FluxSvc Software Code Review Report
July 16, 2002

Reviewers: Xin Chen, Marco Frailis, Heather Kelly, Karl Young

Attendees: Sandhia Bansal, Joanne Bogart, Toby Burnett, Jim Chiang, Seth Digel, Richard Dubois, Sean Robinson

Introduction

The FluxSvc software review was the fourth in our continuing series of code reviews. We would like to thank Sean Robinson who clearly spent much time to prepare for this review. This may be one of the least understood packages in our collection. The review provided a necessary tutorial on the flux code.

We took a look at both the documentation and the code from selected files in FluxSvc v5r9. This is a report on the FluxSvc package as it stood on July 16, 2002. It should be noted that some of the following recommendations are in the process of implementation.

Documentation

General Comments

In general, none of the standard Documentation Task Force documentation recommendations have yet been implemented in the FluxSvc code. This includes the setup of the mainpage, standard class comment headers for each class declaration, method comments, and source header comment blocks. FluxSvc developers are urged to review the Code Documentation recommendations available at:
http://www-glast.slac.stanford.edu/software/core/documentation/doc/codeRecommendations.pdf

In particular, the mainpage needs more content. In fact, much of the detail is provided in an HTML document stored with the package in the doc directory. Some of this detail should be stored in the mainpage instead, where we can take advantage of Doxygen's features, including the automatic linking of class names to their Doxygen documentation.

In addition to details, an explanation of how the classes in the FluxSvc work together is necessary. A general overview would make the contents of this package much easier to digest. In particular, the point that there is a defined interface for Gaudi and a separate section of code that is completely Gaudi independent should be made more clearly. Many review attendees were confused about how the pieces worked together and what purposes they served.

The jobOptions parameters should be documented, as specified in the Code Documentation recommendations. Here is an example:

```
 @section jobOptions jobOptions
 @param FluxAlg.source_name
   explaination of this parameter
 @param FluxSvc.source_lib
   explanation of this parameter
```

Sean put together a nice document, stored in the package's doc directory that begins to explain how to use and extend the FluxSvc classes. This information is invaluable to those who want to do more. It is recommended that a section for use cases be added, showing what classes and methods are invoked for particular tasks. For example, what happens when a user specifies FluxAlg.source_name = "backgndmix"? An updated version of this document should be stored on the SAS web site. Bookmarks should be inserted to allow easier navigation within each page.

Included in Sean's write-up is a description of the XML source_library files. This is a good first draft to show users and devlopers what goes into the source descriptions. There were still some remaining questions, such as the use of the flux tag. Does flux really mean the rate in this context? For example, if one sets flux=1 with a point source, we will generate a source with 1 particle/cm2/s. If one wants to generate a 5 years' worth of data, is there any tag in source xml file which will allow one to do that?

Complete documentation of the current standard sources available in the source library would be very helpful for users. Such documentation would let users know what is available, and provide good examples of specifying parameters and creating new sources.

Code Review

General comments

It is strongly recommended that a standard set of units be adopted for time, energy, distance, etc. While it may be convenient to provide the facilities to handle multiple conventions, i.e. GeV and MeV, in the long run, this is a dangerous practice. Namely, sources can determine whether their energies are provided in GeV or MeV. GLAST has chosen MeV as its standard unit for energy and we should strive to adhere those recommendations.

A list of allowed particle names should be compiled and documented. While it is true that the low-level names used by the simulation are determined by the Monte Carlo generator, in this case Geant4, it would be useful for GLAST to develop a standard list. In fact, it may be best for GLAST to define their standard set of particle names and convert from those names used within external libraries, rather than be held hostage to the naming conventions of external code. This would also seem prudent if we ever intend to use another Monte Carlo generator such as Gismo - which may have a different set of particle names.

When defining spectra external to the FluxSvc package, the Gaudi Tool interface will be preferred over the older mechanisms. Unfortunately, there was not much material concerning the specific interface. The developers are asked to update the documentation and additionally, provide a generic explanation of Gaudi Tool usage for the GLAST Gaudi User Guide.

The transformation from galactic coordinates to GLAST zenith-pointing frame should be reviewed. Seth Digel recommended that the x-axis orientation should be determined by how to keep the solar panels oriented towards the sun.

Plans should be made for upgrading from strstream to sstream, especially before we upgrade compilers. The use of strstream can be found in CompositeSource, EventSource, FluxException, FluxMgr, FluxSource, and SimpleSpectrum.

Older code, such as FluxSource should be cleaned up as time allows. There are certainly old deprecated methods that should be removed to make the code more readable.

FluxAlg

FluxAlg::execute has an ugly hack for handling protons.

```
   //here's where we get the particleID and mass for later.
   if( particleName=="p") particleName="proton";
```

This would be better handled by having a standard set of particle names for GLAST. Are we certain that there are no other particles names that would require special treatment?

FluxSvc

FluxSvc which is a part of the Gaudi interface to the flux classes, defines its interface id according to the Gaudi standard:

static const InterfaceID IID_IFluxSvc(910, 1, 0);

The three parameters are interface id, major version, minor version. How are these values determined? Does the core group have a standard mechanism for choosing the primary id for each service? When are major and minor versions updated?

GPS

There is a rockingAngleTransform(double time) method. The time parameter is not used in the function. Is this just a placeholder for a future version of this routine?

FluxSource

The FluxSource class contains two member variables: m_lauchDir, m_correctedDir which is the direction corrected by tilting angle. If the source is just a simple particle source with GLAST not in orbit, m_correctedDir should be equal to m_lauchDir. The FluxSource::event method will always apply the correction via the call to correctForTiltAngle(). It seems there should there be a mechanism to turn this correction off.

FluxSource defines a sqr function that computes the square of an input value. Is this method really necessary? If so, it belongs in the facilities package.

Persistence of source and spectrum used to generate events

Some mechanism should be put in place to store the source and spectrum that generated each Monte Carlo event. The first issue to be addressed is how to uniquely identify which spectrum produced an event? Sean recommended a combination of the numSource and title for every event. How is numSource determined? Is it just a value assigned at run-time dependent upon the order the spectra are loaded? Are we interested in storing the source name - the name used in the source_library as well as the title of the spectrum that actually created the event? The actual source, may be a composite source, so knowing the source name would not necessarily uniquely identify the spectrum.

Where will this information be stored on the TDS? What should the persistent form look like? The source name could be stored in a file header, while the specific spectrum name and id number should be stored on a per event basis. These are issues that should be sorted out by the Data Structures Task Force.

Conclusions

This review was for the most part an introduction to the package and its contents. FluxSvc should be reviewed again in the near future to see that the documentation standards are upheld and to really take the time to look over the code. The documentation prepared by Sean Robinson is a step in the right direction.

Action Items for FluxSvc Developers

- *Documentation Issues*
  - Update the mainpage to provide more details about the FluxSvc package
  - Document jobOptions parameters in the mainpage.
  - Add standard class headers to all class declarations.
  - Update all source files to include standard source file comment blocks.
  - Provide meaningful method comment blocks adhereing to our standard:
    // Purpose and Method:
    // Input:
    //Output:
  - Add a use cases section to the document FluxSvcDoc.htm.
  - Provide an updated copy of the web document, FluxSvcDoc.htm, to be stored on the SAS web site.
  - Document standard particle names
  - Document FluxSvc's use of the Gaudi Tool interface and provide a generic description for inclusion in the Gaudi User Guide.
  - Document the current set of standard sources available in the source_library.
- *Implementation Details*
  - Standardize units for energy, time, distance, etc.
  - Consider creating a GLAST standard set of particle names.
  - Review the mechanics of the galactic coordinate transformation.
  - Upgrade from strstream to sstream whereever possible.
  - Provide a mechanism to turn off the tilt angle correction, if desired.
  - Propose a scheme to uniquely identify the source and spectrum that created a particular Monte Carlo event.

Action Items for Core Software Group

- Document standards for setting the identifiers for Gaudi Services.