TkrRecon Code Documentation Review
Documentation Task Force
February 27, 2002

Attendees: Toby Burnett, Richard Dubois, Leon Rochester, Tracy Usher
Review committee: Joanne Bogart, Heather Kelly (chair), Alex Schlessinger, Mark Strickman, Karl Young

Introduction
The TkrRecon Code Documentation Review was the first in a series of code documentation reviews. We would like to express our appreciation to the TkrRecon developers for their efforts in documenting their code. This first review was certainly a learning experience for both developers and those writing up code documentation recommendations. As a result, there will be updates to the recommendations and some additional work for developers to adhere to the standards.

Mainpage
The TkrRecon mainpage contains all recommended components: package overview, literal inclusion of requirements file, and reference to release.notes. Tracy and Leon provided a nice introduction to TkrRecon. More of this detailed information should be provided in the Introduction section of the mainpage.

release.notes
Leon went to great effort to update the release.notes in the TkrRecon package. Unfortunately, the ChangeLog does not contain information about when tags are applied to a package - so he was forced to go back through the CVSweb to determine what updates pertain to which tags. He made a valiant effort and the format he used for the release.notes is worthy of mention:

```
Package TkrRecon
 Manager: Leon Rochester

 v4r4p6  26-Feb-2002  LSR   same as previous, but with more documentation
 v4r4p5  22-Feb-2002  LSR   Works on unix after many little fixes from Alex and Toby
```

This format will be provided as an example in the Code Documentation Recommendations. Leon noted that keeping the release.notes up to date as the code is modified is much easier than trying to go back and recreate the history. It was requested that the Documentation Task Force look into the possibility of storing tags in the ChangeLog.

Doxygen Documentation
It was noted that all of the GF* classes will be removed, so there is little cause to document these classes now. All code that is available for the upcoming May release of TkrRecon will be properly documented according to the code documentation recommendations. In the meantime, we evaluated those parts of the code that were documented for this review: TkrCluster, TkrClusters, TkrClusterAlg, and TkrBadStripsSvc.

In all cases, the Doxygen class descriptions need more information. The class description should provide an overview of a class, what it does, and how it does it. In some cases the "how" is delegation to another class; if so, a comment to this effect belongs in the class description. In the case of the TkrClusters class, an overview of the class was provided with the constructor's declaration. These comments are more properly located in the Doxygen class description comment block.

The output provided by Doxygen is at times untidy and aligned in unexpected ways. One example pointed out during the review was the view enum in the TkrCluster header file:

```
00032     {
00033         X,
00034             Y,
00035             XY
00036     };
```

```
However, when one goes back to the original TkrCluster.h file, one finds
irregular indentation:
    enum view
    {
        X, /**< cluster measures X */
            Y, /**< cluster measures Y */
            XY /**< not valid for clusters */
    };
```

Developers must insure that standard indentation is used within all files to produce more readable Doxygen output. Specifically, tabs should not be used rather editors should be set to insert blanks rather than tabs. Other issues, such as line width, will be considered by the Documentation Task Force, to see what control we have over the Doxygen processing.

It was decided that Gaudi required routines need not be documented as "required by Gaudi". Rather, providing Doxygen comments should be limited to those instances where the Gaudi required routines do something special within the class in question.

In general, the developers did a good job of avoiding documenting obvious routines such as constructors and destructors. In those instances where a constructor required explanation, comments were present.

Source Code Documentation

The recommendations for the standard source code comments will need to be updated. It was not clear what and how much detail is required. The current recommended comment header is:

// Purpose and Method
// Inputs
// Outputs
// Dependencies
// Restrictions

The comment headers required for major methods should explain what the routine does and how it does it. This is what is meant by "Purpose and Method". Multiple line explanations are allowed and in many instances encouraged. Some methods may not require all of the recommended fields, so the unused fields may be omitted. The input and output sections do not make it clear whether we are referring to explicit parameters or to data put on or retrieved from

the transient data store (TDS). It was suggested that new sections be provided for TDS INPUT and TDS OUTPUT.

It was unclear how to handle interface classes, which just provide a wrapper for a class that does all of the work. It is felt that the interface class methods should denote where the actual work takes place and point the reader to the appropriate routines. One should still denote what the purpose of the routine is - even if it is just an interface to another class' method. For example, TkrClusterAlg::execute needs more details concerning its purpose and method. The comment *// Purpose and Method: makes TkrClusters* does not provide enough information. While all of the work is performed in the TkrClusters constructor, we have no way to know that without reading the code itself. Here is an example:
// Purpose and Method: Creates TkrClusters using the TkrClusters constructor and stores the
// collection on the TDS.

All files should be checked to insure that indentation is consistent. Editors should be set to insert blanks rather than tabs. This may or may not be the cause of the uneven indentation in some of the TkrRecon files. For instance, TkrBadStripsSvc.cxx contains uneven indentation:

```
StatusCode TkrBadStripsSvc::initialize()
{
    // Purpose: reads in bad strips and constructs in-memory bad strip vectors
         // Inputs:  None
         // Outputs: Status code (Success/Failure)

         MsgStream log(msgSvc(), name());
    StatusCode sc = StatusCode::SUCCESS;
```

Conclusions

The TkrRecon developers made a good first effort at adhering to the new code documentation recommendations. This review pointed out some areas where the code documentation recommendations need to be made clear. The Documentation Task Force will begin revising the existing document as soon as possible.

There is much code left to document in the TkrRecon. It is expected that this will be completed for the May release of TkrRecon. It is recommended that all TkrRecon developers be involved in this process. This job is too much for one individual and having input from multiple sources will help produce more useful documentation.

During the documentation process, exchanges between the developers and the Documentation Task Force to clarify the requirements proved useful to both sides. It is hoped that these conversations will continue even though we are now beyond the documentation review.

Another review of the TkrRecon documentation should take place after the next release in May, 2002.

Action Items for TkrRecon Developers

- Involve all TkrRecon developers in the documentation workload.

- All code that is available in the TkrRecon release of May 2002, should be updated to conform to the code documentation standards.
- Update the mainpage description to include more details concerning the TkrRecon package.
- Provide more information in the Doxygen class headers.
- Update source file comments to include more details about the "Purpose and Method". Even interface classes should include some information about what the routine does and inform the user that real work occurs in some other routine.
- Provide consistent indentation within files.

Action Items for the Documentation Task Force

- Consider the possibility of storing tags in the ChangeLog
- Determine how to control Doxygen processing, such as the line width of output files.
- Update the recommended source code header to include TDS INPUT and TDS OUTPUT. Clarify meanings of all fields, including non-trivial examples as needed.
- Update the code documentation recommendations to include the TkrRecon release.notes format as an example.