Skimmer FAQ

Warning

The tool has been generalized and externalized. The last releases of source code, documentation and pending issues are now available at the new TRAC server. Before looking below, you should first consult the new FAQ. What remains below are old questions, still valuable for some old skimmer releases, or questions which are specific for Fermi.

How to use the skimmer to merge files ?

The skimmer has been given recently the ability to merge similar files. So to do it with a skimmer release prior to v6r0, one needs to do the following :

1. ensure SK_TCUT is undefined or empty string.
2. ensure SK_EVENT_LIST_FILE refers to a file which do no exists.
3. set SK_SKIP_GET_EVENT_LIST to true.

With recent releases :

1. ensure SK_TCUT is undefined or empty string.
2. ensure SK_INPUT_EVENT_LIST is undefined or empty string.

When there is no cut and no event list, the skimmer understands you want to merge, and perform it with the function TChain::Merge(...). Only if the maximum file size is set to 0 (which means no maximum), the skimmer will be able to go faster thanks to TChain::Merge(...,"fast").

I see a warning about right access problems with /tmp/Skimmer. Should I worry ?

In releases v6r1 and before, the skimmer relies on ROOT ACLiC for some tasks, and ACLiC is used to work in /tmp. If several users are working on the same machine, this could result in access right conflicts. This does not hurt when one work with tuples, but could be fatal to the jobs which needs some data definition libraries (recon/digi/mc/gcr). So to instruct ROOT ACLiC to work somewhere where the users has Unix right access, try to set the variables TEMP and/or TMPDIR to a value such as "$HOME/tmp". This should help.

How the skimmer is trying to guess the code release for a given task ?

So to find the code release for a given task (the GET_RELEASE step), the skimmer browses various input data files, tries to find some FileHeader instance, search within each header what was the main package and release, and finally check if it can find the corresponding library libcommonRootData.so .

Input data files taken into account

The skimmer tries to scan the first file of each data kind, and looks for some instance of FileHeader.

NOTE: Before trying to get some instance of FileHeader, the skimmer compiles on the fly (thanks to ACLIC) a FileHeader class provided with the skimmer. The FileHeader class is expected to be very stable, and we hope the one we provide with the skimmer is valid for all tasks. We have prefered this solution, rather than loading an arbitrary release of libcommonRootData.so, as was done before. If the FileHeader class change in the future GLAST projects, we will have to make our implementation more flexible, and the choice of FileHeader class depending on the task.

FileHeader interpretation

The skimmer search for some parameter called "CmtPackages". Then, it is looking for some release package called "GlastRelease" or "BeamtestRelease" or "EngineeringModel". Then it is extracting the corresponding release value, and the corresponding directory path.

If no release can be found out this way, the skimmer use SK_EXPECTED_RELEASE in further steps. SK_EXPECTED_RELEASE has a specific format one should conform to. For example, for GlastRelease v9r3, SK_EXPECTED_RELEASE should be given the value "GlastRelease/GlastRelease-v9r3".

Search for libraries

Once the release is established, the skimmer tries to find the corresponding libcommonRootData.so . It first check within the directory whose path is given in the FileHeader. If there is no more such library in the file system, it is trying under each element of SK_LIBRARY_DIRS. For each <top_dir> element, a given <main_package> and a given <release>, it looking for <top_dir>/<main_package>/<main_package>-<release>/lib/libcommonRootData.so.
For example, with the default value of SK_LIBRARY_DIRS, one the file which will be looked at is "/nfs/farm/g/glast/u09/builds/rh9_gcc32/GlastRelease/GlastRelease-v9r3/lib/libcommonRootData.so" .
Once libcommonRootData.so is found in some directory, this directory is registered as the one
where all the data definition libraries should be found.

How to install and run the skimmer outside of SLAC ?

For people who like risky life.

If you want to skim a kind of data which is not tuple-like, remember the skimmer will need to load the corresponding libraries (such as libmcRootData.so for mc data). So you must ensure those libraries are reachable from the machine where the skimmer will be running, for all the releases which were used when generating the data to be skimmed. If you recompile those libraries locally or copy them from SLAC, you must mimic the directory structure used at SLAC (as expected by the skimmer). Especially, for a given <main_package> and a given <release> (for example GlastRelease and v9r3) the path of the directory containing the libraries should end with "/<main_package>/<main_package>-<release>/lib/" (for the example : /GlastRelease/GlastRelease-v9r3 /lib/). The top directories of those libraries should be declared in shell variable SK_LIBRARY_DIRS.