Database Package Code Review

In 2 parts Nov 24, 2003 and Jan. 29, 2004

Database Package Authors: Chunhui Pan, Tom Stephens
Reviewers: Joanne Bogart, Heather Kelly, Matt Langston, Julie Mcenery
Attendees: James Chiang, Dave Davis, Richard Dubois, Yasushi Ikebe, James Peachey, Bob Schaefer

Introduction

This was the first code review performed on code produced solely by the SSC. This is noteworthy in that, this was the first time the LAT collaborators took an active look at this important new package. This code is actively developed using the SSC CVS repository and then deliveries are made to the LAT CVS repository for major tags.

The Database package is currently a container package, not precisely in our use of the word container, but that is easily fixed if we desire to do so. There are three sub-packages: D1, D2, QueueManager. This review considers all three packages' tag v1.

We would like to thank Chunhui Pan and Tom Stephens for their hard work and cooperation during this review process!

Documentation

General Comments

The LAT maintains a general Doxygen initialization file located in the GlastPolicy package. Would it make sense for that file to be used when generating the Doxygen for the code produced by the SSC?

It appears that the subpackages maintain their own copy of the common include file: message.h? If so, this should be located in one "common" subpackage and shared.

QueueManager

The mainpage appears to be in good shape. It would be nice to include a diagram of the system that shows where the QueueManager fits in. There was a duplicate copy of the mainpage.h file in the LAT CVS repository – that should be removed. The introduction is helpful and detailed. The Requirements section is usually meant to be a copy of the requirements file in the cmt directory. It was nice to see the requirements of the package, though. Perhaps we consider calling the section for the requirements file – "Requirements File", and then retain a new section for the requirements of the package. The Documentation Task Force should take that up as we revamp the code reviews.

Many of the classes are well documented with Doxygen comments. There are some files where the Doxygen is missing entirely, such as: Selector.h and ServerSocket.h. In the case of ServerSocket – the comments that would typically be included in the *.h file as Doxygen comments are located in the source file. It should be relatively easy to move these comments to the *.h file, and format them as proper Doxygen.

Action Items:

- Include a diagram in the mainpage.
- Remove duplicate mainpage in the LAT CVS repository.
- Complete Doxygen comments for all classes.

D1 and D2

The mainpage should include a copy of the requirements file located in the cmt directory. Each major component includes a nice introduction that explains the background of the component. Overall the Doxygen documentation is very well organized and easy to read.

Code Review

General

This package should be formatted as a proper container package. This can be accomplished by adding a cmt directory at the top level of the Database package, where the requirements file is used to specify the versions of the subpackages – as is done in the IExternal package.

The name of the package should be reviewed – it has been stated that it is too general.

If we do not already set cflags in GlastPolicy, it seems we should consider doing so. The D1 and D2 servers are in C, as well as the OnboardFilter package.

QueueManager

At the time of the review, the main program contained a number of subsections that should be extracted and made into separate routines. All literals should be replaced by proper constants. Both of these items have since been addressed.

A test program should be made available. This will be done once the new testing standards are in use.

It is best not to name types or variables after their implementation (e.g., QueryMap), but rather after their application function. In the future, the implementation may change and one would not want to be forced to modify the type name.

Within QueryManager.cxx: please follow standard naming convention: class names start with a capital letter, variables for individual objects start with a small letter. See, for example, line 46:

```
ServerMap *MapServer = new ServerMap;
```

The file, convertToString.h, provides a method to convert a double into a string. The LAT has a package called “facilities” that includes a similar routine. It is recommended that this package’s utilities be used when possible.

Action Items:

- Minimize the size of the main routine, and extract separate tasks into separate methods. {This has already been done}
- Provide a test routine.
- Follow standard naming conventions for variables.
- Avoid naming types or variables according to their implementation.
- Consider using the LAT facilities package for low-level utilities such as convertToString.

D1 and D2

At the time of the review, the main programs for both the D1 and D2 servers, there were a number of subsections that should be extracted and made into separate routines. The author is already aware of this and there are plans to reorganize the main routine.

There seems to be extensive shared functionality between the D1 and D2 servers. It would increase maintainability if those bits could be extracted and shared between the two components.

Is the extract function (located in both the D1ServerFunctions and D2ServerFunctions files) necessary? Could the standard functions strtok or strsep be used instead?

Action Items:

- Minimize the size of the main routine, by extracting the separate tasks into separate functions.
- Share the common functionality between the D1 and D2 servers as a single set of routines.
- Consider using standard functions such as strtok or strsep rather than creating a user defined routine that performs a similar purpose.