

Things to know while on-call for Data Processing

- [Task of the Data Processing on-call expert](#)
- [Data Organization](#)
- [Different types of failures](#)
 - [Transient failures](#)
 - [Infrastructure failures](#)
 - [Permanent failures](#)
 - [Other failures](#)
- [Everything to know about the Rollback](#)
 - [How to rollback](#)
 - [Rolling back substreams](#)
 - [When to rollback](#)
 - [The dontCleanUp file](#)
 - [The run lock](#)
 - [HalfPipe locks](#)
 - [findChunks](#)
 - [The throttle lock](#)
 - [Suspending jobs](#)
 - [Skipping events](#)
 - [Skipping events when Gleam segfaults](#)
 - [Re-trying FASTCopy ingestion or ProcessSCI](#)
- [Useful resources](#)
- [Other things you may want to know](#)
 - [setL1Status task](#)
 - [setL1Quality task](#)
 - [How to contact SCCS](#)
 - [Log Watcher Messages](#)
 - [cancelProcessInstance](#)
 - [Host or host group is not used by the queue.](#)
 - [Red question marks in display](#)
 - [Multiple lines for run in a given delivery](#)
 - [Stuck processes](#)
 - [Repiping](#)
 - [How To Fix common FASTCopy problems](#)
 - [Batch commands](#)
 - [Shift sign up](#)
 - [Useful L1Proc links](#) (these are designed to keep you from waiting for the entire L1Proc history to be retrieved ... which could be 10 min and up).

Task of the Data Processing on-call expert

Monitor L1Proc and halfPipe: Every time a [red cross](#) appears on the [Data Processing page](#), next to the L1Proc or halfPipe processing status bar, try to resolve the failure. We are not on-call for ASP/GRB search (The ASP expert - currently Jim Chiang (jchiang@slac) - should be emailed (not paged) for these failures. The ASP expert is listed on the [How to Fix](#) page.) and we are definitely NOT on-call for infrastructure problems (can't see monitoring plots, etc.). Despite not being on-call for these issues, we will often fix issues that arise if simple (i.e. just a rollback). If you get paged for something that is not under your responsibility, don't try to fix it: forward the message to the appropriate people and report everything to the OpsProb mailing list (opsproblast2.stanford.edu).

Familiarize yourself with understanding the [Pipeline-II](#) page by reading through the [Pipeline-II User's Guide](#). It is a good starting point for understanding the general organization of the pipeline and the tools needed to track down problems.

It may be good to look at the [task chart](#) (*note: this one is in the Dev database and is old and out of date, needs updating*) to see the interdependencies of tasks ("tasks" defined in [Pipeline-II User's Guide](#)). "Success" dependency means that a process needs to successfully complete in order for the dependent process to continue, while "All Done" means that even failed processes will result in the dependent process continuing.

Watch the [Usage Plots](#) and look for L1Proc/HalfPipe related tasks (doChunk, doCrumb, etc). Default rule of thumb of time is to use last 2 hours, because more than that will not give enough fidelity in the plot. If you see a series of points that make a flat line for an extended period of time (anything over 1/2 hour is suspect but mergeReconCunks can take an hour or two), it may indicate problems with the pipeline.

Data Organization

There are four main categories of data organization. At the top, there is the "Delivery", which is the data that is sent down from GLAST. Completely unrelated are the "Runs", which are time-segments determined by GLAST. A delivery can consist of a part of a run, many runs, or pieces of runs - there is no particular order that is guaranteed within a delivery with regards to a run. Runs (or parts of the run contained in a delivery) are broken into "Chunks", which are always contiguous blocks of data. Chunks are further broken down into "Crumbs", which are also contiguous.

When looking at files or directories, run numbers are typically prefixed by an "r", chunk numbers with an "e", and crumb numbers with a "b".

Different types of failures

There are 3 main type of failures, and should be handled differently.

Transient failures

- afs hiccup (can't read from/write to a resource located on an afs disk)
- nfs hiccup (can't read from/write to a resource located on an nfs disk)
- LSF hiccup (an lsf batch machine crashes or is rebooted)
- lost Data Base connection : MOOT, Calibration database [log](#), or more rarely on Oracle
- recon segfaults for no apparent reason (nothing in logfile)
- Pipeline deadlock

How to recognize transient failures: they usually affect only one single job (disk or DB failures), or a small number of jobs all on the same machine (LFS failure). If a process complains about a missing file but the file exists, or gets a read error after opening a file, it's probably a disk server hiccup. If the log file (linked from the pipeline web client) ends suddenly, without the usual LSF postlog, the batch host probably crashed. There will probably be several jobs failing the same way at the same time.

What to do in case of transient failures: rollback the affected process(es) when possible (see below for the rollback procedure). Look for the dontCleanUp file and check the [Log Watcher](#) (see below) ([check for what?](#)). If recon segfaults for no apparent reason, email Heather (general emergency contacts are on the [How to Fix](#) page) before the rollback, including a link to the log file, which will tell them where the core file is. For pipeline deadlocks, email the Pipeline Expert (currently Brian) and include a link to the process instance.

Transient failures are rare lately. For the last couple of months, most failed processes are automatically retried once. This usually fixes transient issues, so usually when there's a failure it indicates an actual problem.

Bad merges: If a process that's merging crumb-level files into chunks or chunks into runs can't find all of its input files, it won't fail. See the "dontCleanUp" section below. Processes downstream of such a merge may fail because they are trying to use different types of input files (e.g., digi and recon) and the events don't match up because some are missing from one file and not the other. Then you need to roll back the merge even though it "succeeded" the first time.

Infrastructure failures

- single batch host is killing several jobs ('rogue' LSF host)
- group of batch hosts crashed or went offline
- afs crashed on one of the host machines
- scratch disk is full on one or more of the host machines - see what to do [here](#) ([This is a log of a conversation, probably posted from the OpsLog, should be made into a child page](#))
- A staging disk is full ([need instructions on how to check xroot files](#))
- /nfs/farm/g/glast/u41 or /nfs/farm/g/glast/u28 is full

How to recognize infrastructure failures: they usually affect a large number of jobs, either on the same LSF host or on different LSF hosts.

What to do in case of infrastructure failures: these failures involve a large number of people to be taken care of (the infrastructure expert on-call and often also the SCCS), so for the time being e-mail the appropriate on-call expert (see L1 shift schedule - [need to get link to new one once it's ready](#)) and the opsprob list if you think that one of those failures might be happening during the night (if in doubt, e-mail anyways).

Permanent failures

- G4 propagator error. If recon dies complaining about g4propagator, we can't fix it. If this happens, email Heather (heather625{at}gmail) ~~and Anders (borgland{at}slac)~~ (and possibly Warren (focke{at}slac) and Maria Elena (monzani{at}slac)). Include a link to the log file, which will tell them where the core file is.
- Overlapping chunks. If findChunks complains about overlapping chunks, tell Steve Tether; if trending merges complain, tell datamonlist@glast2.Stanford.EDU. Rollback won't help.
- Deliveries arriving too closely will mangle the [display](#). The underlying processes are fine, but email obsproblast{at}glast2.stanford.edu, Steve Tether ([need e-mail](#)) or Jim (jchiang{at}slac) to fix the display. ([Need instructions for this added to the HTF FC Incoming page](#))
- Digitization crashes. Sometimes we get a bad event and have to skip it. Instructions for that are in the rollback section below.

How to recognize permanent failures: besides those 2 cases, everything that doesn't get fixed after a rollback is by definition a permanent failure.

What to do in case of permanent failures: contact the appropriate people above, if you are sure you know what happened. Otherwise, e-mail the opsprob list and the appropriate on-call expert (see L1 shift schedule - [link](#)). If there is another part of the run waiting, the run lock (see below) will have to be removed by hand; e-mail unless you're really sure of what you're doing.

Other failures

This is a list of failures that don't really fit into the other major three categories

- Too few events in the run, or gaps - can lead to too few events in the magic7 file ([example log file of an FT2 failure](#)). Try to copy the /nfs/farm/g/glast/u28/stage/XXX/magic7_XXX.txt (where XXX is the delivery number) to /nfs/farm/g/glast/u28/stage/YYY/magic7_YYY.txt (where YYY is the delivery where fakeFT2 failed ... remember to change XXX to YYY on the magic7 file name). Then rollback fakeFT2. If this fails, email Giacomo Vianello ([need e-mail](#)) for additional information regarding problems with FT2.
 - See [opsprob discussion](#) of FT2 failure due to a very small chunk containing only 1 crumb.

Everything to know about the Rollback

A rollback is essentially a restart of the stream or substream. It will re-run a particular process and all processes that depend on its output.

How to rollback

You can roll back from the pipeline [front end](#). The entire stream can be rolled back by clicking "Rollback Stream" at the top, or individual streams in the main stream can be rolled back by selecting the pink boxes under "Stream Processes" and clicking "Rollback Selected".

But if multiple processes have failed (common), it's usually better to use the command line.

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD rollbackStream --minimum 'L1Proc[80819007]'
```

This will roll back all of the failed, terminated, or cancelled processes in delivery 80819007. If you don't say --minimum, it will roll back the whole delivery. That's usually not what you want. Also note that it will not rollback processes that have succeeded, but with incomplete information (ie - problems arising from afs/nfs hiccups). Such processes may need to be rolled back via the front end. (for a list of pipeline commands, ``/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline --help`` shows what is available)

After a rollback, the red x on the data processing page will be gone, but the L1 status will still say Failed. This tends to confuse the duty scientists. You might want to use the `setL1Status` task (see below) to make it say Running. This is really optional, it won't affect the processing in any way. But there will be fewer pagers beeping.

Removing "dontCleanUp" is not necessary to process the data. The file just stops temporary files from getting deleted when we're done with them.

Some notes about how to roll back the half pipe can be found at [Instructions for rolling back a halfPipe stream and subsequent L1 stream](#)

Rolling back substreams

From the [front end](#), find the "Substreams" area and click the pink boxes for substreams that you want to roll back. Then click "Rollback Selected SubStreams".

From the command line it's a bit more tricky:

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD rollbackStream --minimum 'L1Proc[90117001]/doRun[253889937]'
```

Remember to escape the square brackets if you are in tcsh.

As of 2011/04/19, the prod command-line client fails if you try to roll back substreams. But the dev client running in prod mode can.

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/dev/pipeline -m PROD ... (was this ever fixed? why or why not?)
```

When to rollback

Wait for the "setStatus" stream to have run.

Rollback won't work unless everything downstream of the failed process is in a final state. It's generally not harmful to try too soon, you just get an unhelpful error message. Most things at run level can be rolled back right away. If a recon job fails, you'll have to wait at least an hour. Maybe half a day. Basically wait until all downstream processes have finished. If you try to roll back too soon, the attempt will fail harmlessly. So don't be afraid to try.

Notice that the GRB search is executed per delivery and depends on all the FT1 and FT2 files in each run to be registered (therefore, it depends on almost "everything" that belongs to that delivery). For this reason, you might need to wait for the entire delivery to be completed before being able to roll back any failed recon jobs. And because of the run lock (see below), some of the (parts of) runs in the delivery might have to wait for other deliveries to finish, which might have their own failures... It's possible, but rare, to get deadlocks, where nothing can proceed until a lock is removed by hand. Best to ask for help then ([who do we ask? and if us, what do we do to fix it?](#)).

In general, experience will tell you when you can roll back what. So, in doubt, you can try anyways (if it's too soon, nothing will happen and you will get an error)!

Often you can roll things back sooner if you cancel some processes. If there is a delivery with some runs that are ready to roll back and others that aren't, you can do the rollback if you cancel `kludgeAsp`. ``/afs/slac.stanford.edu/u/gl/glast/pipeline-II/dev/pipeline -m PROD cancelProcessInstance 13388768`` The number (or numbers, you can use a space-separated list to do more than one at a time) is the oracle PK for the process instance, it's in the URL for the process instance page in the frontend. This takes a long time, 10-30 minutes. Check to make sure the cancel actually happened. This often takes so long that it times out and the cancel fails.

The dontCleanUp file

Any time one of the merges processes can't find all of its input files, a message is generated in the [Log Watcher](#) (and there will be errors in the log of the failed processes complaining about "Different number of events ...") and cleanup for the run is disabled by a file called `dontCleanUp` in the run directory on `u41/L1`. All cleanup jobs will fail if the `dontCleanUp` file is present. If everything is OK (see instructions below), that file can be removed and the jobs rolled back.

To check that everything is OK, follow these steps:

The new way:

- Look at the `dontCleanUp` file. It should contain a list of all merge processes that missed files, and what files were missing. It's possible that it will get garbled by multiple jobs writing to it at once, so if it doesn't seem to make sense, you can still get the information by following the steps below.

The old way:

- From the [glast-ground page](#), click "Logging" (about 1/3 of the way down left side)
- Click "Selection" (upper right) - page should look like [this](#)
- Click "More Selections" (lower right) - page should look like [this](#)
- In the "Select Program" field, enter "mergeStuff.py"
- In the "Select Target" field, enter the run number.
- Click "Submit Selection" - page should look like [this](#) after filling Program and Target fields and submitting
- Click "Table" (upper right, next to "Selection") - page should look like [this](#)

Now you should have a list of ([what??](#)) complaining about not being able to find input files. If any of the complaining merges are downstream of a failed process, you can ignore them. This is usually the case. But if they are not downstream from a failure, it means there was a disk read error while setting up the merge. Look at the logfile for that merge, it will say what files are missing. If they are really there, the merge can be rolled back and will probably work the second time. If they are not there, there's a bigger problem, and this should be considered a "permanent failure".

Any time one of these messages is generated, cleanup for the run is disabled by a file called dontCleanUp in the run directory on u41/L1. All cleanup jobs, and checkRun, will fail if that file is present. If everything is OK, that file can be removed and the jobs rolled back. The cleanupCrumbs jobs shouldn't fail if they're rolled back after cleanupChunks has run (they used to, but don't anymore).

The run lock

Only one delivery can process a run at a time. This is enforced by a lock file in the run directory on u41/L1. If findChunks fails or there are permanent failures in the run and another part of the run is waiting, it has to be removed by hand. It should never be removed unless the only failures in the run are findChunks or permanent ones, or there's a deadlock. Even then you have to wear a helmet and sign a waiver.

HalfPipe locks

The halfpipe uses several locks that can get left behind when jobs die and then cause trouble with retries and rollbacks and other deliveries.

Halfpipe.mergEvt has a run lock. The names are like /nfs/farm/g/glast/u42/ISOC-flight/Downlinks/lock/135d293d where the last bit is the run number in hex. If one of these jobs fails, it may leave this behind, then the retry or rollback will wait until it is removed by hand. But make sure that there actually isn't another instance running.

If cleanup dies, it leaves behind
/nfs/farm/g/glast/u42/ISOC-flight/Downlinks/lock/cleanup
and then all cleanup jobs just pend.

Similarly for launchOnline and
/nfs/farm/g/glast/u42/ISOC-flight/Downlinks/lock/launchOnline
and if
/nfs/farm/g/glast/u42/ISOC-flight/Downlinks/stage/.decode
is there, launchOnline will fail.

findChunks

This process is now automatically retried like most of the others. When it fails, it attempts to remove the run lock (see above) and the throttle lock (next section) by hand (and usually succeeds). But if it fails harder than usual, you might still have to do that by hand. Also, you'll probably have to:
mv /nfs/farm/g/glast/u41/L1/\${runId}/\${runId}\${deliveryId}_chunkList.txt /nfs/farm/g/glast/u41/L1/\${runId}/\${runId}\${deliveryId}_chunkList.txt.tmp

#WBF I think we should have a section on chunkLists. If anyone beats me to writing it, go ahead!

In addition, findChunks checks the chunks to make sure they don't overlap, go backward, or otherwise act crazy. That doesn't happen much anymore. Except when RePiping. See the section on that.

The throttle lock

There's a throttle that limits the number of RDLs that can be in process (or in the hard part, at least) at once. It works by making files with names like /nfs/farm/g/glast/u41/L1/throttle/1.lock at the same time as it makes the run lock. config.throttleLimit is normally set at 2, 3 is usually safe but not always. Leave it at 2 unless we're way behind and you're up for watching it closely.

You can tell from the contents of the file what the run & delivery were.

Suspending jobs

If the first part of a run is processing and you want to stop the second part from starting at a bad time, use the pipeline front end to get the LSF job ID of the findChunks process for the second part (which will be pending due to the run lock placed by the first part), log into a noric as glastraw and use bstop to suspend it. bresume it when you're ready to let it run.

If both parts of a run arrive while it's locked out, you can reduce the total amount of I/O that it does by letting the smaller part go first, since all of the data in the part that goes first has to be merged twice. Suspend findChunks for both parts, remove the run lock, then resume findChunks for the part with less data. "Less data" == "fewer chunks" unless it's highly fragmented, in that case du on the evt chunk directory (like /afs/slac.stanford.edu/g/glast/ground/PipelineStaging6/halfPipe/090512001/r0263753970) may give a better idea.

Skipping events

Sometimes we get an event that crashes digi. Usually with an error message like

```
Setting Missing Cal Readout Flag, 2 CalDigis are missing readouts for Event: 10641536 Apid: 956 Event Index: 20074
/afs/slac.stanford.edu/g/glast/ground/releases/volume14/GlastRelease-17-35-24-lp29/bin/redhat4-i686-32bit-gcc34-Optimized/Gleam: line 19: 10505
Segmentation fault (core dumped) $INST_DIR/exe/redhat4-i686-32bit-gcc34-Optimized/Gleam "$@"
```

Then we have to skip the event. To do this roll back the whole doChunk stream and set a variable EVENTSTOSKIP to a space-separated list of event indices. That's the offset into the chunk, NOT the GEMID. 20074 in the example above.

Skipping events when Gleam segfaults

Sometimes we get an event that crashes Gleam in recon.

It will leave a core dump, which is preserved. Instruction for skipping the event are:

- Fire up gdb on the core file (using Gleam as the executable)
- go up the stack 26 levels to RootloSvc::run ("up 26")
- print eventNumber
- add 1 to the number it gives you, and add that to the starting index of the crumb (which is the doCrumb stream number)
- roll back the doChunk stream, defining EVENTSTOSKIP=the number you just calculated

Re-trying FASTCopy ingestion or ProcessSCI

The instructions on how to retry FASTCopy ingestion or ProcessSCI can be found [here](#)

Useful resources

You should definitely join the following mailing lists:

- Data Handling (datalist)
- Level 1 processing (l1proclist)
- Data processing pipeline (pipelinelist)

And probably these:

- LAT Instrument Data Monitoring (datamonlist)
- Questions and problems concerning GLAST software (helpsoftlist)
- LAT operations team (latopslist)
- Nagios alert messages (nagios)
- People who will work ISOC shifts (shiftslist)
- ISOC Science Operations (solist)

Other things you may want to know

setL1Status task

Tired of being paged because L1Proc status still says Failed after a rollback?

```
/afs/slac/g/glast/ground/bin/pipeline --mode PROD createStream --define "runNumber=240831665,l1RunStatus=Running" setL1Status  
OR:
```

```
/afs/slac/g/glast/ground/bin/pipeline --mode PROD createStream --define "runNumber=240837713" setL1Status
```

l1RunStatus defaults to Running, but you can set it to any of the allowed values (Complete, InProgress, Incomplete, Running, Failed).

Advanced: in case a run needs to be set as Complete/Incomplete in the ACQSUMMARY table, see here: [HTF Run completion status](#).

setL1Quality task

This can be used when you need to preset a run to Bad (upcoming tests, etc.).

```
/afs/slac/g/glast/ground/bin/pipeline --mode PROD createStream --stream 240831665 --define "runNumber=240831665,newQuality=Bad" setL1Quality
```

How to contact SCCS

- Send an email to unix-admin (this usually works, even in the night and in the weekend).
- If you don't get an answer and the issue is urgent, call xHELP (650-926-4357). Choose 3 to page the on-call person.
- If your call/email isn't answered and the issue is REALLY urgent, page again the on-call person at 650-926-2230.

Log Watcher Messages

Message text: Can't open lockfile /nfs/farm/g/glast/u41/L1/r0248039911/r0248039911.lock.

* The monitoring shifter doesn't need to do anything about it. The L1 shifter should figure out why it happened. In this case, it was because I rolled back a merge job in a run that was already done. (Warren)

cancelProcessInstance

Here's the syntax to cancel a process that is not in final state, and all its dependencies (might be useful when you don't want to wait for it to finish before rolling something back. However, it's usually faster to wait as the cancel can take a long time):

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/dev/pipeline --mode PROD cancelProcessInstance 8073657
```

The Process Instance Primary Key (ie - 8073657 in the example) can be determined in the the URL for the process instance page (eg: the PIPK for <http://glast-ground.slac.stanford.edu/Pipeline-II/pi.jsp?pi=20253756> is 20253756).

NOTE: Please don't use this unless you really (REALLY!!!) know what you are doing.

Host or host group is not used by the queue.

This shouldn't happen anymore, we got rid of the host lists.

If you see jobs being terminated with exceptions in the message viewer saying things like "yili0148+5: Host or host group is not used by the queue. Job not submitted.", it means the hosts available to glastdataq have changed. The solution is to roll back the affected chunks (the doChunk streams) with a new value for HOSTLIST. When you roll back the streams from the frontend, on the confirmation page you are presented an opportunity to set or redefine variables. To figure out what the new value needs to be, do a "bqueues -l glastdataq". The output will include a line like "HOSTS: bbfarm". In this case you'd enter HOSTLIST=bbfarm in the box on the confirmation page. bbfarm is actually a temporary thing for the cooling outage, when things get switched back to normal, the relevant line from bqueues will probably look more like "HOSTS: glastyilis+3 glastcobs+2 preemptfarm+1". Then the thing to enter in the box would be HOSTLIST="glastyilis glastcobs genfarm".

Red question marks in display

Red question marks show up in the display for HalfPipe. Click on the FASTCopy "Logs" number. First check to see if the "ProcessSCI.end" event has occurred. If not, then wait until it occurs and see if the red question mark persists. If it persists and the log shows messages saying "pipeline nonEventReporting submission succeeded" or "no reassembled datagrams found", it just means that it contained data that had already been received. Meaningful deliveries will say "found XXX LSEP datagrams in". If it is a meaningful delivery, then further intervention is required.

Multiple lines for run in a given delivery

Occasionally, if deliveries arrive too close together, there may be a problem with the assignment of IDs. In this case, there may be several lines that appear for each run in a particular delivery, as shown [here](#). It is a display issue and not a problem with the processing itself, and Jim Panetta (panetta{at}slac) or Steve Tether (tether{at}slac) should be able to resolve the problem with the display (**Who should be doing this going forward? Also, documentation needed on how this is accomplished**). Also email OPSPROB (opsproblast{at}glast2{dot}stanford{dot}edu)

Stuck processes

If you suspect a process is stuck, visit the process page from the run page (ie - click on the green task bar in the Data Processing Page, then click on the particular job that is running in the list at the bottom of the page). Each entry in the table on the page will contain a link to "Messages", "Log", and "Files". If you click on "Log", it should show you the log, but more importantly, it'll show the location of the log file right above the box containing the log information (note: if the file is too big, it will not open, but the link to "Files" will show where the log is located, so you can get the information that way as well). If you visit the location of the file on the public machines, you can determine when the last time the file was updated (ls -l, or tail -f to see if it's still streaming). If it is not updating, find out the batch job number (JobId from the stream table), log into noric as glastraw, and issue a bkill <JobId> (bkill -r might be necessary if they're not going away after doing the bkill). The process should automatically be restarted after this command is issued.

I like to sometime do a sanity check by (on a noric)

```
date ; ls -lt /nfs/farm/g/glast/u41/L1/throttle/ ; bjobs -wu glastraw | head ; busers -w glast glastmc glastraw ; bqueues -w glastdataq
```

The throttle locks should be ~an hour or less old. If much older, maybe a stuck job or findChunks wasn't rolled back right.

So should the oldest batch jobs except: mergeReconChunks, usually 3-4 hours? could be up to a day; mergeCalChunks, mergeSvacChunks, fastMonHist, could be ~2 hours. If much older, they might be stuck.

If you suspect a machine to be hung (several processes stuck on the same machine), then also email unix-admin{at}slac about the machine, along with bkill-ing the processes. You can also remove it from LSF (see below).

Repiping

When weird things are happening with the delivery that you can't figure out or it has too many chunks to process, it may be necessary to repipe the stream (probably a good idea to check with someone first - **Who?**). Log in as glastraw and issue the following command:

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD createStream --stream ${ID} --define downlinkID=${ID},runID=${ID} RePipe
```

where \${ID} is the Run ID (just numbers, no leading r0). Set a shell or environment variable, it will be easier to type and you'll use it again in the next step. Why are you setting the stream, downlink, and run all to the same thing? You don't have to. The run has to be right, the stream and downlink could be anything. But it will be easier to check progress of RePipe and L1 in the frontend if they're all the same.

If you're RePiping an LCI run, you'll have to set the chunkApid variable when creating the stream. For run 270108008 it was 965, I'm not sure if there are others.

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD createStream --stream ${ID} --define downlinkID=${ID},runID=${ID},chunkApid=965 RePipe
```

This will create a directory in /nfs/farm/g/glast/u28/RePipe ... which should look like a normal delivery, plus some harmless junk. It will contain the entire run, you only have to do this once no matter how many deliveries the run was split between.

Wait for RePipe to finish. You can get to the current version here: <http://glast-ground.slac.stanford.edu/Pipeline-II/?taskFilter=RePipe®Exp=on&include=all&versionGroup=latestVersions&submit=Filter>

You will need to then manually re-enter the run into L1Proc. To do this, bkill any findChunk processes that are associated with the RunID, remove the run lock from /nfs/farm/g/glast/u41/rXXX (where XXX is the run number), also move all *chunkList*.txt (leave the .txt.tmp ones alone) files to something else (just suffixing them with ".ignore" should work)

#WBF I think we should have a section on chunkLists. If anyone beats me to writing it, go ahead!

and issue the following command:

```
/afs/slac.stanford.edu/u/gl/glast/pipeline-II/prod/pipeline -m PROD createStream --stream ${ID} --define DOWNLINK_ID=${ID},DOWNLINK_RAWDIR=/nfs/farm/g/glast/u28/RePipe/${ID} L1Proc
```

The repiped run(s) will create run and throttle locks just like any other, so you don't need to worry about what else is going on in L1 when you inject them. Except there shouldn't be anything else for this run running or waiting, or you'll have trouble renaming the chunklists as described above. But they won't show up on the data processing page, and the regular runs that do show up there will appear to be slow starting for no obvious reason, so you might want to put a note in opslog ([is the opslog still being used? where is it?](#)) to avoid confusing the monitoring shifter.

You should be able to see if the process restarted correctly by visiting [here](#), (Updated to point to L1 v5.7). Click on "Last" to get to the end of the list.

NOTE: If repiping a process that has completed, then it may be necessary to rename the chunkList files for that run before issuing L1Proc command (in /nfs/farm/g/glast/u41/L1/runs/YYYY/r0XXX ... where YYYY are the first three numbers to XXX).

How To Fix common FASTCopy problems

From Steve Tether: I have a prototype script that corrects a test problem I created in the NIGHTLY database. If you see an ingestion failure for one or more Level zero files of a delivery first take a look at the FASTCopy logs available through the Data Processing web app. (Click on the progress bar for FASTCopy then on the links for those files under Input Products that show a status of INGESTFAIL). If it looks like an NFS glitch, i.e., a message says such-and-such a directory or file does not exist, run the following command in a terminal window that is logged into the SLAC AFS cell and has the ISOC PROD environment set up ([where are the instructions for setting up that environment?](#)):

```
python ~tether/retryIngestion.py delivery-id-number
```

For example

```
python ~tether/retryIngestion.py 100308011
```

The script will check whether the files for that delivery are still on disk. If any are not or if the script reports a failure then you'll have to refer the problem to me ([will be Don? Need procedures for that person](#)). Normally the files received for a given day (UTC) are archived and removed from disk at about noon (Pacific time) the next day. If all goes well then the last output from the script will be a listing of the L0 files whose statuses were reset to NEW. After that the L0 file states should go through SUBMITTED and stop at INGESTDONE. If ingestion fails again then Jim or I will have to handle it.

Batch commands

bjobs -wu glastraw - running this from noric will list all the jobs in the LSF batch queue owned by glastraw (which is the user submitting pipeline streams).

To close a specific machine from LSF: sudo badmin hclose -C "test" dole0022 (from a machine where you have sudo privileges, i.e. not a noric).

To reopen a specific machine to LSF: sudo badmin hopen -C "test" dole0022

Here's a little script that is useful for monitoring:

```
#!/bin/bash
lines=$1
export LSB_BJOBS_FORMAT="id name:15 user submit_time stat:5 exec_host:10 start_time mem cpu_used"
date
ls -lt /nfs/farm/g/glast/u41/L1/throttle
bjobs -u all -q glastdataq | egrep -v '(sleep|fitEnergyBand|sourceAnalysis)' | head $lines
bjobs -p -u all -q glastdataq | head
busers -w glast glastmc glastraw
bqueues -w glastdataq
```

Shift sign up

Sign up for shifts [here](#). View shift calendar [here](#)

Useful L1Proc links (these are designed to keep you from waiting for the entire L1Proc history to be retrieved ... which could be 10 min and up).

- [All L1Proc Deliveries that have not completed](#)
- [All L1Proc Runs that have not completed](#)
- [All L1Proc Runs that have failed with a timestamp](#) (it is a good idea to update your timestamp to shorten the search).