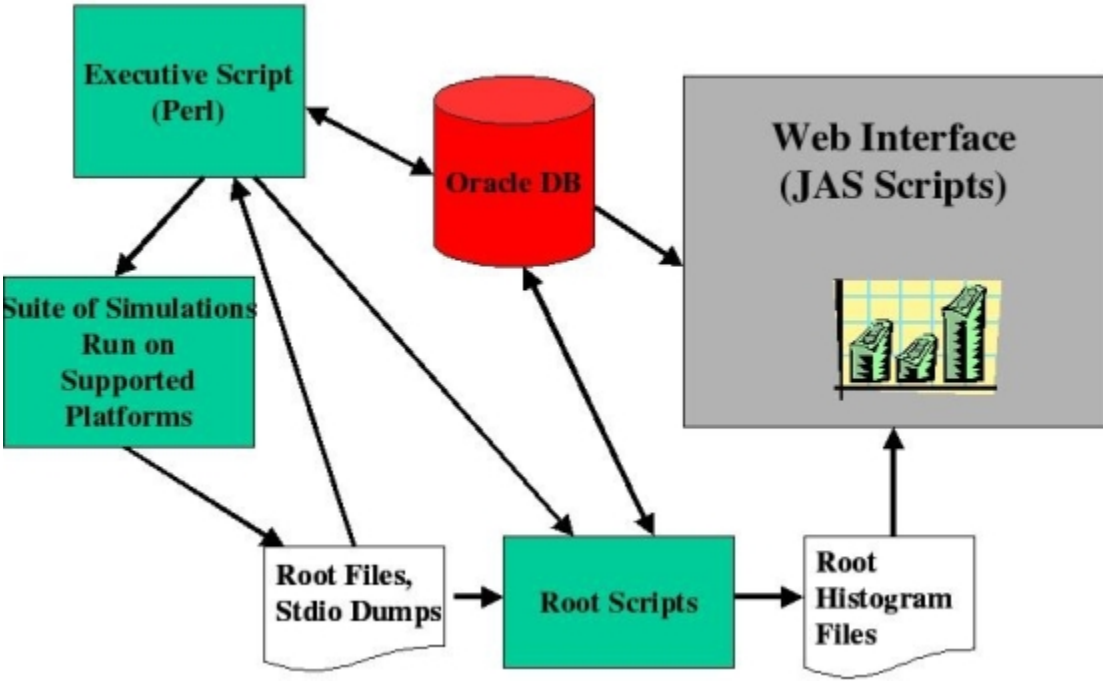
System test overview and instructions

Overview

This is a description of the back end of the system tests. They consist of a perl module, several perl scripts and some root macros. The bulk of the original development was by Karl Young. You can find a copy of the documentation he wrote attached to this page (you'll notice that these pages bear some similarities to his work!)

The purpose of the GLAST SAS system tests is to provide and end to end test of the offline software. They are typically run for each tagged release of the GlastRelease, EngineeringModel and BeamtestRelease packages.

The components are outlined in the following diagram.



The text below describes how to run the system tests manually. In principle, the Release Manager runs these tests automatically after a new release is generated, comparing that release to the previous one. For reasons not yet understood (by Julie and Navid), this job consistently fails to copy the files from the working directory to the archive directory. (In fact, typically the first step in running the system tests in manual mode is to cancel all the jobs which have been automatically submitted for that package.) When this problem is fixed much of the following will be unecessary.

The code for the system tests currently resides in */nfs/farm/g/glast/u17/systests/src*. (TBD: commit the current working version of the code to CVS!)

The system test area, *$SYSTESTS*, is owned by *glastsys* and the tests are run via this account (most necessary environment variables are set up in the scripts. The password for this account is kept in escrow and should be kept up-to-date there.

NB: if you have logged in as *glastsys*, which you need to do to get write-access, the envvar *$SYSTESTS* is set to */nfs/farm/g/glast/u17/systests*. Also, you need to login to a SLAC *linux* machine; practically speaking, this means a *noric*. This is because the output directory is named with the operating system of the machine that submits the batch jobs, and the batch job looks for the name if *its* operating system. (This probably isn't necessary, since that directory contains platform-independent files.)

The main executive script is *runSysTests.pl* in the the *exec* subdirectory, and is supported by utility routines in *SysTests.pm*. It runs the suite of shell scripts that it finds in the *EMtests, GRtests and BTtests* subdirectories of *src/*, tracks the results and updates Oracle database tables. Each of the shell scripts that it runs consists of at least two parts: it first runs either a Gleam or LatIntegration job, and then runs a RootAnalysis macro, which reads the output files produced in the Gleam step and fills a collection of histograms. Once the histograms have been created, the executive script sets the standard and threshold for each histogram, that is, the version that this release is to be compared against, and the statistical threshold that defines whether the test has passed or failed. Finally the executive script runs a root macro which calculates several quantities from each histogram (number or entries, rms, KS probability that the histogram differs from its standard etc) and updates the database.

The EMtests, GRtests and BTtests directories contain one directory for each test type used for the EngineeringModel, GlastRelease and BeamtestRelease packages respectively. The exec directory contains the perl scripts that run the tests. The output of the tests (root files and run log and summary) ends up in subdirectories of *GlastRelease, EngMod* or *BeamtestRelease.*

Running the system tests

The system tests are configured using a configuration file. An annotated example can be found in */nfs/farm/g/glast/u17/systests/src/exec/Config_example.txt*.

Glossary

- **package name**: The software package to be tested, e.g., GlastRelease.
- **package directory**: Each package has a setup directory containing test files, e.g., $SYSTESTS/src/GRtests, and a separate result directory, e.g., $SYSTESTS/GlastRelease.
- **version name**: A name for the version of the the package to be tested, e.g., v17r25p20 or v17r35p20_geom. Note that this name usually matches the release version, but can include additional information that indicates details of how the system test was run.
- **version result directory**: Each version of a package that is tested has a separate result directory in the package result directory, e.g., $SYSTESTS/GlastRelease/v17r35p20.
- **test name**: Several tests are run for each version of a package, e.g., AllGamma, VerticalProton1GeV. These typically specify different MC inputs to Gleam.
- **test directory**: Each test has a setup directory in the package setup directory. New tests can be added by simply creating a directory and placing the appropriate files and inputs in the package directory. The tests have result directories in the version result directory, e.g., $SYSTESTS /GlastRelease/v17r35p20/AllGamma or $SYSTESTS/GlastRelease/v17r35p20/VerticalProton1GeV
- **config file**: Common adjustments to how the system tests are run can usually be made in a configuration file. Typically, this is placed in the version result directory and called Config.txt. Other locations and names may be used if specified when running the test script.

Running all parts of the system tests

Instructions for running the system tests from scratch, i.e., run Gleam to create the full root trees, RootAnalysis to create the histograms, set the standard version and threshold for comparison, calculate the metadata, and run the comparison:

For this example, we setup a test for GR v17r35p20. The package is GlastRelease, the version name we choose is v17r35p20, and we are running all the tests, AllGamma, BackGndMixDC2, VerticalGamma100MeV, VerticalProton1GeV, etc. We want the default comparison to be to version name, v15r47p12gr11.

System Test Setup

Log into a machine with access to the batch farm.
> ssh noric.slac.stanford.edu
The glastsys password is stored in escrow.
> su glastsys
Create the version result directory in the package result directory. If space on the system test disk, u17, is slim, delete the mc.root, digi.root, recon.root, and result.root files for old tests.
> cd $SYSTESTS/GlastRelease
> mkdir $SYSTESTS/
Grab a recent config file from another version directory and edit for this run. See the example below.
> cp ../v17r35p14/Config.txt ./

Config File Example

```
packag GlastRelease
pversion v17r35p20
sversion v15r47p12gr11
ksthresh 0.5
builddir /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20
cmtpath /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20
cmtconfig rhel4_gcc34opt
executable /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20/Gleam/*/rhel4_gcc34opt/Gleam.exe
systests /nfs/farm/g/glast/u17/systests/src/GRtests
output /nfs/farm/g/glast/u17/systests/
glastext /afs/slac/g/glast/ground/GLAST_EXT/rhel4_gcc34opt
```

Now run the tests. These will be submitted to the batch queue.

Running the System Tests

```
> $SYSTESTS/src/exec/runSysTests.sh
Optionally, you can specify a different configuration file name.
> $SYSTESTS/src/exec/runSysTests.sh Config_somethingelse.txt
Note that rerunning the tests in the same directory with a different configuration file will overwrite the existing outputs and log files.
```

You will see lots of console output, but the jobs are submitted to the batch queue and do not require you to stay logged in to the terminal. You can come back and find top level errors in the version result directory in testerr. The output files and logs will be in the test result directories, e.g., $SYSTESTS /GlastRelease/v17r35p20/<test_name>/linux/

You can check on the runs using bjobs and looking at the output files. If there is a problem with the tests or you want to end the run early, you can kill the jobs with bkill 0. It is fine to do this and start them again. A run will completely overwrite a previous run with the same version name unless you set special options in the config file.

To examine the output of the job see http://glast-ground.slac.stanford.edu/SystemTests/summary.jsp. You will see completed jobs appear there, but no histograms or comparison results until you do the next step.

Once all the runs have finished, you will need to load the histograms and results to the database for web viewing.

Loading the histograms and comparison results to the database

```
> cd $SYSTESTS/src/exec
> source setup.csh
> python insert_metadata_version_sequence.py -b <version_name>
```

Note that the above step can be run multiple times without penalty before all the tests have completed to check on shorter test results as they become available.

More Details

In slightly more detail, *runSysTests.sh* sets up the run environment and calls *runSysTests.pl*. This launches a batch job, which runs a series of shell scripts, e.g., $SYSTESTS/src/GRtests/AllGamma/systest_AllGamma. This job runs Gleam using the *jobOptions.txt* file in the test setup directory, and then runs the *RUN_linux* root macro in the test setup directory to generate the histograms. Throughout this process, the Oracle database is updated as the tests progress (except for loading the histograms and comparison results due to changes in db access - hopefully fixed soon).

Only one set of system tests for each package can run at the same time. You can run a set of system tests for EngModel and GlastRelease simultaneously, but you cannot run two sets for different GlastRelease versions simultaneously because the tests will try to store the output files in the same working directory. (This can be fixed, but needs more work to separate out some directory structure assumptions properly.)

Running the System Tests for SCons builds

The only things that change for SCons builds (on the surface, anyway) are the configuration file and the script used to call the tests. Essentially, the environment is set in the configuration file and propagated through the test scripts as needed.

Other optional modifications to the configuration file described here can be used as normal.

> $SYSTESTS/src/exec/runSysTests.sh
becomes
> $SYSTESTS/src/exec/runSysTests_SCons.sh

The SCons version of the config file looks like this:

Config File Example for SCons

```
packag GlastRelease
pversion v17r35p20
sversion v15r47p12gr11
ksthresh 0.5
bldtype redhat4-i686-32bit-gcc34
instdir /nfs/farm/g/glast/u52/ReleaseManagerBuild/redhat4-i686-32bit-gcc34/Optimized/GlastRelease/17-35-24-gr17
systests /nfs/farm/g/glast/u17/systests/src/GRtests
output /nfs/farm/g/glast/u17/systests/
glastext /afs/slac/g/glast/ground/GLAST_EXT/rhel4_gcc34opt
```

While cmt and scons builds exist in parallel, versions names for tests of scons builds will end in *-scons*.

Tips and tricks for the savvy tester

Skipping the Gleam step (MC generation, digi, recon, and merit)

If the mc, digi, recon, and merit trees already exist and are populated and you wish to simply regenerate the system test histograms, then adjust the config file to select a root only run.

Root Only Example

```
> cd $SYSTESTS/GlastRelease/<version_name>
> cp Config.txt COnfig_rootonly.txt
Edit Config_rootonly.txt adding this as a separate line.
debugflag 1
Run the tests using the modified config file.
> $SYSTESTS/src/exec/runRootOnly.sh Config_rootonly.txt
Reload the results to the database if needed. (This is currently necessary - Aug. 22, 2011)
> cd $SYSTESTS/src/exec
> source setup.csh
> python insert_metadata_version_sequence.py -b <version_name>
```

This will access the database to find the locations of the root files that were already produced by Gleam and will regenerate the system test histograms. This is very convenient for testing and updating the RootAnalysis macros.

Re-running the comparisons/setting the default reference version

Once the gleam and root jobs have run, and the output files and system test histograms have been created, the comparison can be re-run quickly. In fact, new comparisons between releases or other special runs can be done in the same way. Note that **you can check the statistical comparison and overlay plots for any existing version anytime without doing this** by using selection boxes on the statistics page or plots page of the system test web interface. This step simply sets the default, i.e. the one that shows up in the summary with the plot failure counts and comes up automatically for a version.

Histogram Comparison Example

```
> cd $SYSTESTS/src/exec
> source setup.csh
First, set the standard (reference) test to be used for the default comparison.
> runSetStandard.sh <version_name> <standard_version_name>
Then, run the comparison and load the results to the database.
> python insert_metadata_version_sequence.py -b <version_name>
```

Running a subset of the tests

This is incredibly useful for debugging or rerunning tests that encountered an issue.

Run the tests by the normal method, but use a modified configuration file.

Config File Example for a subset of the tests

```
packag GlastRelease
pversion v17r35p20
sversion v15r47p12gr11
ksthresh 0.5
builddir /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20
cmtpath /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20
cmtconfig rhel4_gcc34opt
executable /nfs/farm/g/glast/u30/builds/rhel4_gcc34opt/GlastRelease/GlastRelease-v17r35p20/Gleam/*/rhel4_gcc34opt/Gleam.exe
systests /nfs/farm/g/glast/u17/systests/src/GRtests
output /nfs/farm/g/glast/u17/systests/
glastext /afs/slac/g/glast/ground/GLAST_EXT/rhel4_gcc34opt
subset AllGamma, VerticalGamma100MeV, VerticalGamma100GeV
```

Note the additional and optional last line. If the config file includes the subset keyword, then the list of comma-delimited tests following will be run without the others.

If I call this *Config_subset.txt* then I just run the tests giving that file as the argument, *../../src/exec/runSysTests.sh Config_subset.txt*.

If I do this in an existing test result directory, then existing files for the selected tests will be overwritten.

Deleting a version

You probably do not need to do this.

You can always delete an outdated/uninteresting or garbage version of the system tests by simply removing the version result directory. The version will still be listed in the database, but links to the files will be broken.
There is no pressing need to actually do this, but if you do want to eliminate a version of the tests from the database as well, then this might do the trick.

Note that you do not want to do this for any version that you want to keep the histograms around for reference. If the version to be deleted from the db has been used as a reference version for other versions, this will make the java interface unhappy for those versions. Deletion form the database has generally only been used when a run was particularly ill-contrived, on throwaway test/debug runs, or if the package version itself was a disaster for some reason.

Deleting a version of the tests from the database
> cd $SYSTESTS/src/exec > source setup.csh > runDeleteVersion.sh <version_name>

Setting JobOptions for L1proc or Pass8

The JobOptions parameters for the current v17 version of L1proc are slightly different than the settings for Pass8. Some of which is due to the different G4 versions in use. To switch back and forth, before running system tests, you'll want to "source setMode*.csh" for the appropriate version you are planning to run. There are also two versions of the ROOT scripts for L1proc and P8, TupleFragment.cxx and ReconTkrFragment.cxx. This script handles switching these as well.
The relevant files are in systests/src/exec:

setModeLevel1.csh, setModePass8.csh, which both call setModeVersion.csh.

Adding new Histograms to the scripts

Sometimes it is necessary to add or modify histograms to the existing ROOT scripts when new analyses are requested. The copy of the source to modify is located in $SYSTESTS/src/root. Note that some of the scripts have both a L1 and P8 version associated with them.

When running system tests, with new histograms for the first time, you could use the initial run as the reference version to itself - otherwise when running the comparison, there will be errors about missing variables if a previous run is used as reference.

System Test Reports