you it is unavailable, in which case you're probably safe. If it is still open and accepting jobs, however, you want to prevent it from getting new batch jobs as they will just hang as well. From Warren your options are:

Close the host.

```
sudo badmin -C "Witty comment" <hostname>
```

Which requires sudo permission which you probably don't have yet and they'll want to limit to some machine we own and we should find a box that's not my workstation to use for that.

Fill it up with dummy jobs.

```
for ii in $(seq 8) ; do bsub -q glastdataq -sp 90 -m <hostname> sleep 86400
```

in bash, as glastraw

So probably really only one realistic option at the moment.

Also you should probably notify unix-admin.

Files missing from NFS file server

You may need to do something with files on XrootD. See the [Intro to using xrootd](#).

This one came up the incident reference by this thread in the opsprob mailing list: <https://www-glast.stanford.edu/protected/mail/opsprob/11338.html>

Basically several jobs were hung, when they were killed, they still didn't work and rolling things back made it worse. In the end, we needed to copy the files back from xroot to NFS and restart the process. Some questions, comments, and answers from the thread ([some of these still need answers](#)):

1. How do we know that? Is it just because the logs complain that they are not there? How do we check for their existence?
2. Where do they live to copy them from?
3. I'm assuming they need to get copied to directory that it can't find them in ([root://glast-rdr.slac.stanford.edu/glast/Scratch/l1Stage/runs/542/r0542185591/e00000000000002729588/event/](#) in this case). Or do they go somewhere else?
4. How do we perform the copy? *See #10 below.*
5. I'm guessing we'll need to move/rename the existing chunkList file so a new one can be created at this point? Is this correct? (BTW the notes say we should have a section on chunkList files that no one has written yet)
6. Where do we roll back at to get everything going again? Just a general rollback on the command line? Or is there a specific task that can be rolled back to kick everything off properly again?
7. The .evt files reference by the chunkList file (in `/nfs/farm/g/glast/u28/stage/180308007/r0542185591/`) don't exist.
They've been moved to xroot. It really would be better if they were copied, I [Warren] don't remember why I get discouraged every time I try to make that change.
8. The directory in xroot ([root://glast-rdr.slac.stanford.edu/glast/Scratch/l1Stage/runs/542/r0542185591/e00000000000002729588/event/](#)) only contains a single evt file: `r0542185591_e000000000000002729588_v246043240_event.evt`. The failed jobs are complaining about other files which are missing.
Each was moved to it's own directory.
9. In the `/nfs/farm/g/glast/u41/L1/deliveries/1803` directory, there is no entry for this delivery (180308007). The entries for 180308006 and 180308008 are there though. Found it. It's in the `/nfs/farm/g/glast/u28/stage` directory. I looks like that gets moved over to deliveries once it's done.
10. I still can't seem to find the original evt files.
*Each has a directory corresponding to the chunk number. The second-to-last component in the path you give above. I use a command like this to generate a script to move them back to NFS: "awk '/\afs.*xrscp/{print \$1, \$2, \$3, \$5, \$4}' /nfs/farm/g/glast/u41/L1/logs/PROD/L1Proc/5.5/doRun/findChunks/160xxxxxx/515xxx/011/485xxxxxx/001xxx/694/archive/198092024/logFile.txt*

Here's an example the output for the issue in <https://www-glast.stanford.edu/protected/mail/opsprob/15272.html>

Example

```
% awk '/^\/afs.*xrscp/{print $1, $2, $3, $5, $4}' /nfs/farm/g/glast/u41/L1/logs/PROD/L1Proc/5.9/doRun
/findChunks/220xxxxxx/805xxx/013/681xxxxxx/413xxx/156/archive/342431233/logFile.txt | tee cp.sh
/afs/slac.stanford.edu/g/glast/applications/xrootd/PROD/bin/xrscp -np -f root://glast-rdr.slac.stanford.edu
//glast/Scratch/l1Stage/runs/681/r0681413156/e00000000000009129671/event
/r0681413156_e00000000000009129671_v342431233_event.evt /nfs/farm/g/glast/u28/stage/220805013/r0681413156
/r0681413156-e00000000000009129671.evt
/afs/slac.stanford.edu/g/glast/applications/xrootd/PROD/bin/xrscp -np -f root://glast-rdr.slac.stanford.edu
//glast/Scratch/l1Stage/runs/681/r0681413156/e00000000000008510966/event
/r0681413156_e00000000000008510966_v342431233_event.evt /nfs/farm/g/glast/u28/stage/220805013/r0681413156
/r0681413156-e00000000000008510966.evt
% cat cp.sh
/afs/slac.stanford.edu/g/glast/applications/xrootd/PROD/bin/xrscp -np -f root://glast-rdr.slac.stanford.edu
//glast/Scratch/l1Stage/runs/681/r0681413156/e00000000000009129671/event
/r0681413156_e00000000000009129671_v342431233_event.evt /nfs/farm/g/glast/u28/stage/220805013/r0681413156
/r0681413156-e00000000000009129671.evt
/afs/slac.stanford.edu/g/glast/applications/xrootd/PROD/bin/xrscp -np -f root://glast-rdr.slac.stanford.edu
//glast/Scratch/l1Stage/runs/681/r0681413156/e00000000000008510966/event
/r0681413156_e00000000000008510966_v342431233_event.evt /nfs/farm/g/glast/u28/stage/220805013/r0681413156
/r0681413156-e00000000000008510966.evt
```

From Michael:

The log file is of the findChunks instance that copied the evt files but didn't finish for whatever reason.

It's prudent to check that the script looks reasonable. I execute it as me, it never was necessary to become glastraw. After that remove the run lock and roll back findChunks.

Throttling the Pipeline

If you ever need to limit the amount of work being done on the pipeline (like we wanted to with the LAT restart in April 2018), you can manually create throttle locks to limit the number of simultaneous runs that can be worked on at a time. Right now the pipeline is set to allow up to 6 runs to be worked on at once. If you want to limit that, simply create lock files in the `/nfs/farm/g/glast/u41/L1/throttle` directory of the form `0.lock`, `1.lock`, ... up to `5.lock`. The contents can be anything you want. It is just the presence of the file that stops things from running. Each lock file created will reduce the number of simultaneous runs by one. Creating all six will stop the pipeline from processing anything.

Changing the Reaper Settings

There are some variables found on the JMX page of the pipeline that control the operation of the reaper (the process that looks for dead process, cleans them up, and restarts them).

The first is the **ReaperFrequencySeconds** in the *Control* section at the top. This parameter controls how often the reaper actually runs. Currently set to 60 (once per minute).

The other parameter is set on a per submission source basis. This is the **ReaperDelayMinutes** parameter and is found in each of the later sections on that page. This controls how long a process has to be dead for before the reaper kills it. It is typically set to 60 or 120 minutes. *NOTE (from Warren): I'm not convinced that ReaperDelayMinutes actually does anything. Restarting the pipeline set both of them back to default.*

HalfPipe marked as Failed but L1 started successfully

This came up with delivery 180409011. In this case the launchL1 task was started simultaneously on two different hosts in the batch queue. The first one ran successfully but since the second one failed with an error saying the L1 stream already existed, the process was marked as failed. To clean up the data display do the following:

1. Create a lock file in the appropriate downlink directory: `/nfs/farm/g/glast/u42/ISOC-flight/Downlinks/<downlink id>/haltL1`
2. Check to see if the input directory has moved. It is normally at `/nfs/farm/g/glast/u28/stage/<downlink id>` but L1 may have moved it to `/nfs/farm/g/glast/u41/L1/deliveries/<YYMM>/<downlink id>` where YYMM is just the first 4 digits of the downlink id. If it has moved, just create a symbolic link at the old location. e.g. `ln -s /nfs/farm/g/glast/u41/L1/deliveries/1804/180409011 /nfs/farm/g/glast/u28/stage/180409011`
3. Rollback the launchL1 process so it runs properly.
4. Remove the lock file and the symbolic link.

Files missing from NFS and xrootd servers

This has come up twice now under slightly different circumstances. See the following threads for the OpsProbList chatter:

- First time
 - <https://www-glast.stanford.edu/protected/mail/opsprob/10518.html>
- Second time (this is split between various threads and mailing lists)
 - <https://www-glast.stanford.edu/protected/mail/opsprob/11528.html>
 - <https://www-glast.stanford.edu/protected/mail/datamon/18217.html>
 - <https://www-glast.stanford.edu/protected/mail/opsprob/11532.html>

The end result seems to be that we need to repipe the runs in question. Details (from Warren) on that process (based on the second event):

The evt files weren't on xroot because cleanupChunks Stream 180502013.546962321.546962321 deleted them. I've started a RePipe:

```
$> run=546962321
$> dl=180502012
$> ~glast/pipeline-II/prod/pipeline createStream --stream $run --define downlinkID=$dl,runID=$run RePipe
```

RePipe finished. then:

```
$> cd /nfs/farm/g/glast/u28
$> mv RePipe/$dl/r0$run/*.evt stage/$dl/r0$run
$> cd /nfs/farm/g/glast/u41/L1/runs/546/r0546962321/
$> cd /nfs/farm/g/glast/u41/L1/runs/546/r0546962321/
$> /afs/slac.stanford.edu/g/glast/ground/PipelineConfig/Level1/5.6/newChunkList.py /nfs/farm/g/glast/u28/stage/$dl/r0$run
$> mv r0546962321_180502012_chunkList.txt.new r0546962321_180502012_chunkList.txt
```

You'll have to be glastraw for the last 2 steps.

But then I messed up. I rolled back fondChunks 180502012.546962321 and it failed complaining about overlapping data. The right thing is to roll back the whole doRun stream while defining deliveriesToIgnore=180502013 ([How?](#)). Which I've now done.

And now there are more errors, which I'll have to investigate later, but probably involve the magic7 file.

Multiple Submissions of the same Job

This is usually indicated by one or more instances of the LSF summary report appearing near the end (although not always) of the file. The summary looks like this:

```
-----
Sender: LSF System <lsf@hequ0119>
Subject: Job 217648: <findChunks> in cluster <slac> Done

Job <findChunks> was submitted from host <fermilnx-v08> by user <glastraw> in cluster <slac> at Thu Aug  2 11:37:06 2018.
Job was executed on host(s) <hequ0119>, in queue <glastdataq>, as user <glastraw> in cluster <slac> at Thu Aug  2 11:37:08 2018.
</u/gl/glastraw> was used as the home directory.
</nfs/farm/g/glast/u41/L1/logs/PROD/L1Proc/5.6/doRun/findChunks/180xxxxxx/802xxx/015/554xxxxxx/915xxx/869> was used as the working directory.
Started at Thu Aug  2 11:37:08 2018.
Terminated at Thu Aug  2 11:39:29 2018.
Results reported at Thu Aug  2 11:39:29 2018.
```

Your job looked like:

```
-----
# LSBATCH: User input
bash pipeline_wrapper
-----
```

Successfully completed.

Resource usage summary:

CPU time :	8.33 sec.
Max Memory :	66 MB
Average Memory :	24.33 MB
Total Requested Memory :	-
Delta Memory :	-
Max Swap :	-
Max Processes :	8
Max Threads :	13
Run time :	141 sec.
Turnaround time :	143 sec.

The output (if any) is above this job summary.

In 95% of the cases, all that needs to be done is a simple rollback of that affected process. In the other 5%, there is some other underlying problem that is also affecting the completion of the job and you'll need to search through the log files for the error.

HalfPipe progress bar field has red question mark

This occurs when there are no files for the half-pipe to process in the delivery. There are three causes for this:

1. There really is no data. Looking at the list of files in the delivery (click on the progress bar for FastCopy on that delivery) shows that there are no *.LSCI.* files in the delivery. Nothing needs to be done.
2. It's duplicate data. An example of this is delivery [18729012](#). In this case the same files (but with a .02 version number) were delivered that had come in a previous delivery (in this case delivery [18729011](#)). The system determined that there were no new events so the halfpipe wasn't started. Nothing needs to be done.
3. The half-pipe failed to start. For some reason, The FastCopy processing successfully completed but the half pipe failed to launch. When this happens, and the delivery contains science data follow the instructions on the [HTF FC Incoming](#) page for "Dispatching Runs" (item 2). This will launch the half-pipe for that run.

Unknown LSE_Keys error

The symptom of this is that you get an error of the form:

```
LSEReaders::read: unknown LSE_Keys typeid 551075433 from /nfs/farm/g/glast/u42/ISOC-flight/Downlinks/180619008/0000f5a2-20d8be69-03bd-00a21.evt
```

In this particular case the LSE_Key was set to the run number instead of it's proper value (-1 through 3 from an enum in the code).

To date, I've only ever seen this in the mergeEvt task. This is symptomatic of a problem upstream in the doChunk streams (running the makeEvt task). In every instance, I've found one or more of those makeEvt tasks had a multiple submission. Rolling all the doChunk streams back with the multiple submissions fixes the problem.

Set up a rollback when jobs still running

Sometimes you need to roll back a stream but there are jobs still running that prevent the rollback from occurring immediately. Warren provided this little tidbit to set up the rollback and not have to keep checking back:

When a run is not ready for rollback, I usually do this:

```
$> del=<delivery #>
```

```
$> until ~glast/pipeline-II/prod/pipeline rollbackStream --minimum --force L1Proc[$del] ; do sleep 300 ; done  
(in bash) so I don't have to keep checking.
```

I haven't actually used this yet so I'll update it (or remove this line) when I can verify that it works.

cleanupCompleteRun doesn't start after last run processes

From Michael:

For each chunk, the individual checkChunk should remove the respective chunk token. CheckChunk of chunk 1757481 reports it can't find the token (nfs hiccup) but proceeds by design.

After all the merges etc. checkRun checks for remaining tokens and if present chokes on it. To resolve it I usually just remove the chunk token in question (checking that the chunk was processed properly, of course), and roll back the youngest (if there are more than one) checkRun. If there isn't any other issue checkRun sets the status to Complete, which triggers cleanupCompleteRun.

It's overkill to roll back from the checkChunk. Maybe even harmful, but I guess the chunk data are still on xrootd. But this should be verified.

MeritHist - Different number of events in Merit chain

This has happened a few times. For instance,

<https://www-glast.stanford.edu/protected/mail/opsprob/15247.html>

<https://www-glast.stanford.edu/protected/mail/opsprob/14337.html>

<https://www-glast.stanford.edu/protected/mail/opsprob/11489.html>

A typical rollback probably won't work. It will likely need to be rolled back from an earlier process like scanDigi.

GlastCalibDB (mysql-node03) is down (Halting a specific Task, like L1Proc)

This mysql server hosts the GlastCalibDB database where the L1 pipeline looks up the calibration file to use. If it's down or needs to be taken down (e.g., to install an update), Brian left [instructions](#) to halt job submission for L1Proc so that other tasks may run as normal.

Stopping a single task:

- You need to be a Pipeline admin to do this.
- Navigate to https://glast-ground.slac.stanford.edu/Pipeline-II/exp/Fermi/admin_jmx.jsp.
- Select jobsite you expect jobs to run at (e.g., "SLACDATA").
- Find the "setJobsTotalLimitForTask" button.
- Add your task name (e.g., "L1Proc") in the p1 parameter box next to it.
- Set the value to 0 in the p2 parameter box.
- Click the setJobsTotalLimitForTask button.

This should suspend job submission and leave processes in the Waiting state.

To clear it (and all other limits that were set for the jobsite):

- Put "true" in the parameter box for "userClearLimits" (in the same widget) and then click the button. However, there seems to be a bug, and this may not work. According to Brian userClearLimits will clean out a file that is stored, but it won't clear out the in-memory state. The parameter userStoreLimits is what writes the file, persisting the in memory state. Without the file, it will forget limits that are set in the pipeline in the memory on restart.
- If just setting it to "true" doesn't restart processing:
 - Use the [admin page](#) to restart the pipeline server. According to Brian userClearLimits with "true" and a restart works because you delete the files and pick up database configs on restart.
 - You can also try putting "false" in the box, and then "true".
 - Alternatively, set the limit to something high/reasonable, such as 6000.

getTimes Terminates

When this happens, it will leave a lock file behind (e.g., /nfs/farm/g/glast/u41/L1/runs/697/r0697981861/r0697981861.lock) that you will need to delete before rolling the job back. Otherwise, the rollback will stay queued but not run. If you do the rollback before removing the lock file you should bkill the job, wait for the reaper to terminate it, then remove the lock file, and do the rollback again.

MOOT Key Mismatch between on-board LAT and pipeline expectation

This will cause an error in the logs like:

```
MootSvc FATAL Hw key 8517 in data does not match hw key belonging to Moot Config #3063
```

This has happened in [2023](#), [2017](#), and [2013](#) due to a problem with an update to the hot strip mask on the satellite, e.g., the change was reverted but the pipeline was expecting the MOOT key for the new configuration. Following what was first done in 2013, the fix is to update the databases to change the MOOT key for the failed runs back to the old value and rollback the failures. Other stuff mentioned in the [original thread](#) doesn't seem necessary.

1) Change the moot_key in glastops_acqsummary in the [Oracle database](#).

- Setup an [ISOC environment terminal](#) and then run `sqlplus \@isocflight` to access the database. You will need standard pipeline maintainer permissions.
- For SQL commands, see: <https://www-glast.stanford.edu/protected/mail/opsprob/10899.html>

2) Change the config_fk (mootkey) in MOOD's Acq_summary in the [MOOD/FMX MySQL database](#).

- To access the database, you need to login to fermilnx02 (in 2023) and issue the command `mysql -h glastMOOD -P8208 -u <username> -p`. You will need to get a password from the [secret/fpd/fermi/isoc vault](#). Richard may need to put in a ticket for you to get permission. In 2023, we used Steve Tether's account (tether) since the "magician" account is overpowered for what we need (it's basically the super-user account).
- For SQL commands, see <https://www-glast.stanford.edu/protected/mail/opsprob/10898.html>. Before you have to activate the db with `use mood;`.

3) In some receipts (e.g. in older emails) is written that a repipe is necessary. Michael doesn't recall repipes for the 2017 key mismatch. At least no helpful repipes. In 2023 for sure no repipes were performed. Don't repipe unless really necessary!

4) Rollback the affected deliveries

- If they are in a Failed state, you can just do a standard rollback.
- If findChunks is stuck, then bkill the process and wait for the reaper to get it. Most of them should then auto rollback without intervention.
- Most likely someone has created run locks manually to halt runs from being processed with wrong moot keys and successively failing. Don't forget to remove these run locks.

History

- [The MOOT Key Mismatch problem occurred on 2023 July 18](#). See the Fermi news item for **2023 July 20** at <https://fermi.gsfc.nasa.gov/ssc/library/news/>
 - 15 LPA runs from the LAT had to have their MOOT keys manually fixed in Oracle and MySQL before they could be processed correctly through L1:
 - 711402686, 711405073, 711411046, 711416992, 711422925, 711428850, 711434773, 711440537, 711446088, 711451782, 711457476, 711463169, 711468863, 711474557, 711478257
- [The MOOT Key Mismatch problem occurred on 2017 November 30](#). See: <https://www-glast.stanford.edu/protected/mail/opsprob/10893.html>

Problems with Magic 7 File

For instance, this error in ft2Runs stream 240415006.734877891:

```
> terminate called after throwing an instance of 'std::runtime_error'
> what(): FATAL: the provided Magic 7 file does not cover the requested time interval. To cover the requested interval we would need to extrapolate position and attitude (forward) more than what permitted by the current configuration (see the parameter 'extrapolationLimit').
```

Michael fixed this by:

1) doRun.ft2Runs (case of 240415006.734877891) reads from the runs area:
stageIn for: /nfs/farm/g/glast/u41/L1/runs/734/r0734877891/r0734877891_v000_magic7L1.txt

The magic7 file in the staging area, which is also being copied into the runs area, is complete. makeM7L1 reads from the staging area.

Thus, I just rolled back makeM7L1, which found all packets and created a valid magic7L1 file in the runs area, to be read by ft2Runs.

2) doRun.doChunk.fakeFT2 (case of 240414007.734792595.6674757) stages from the staging area:
stageIn for: /nfs/farm/g/glast/u28/stage/240414007/magic7_240414007.txt

This file was incomplete! I replaced it by the 240414008 magic7 file