Internal Externals

Proposal

For Pass8 GlastRelease will need access to all the irfs packages. Rather than adding these packages in the usual way, the proposal has been made to put them in their own external and put all the packages they depend on (facilities, astro, etc.) in another external. ST and GR would then link to both of these externals. GRBAnalysis could also.

Alternative #1

Put all of the above in a single external. ST, GR and GRBAnalysis all need all of these things.

Pro: would simplify bookkeeping somewhat

Con: irfs could be much more volatile than the other packages

Alternative #2

Put all of the above plus other common packages (celestialSources, flux, xmlBase) into a single new external.

Pro: as in #1. Additionally, would reduce size of GR and ST further which could help somewhat with CVS lock problems

Con: Same as #1. Also GRBAnalysis doesn't use the extra packages (but there is no requirement it link to them all; we could define library groups as we do for ROOT)

Organization in CVS repository

For each new internal external, make a new container with sym links to the packages to be included.

For each existing container which will link to one or more new internal externals (GlastRelease, ScienceTools and maybe GRBAnalysis), make a brand-new container with sym links to packages as before, but excluding those in the internal externals.

Candidate Packages for Internal Externals

See table below indicating dependencies among packages (or package collections like irfs) and on "real" externals and some indication of volatility.

Name	Pkg dependencies	External Dependencies	Recent tags	Used by
facilities	(none)	Swig (build time only)	Oct 2012, Aug. 2012	GR, ST, GRBAnalysis
tip	facilities	ROOT, cfitsio	Nov 2012, Aug 2012	GR, ST, GRBAnalysis
astro	facilities, tip	CLHEP, cfitsio	Jan 2013; Nov. 2012, Oct. 2012	GR, ST, GRBAnalysis
st_stream	(none)	(none)	July 2009 (!)	GR, ST, GRBAnalysis
st_facilities	astro	cfitsio, f2c, cppunit	April 2013, Dec. 2012, Nov 2012..	GR, ST, GRBAnalysis
embed_python	(none)	(none)	April 2013, Aug 2012	GR, ST, GRBAnalysis
irfs*	astro,tip, st_facilities	CLHEP, f2c, ROOT	April 2013, Jan 2013, Dec 2012, Nov 2012	GR, ST, GRBAnalysis
xmlBase	facilities	xerces	Aug 2012	GR, ST
flux	xmlBase, astro	CLHEP, cfitsio	Aug 2012, May 2012	GR, ST
celestialSources	faciliites, flux, astro, xmlBase	ROOT, CLHEP, cfitsio	March 2013, Nov 2012, Aug 2012	GR, ST

irfs will be pared down slightly for the new external in order to reduce the number of other packages required. irfs/irfLoader and irfs/handoff_response will be modified to build only the library. Or irfs/handoff_response could be eliminated entirely since it's not needed by GR.

Sandbox Experience

To start I'm going with alternative #1: single internal external including irfs and minimal other packages to make it self-contained: facilities, tip, astro, st_stream, st_facilities and embed_python. The new internal is provisionally called "CommonExt". After making changes to irfs/irfLoader and irfs/handoff-response as described above I built it with SCons, using option --variant=NONE to produce a build suitable for export.

A Snag

The facilities package does some special things at build time which impact the entire container. It creates a file config.h in the facilities/src directory and the file src/commonUtilities.cxx has a #include for this file. The contents of config.h is a macro defining the set of packages in the container, something commonUtilities needs to know in order to properly define environment variables at run-time so that, for example, job options files and xml files may be found. Each container must have its own compilation of commonUtilities.

Partial Solution

In the new external CommonExt: make a new package, provisionally called commonExtSetup. Move the commonUtilities class out of facilities into the new package. Code in commonUtiliiites stays the same except namespace will be changed to commonExtSetup and internal references to package or namespace need to be changed. The new package will build a library libcommonExtSetup.so (or commonExtSetup.dll on Windows). All references to facilities::commonUtilities in other packages in CommonExt will be changed to commonExtSetup::commonUtilities. All programs belonging to CommonExt will link against the new library and will therefore be able to define environment variables such as XXXDATAPATH for all packages XXX belonging to CommonExt by calling commonExtSetup::commonUtilities::setupEnvironment( ).

Outside CommonExt: make a new package, provisionally called runtimeSetup. Copy the commonUtilities class in here as well, but keep namespace name = facilities. It will build a library libruntimeSetup.so (runtimeSetup.dll). Change linking of any packages in ST and GR using commonUtilities to include the new library (linking has to be changed anyway for most packages in ST and GR; this won't add significantly to the work involved). All programs belonging to ST (respectively GR) will be able to define environment variables such as XXXDATAPATH for all packages XXX belonging to ST (resp. GR) -- but *not* for packages in CommonExt -- by calling facilities::commonUtilities::setupEnvironment( ).

Remaining Issues

1. For the most part environment variables pointing inside CommonExt will not be of interest to ST and GR but there are exceptions: environment variables CALDB, CALDBCONFIG and CALDBALIAS having to do with the caldb subpackage of irfs. Possible solutions include
    a. Programs in ST or GR needing these variables can call commonExtSetup::commonUtiities::setupEnvironment( ) as well as facilities::commonUtilites::setupEnvironment. That might just work as is.
    b. If not (could be, for example, that this only works if you're running with CommonExt where it was built, not if you're running with a copy) add a new routine to commonExtSetup::commonUtilities which defines the CALDB* variables properly for use outside CommonExt
    c. Enhance SCons handling of externals somehow to export these things
2. The class st_facilities::Environment was written to simplify access to commonUtilities::setupEnvironment. However, since st_facilities will be in CommonExt, it only has access to the internal (namespace commonExtSetup) version of this routine. Perhaps the same gambit used for the commonUtilities class will work here. That is, move the st_facilities::Environment class out of the st_facililties package, into some other package outside of CommonExt. We probably could use the same new package: runtimeSetup.
3. commonUtilities.cxx has several sections starting #ifdef HEADAS .. because they organize builds somewhat differently. I believe the copy of commonUtiliites outside of CommonExt can be left as is; I'm not sure about the one inside. It will depend on whether FSSC chooses to build CommonExt with an organization identical to the one we use or not.