Skimmer at Slac

> **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. The specific issues and features for FERMI are in the documentation guide called TSkim for FERMI, which should be more complete than information below.

Use of the skimmer on Noric machines

The skimmer is only usable on linux, and most of its features will not work outside of SLAC network, because it needs the access to intput ROOT data files, and to the metadata database for the pipeline I data files (at least for the step which is preparing the list of input ROOT data files). This is why it is recommended to use the skimmer only on noric machines, where any new release is installed by its developers.

Currently, the user must know which ROOT release was used for the data he wants to skim, and should define $ROOTSYS accordingly before running the skimmer. This ROOT release must be chosen within a subset which is defined by the administrator when installing the skimmer. Currently, for the noric cluster, the valid values for ROOTSYS are :

```
/afs/slac/g/glast/ground/GLAST_EXT/$CMTCONFIG/5.10.00/root
/afs/slac/g/glast/ground/GLAST_EXT/$CMTCONFIG/5.14.00g/root
/afs/slac/g/glast/ground/GLAST_EXT/$CMTCONFIG/5.16.00-gl1/root
/afs/slac/g/glast/ground/GLAST_EXT/$CMTCONFIG/5.18.00c-gl1/root
```

For simple kinds of data, such as merit tuples, using a different recent ROOT release should be ok.

If $ROOTSYS is not defined, the skimmer will search for $GLAST_EXT/ROOT/v5.14.00g/root. If $GLAST_EXT is not defined, it will be set to /afs/slac/g/glast/ground/GLAST_EXT/$CMTCONFIG. If $CMTCONFIG is not defined, it will be set to rh9_gcc32.

Thus, once $ROOTSYS is eventually defined, together with the relevant SK_* shell variables (as described in the user guide), one can simply type :

```
/afs/slac/g/glast/ground/DataServer/v6r1/bin/skimmer
```

Of course, it is more confortable to define an alias for it in your Unix account setup files, or add the DataServer bin directory to the Unix PATH :

```
sh> export PATH=<SKIMMER_DIR>/<release>/bin/skimmer:${PATH}
csh> setenv PATH <SKIMMER_DIR>/<release>/bin/skimmer:${PATH}
```

For what concerns the external tools, at runtime, the skimmer depends on :

1. Perl 5, which should be found with "/usr/bin/env perl".
2. One of the supported ROOT releases (see above), specified wirh $ROOTSYS.

The predefined data kinds

The kinds of trees which can be skimmed is currently hardcoded, and especially targetted to GLAST needs. For each of those tree, we expect a top main branch. Actually, we distinguish two sets of trees : the ones which only rely on predefined ROOT types and have a single level of branches under the main one (the tuple-like trees), and the others. The latter typically requires the loading of a dedicated data description library before the concrete skimming takes place. The last release v7r0 should understand the following types :

- **merit** (tuple-like)
- **pointing** (tuple-like)
- **jobinfo** (tuple-like)
- **cal** (tuple-like)
- **svac** (tuple-like)
- **mc** (requires libmcRootData.so)
- **digi** (requires libdigiRootData.so)
- **recon** (requires libreconRootData.so)
- **gcr** (requires libgcrSelectRootData.so)

Worth to be noted, some different kind of trees can be grouped in common files, such as `merit`, `pointing_history` and `jobinfo`. When skimming, the result is split among several files, one for each kind of tree. Also, some kind of trees do not have branches for the run and the event ids, such as `pointing_history` and `jobinfo`. Those trees are never skimmed, rather always merged.