LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Document Type	**LIGO-T980117-00 - E**	12/4/1998

The Frame API's baseline requirements

James Kent Blackburn

Distribution of this document:

LIGO LDAS Group

This is an internal working document
of the LIGO Project.

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (818) 395-2129
Fax (818) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

WWW: http://www.ligo.caltech.edu/

The Frame API's
baseline requirements

James Kent Blackburn

California Institute of Technology
LIGO Data Analysis Group
December 4, 1998

I. Introduction

A. General Description:

1. The frameAPI is responsible for reading, writing and customizing *Common Data Frame Format Files* for Interferometric Gravitational Wave Detectors (*see LIGO-T970130-B-E*) in the LDAS distributed computing environment using an interpreted command language. However, it will not change any values stored in the Frame file.
   a) The interpreted command language to be used is TCL/TK, which provides a command line, scripting and a graphical interface.
   b) The TCL/TK commands are extended to support low level system interfaces to the Frames and system I/O functions to the Frame files, as well as greater computational performance using C++ code that utilizes the standard TCL C code API library in the form of a TCL/TK package.
2. The frameAPI's TCL/TK script accesses the frameAPI.rsc file containing needed information and resources to extend the command set of the TCL/TK language using the frameAPI package, which exists as a shared object.
3. The frameAPI will receive its commands from the managerAPI, reporting back to the managerAPI upon completion of each command. This command completion message will include the incoming identification used by the manager to track completion of sequenced commands being handled by the assistant manager levels of the managerAPI.

B. The frameAPI.tcl Script's Requirements:

1. The frameAPI.tcl script will provide all the functionality inherited by the genericAPI.tcl script (*i.e. help, logging, operator & emergency sockets, etc.*).
2. The frameAPI.tcl script will report to the managerAPI's receive socket upon completion of each command issued by the managerAPI's assistant manager levels. This involves transmission of a message identifying the specific command completed as coded by the managerAPI (*see LIGO-T980115-0x-E for details*).
3. The frameAPI.tcl script will validate each command received on the operator or emergency socket as appropriate for the frameAPI to evaluate. This includes validation of commands, command options, encryption keys and managerAPI identification indexes.

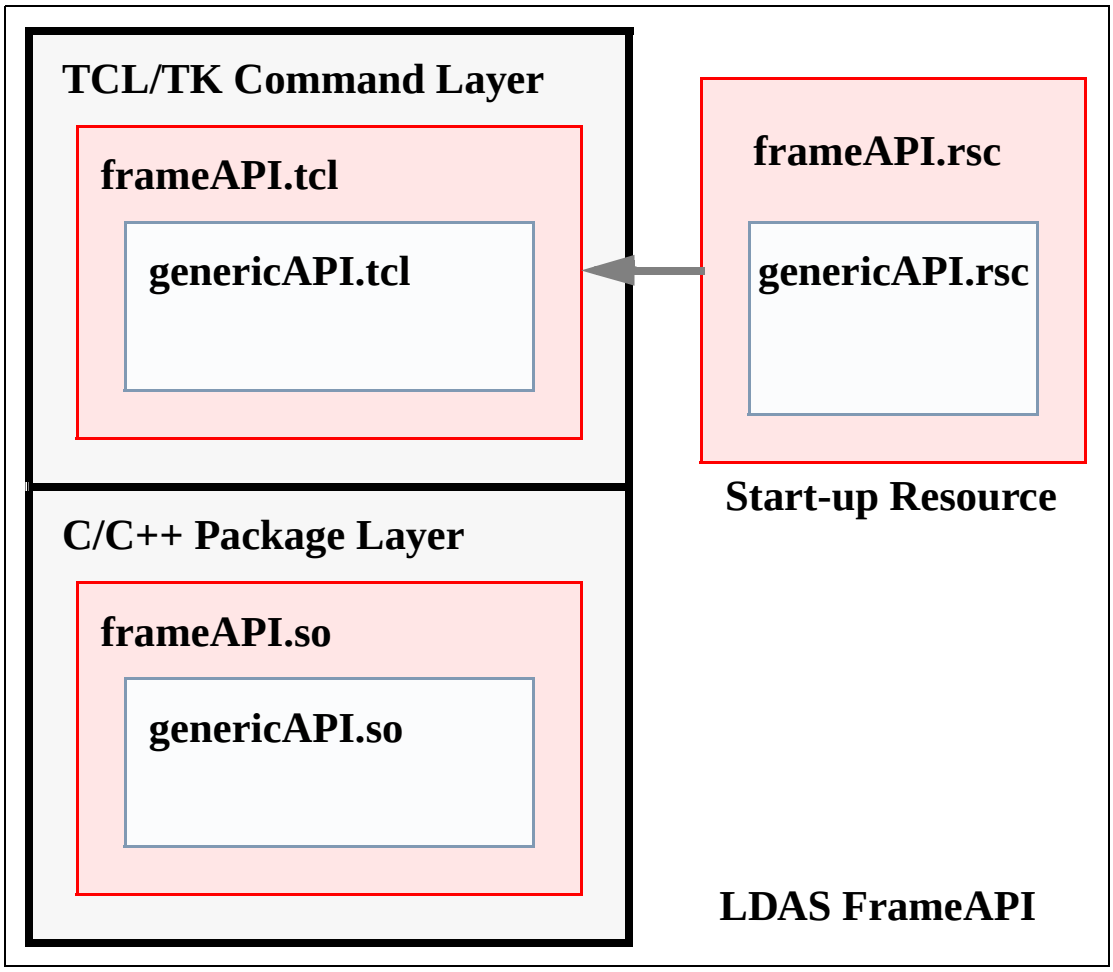
4. The frameAPI.tcl script will manage anonymous FTP connections to client machines for the purpose of transferring Frame files or resulting Frame files constructed by the FrameAPI.
5. The frameAPI.tcl script will provide the functionality to connect to the LIGO Data Acquisition System (via a TCP/IP socket) for the purpose of listening for announcements of new LIGO Frames written to the file system by the DAQ.
6. The frameAPI.tcl scrip will provide the functionality to continuously read in and process the next available Frame file written by the LIGO DAQ until a command from the ManagerAPI stops the reads. This functionality should be supported under a separate TCL/TK interpreter.
7. In the event that the LIGO Data Acquisition System is not on-line, the frameAPI.tcl script shall be able to monitor / poll the filesystem in order to detect the appearance of new Frame files for use in LDAS.
8. In the event that an exception occurs while processing a command, the frameAPI.tcl layer will report the exception to the ManagerAPI's *emergency socket* along with the necessary command identification issued by the managerAPI with the FrameAPI command.
   Note: Once reported to the managerAPI, the appropriate *assistant manager* will terminate the high level command and the userAPI that issued this high level command will be notified of the exception.

C. The frameAPI.so Package Requirements:

1. The frameAPI.so package will be developed in C++ using the C language interface to TCL/TK to communicate with the TCL/TK command layer. The wrappers between C++ functionality and TCL/TK command language extensions will be *machine generated* using the *SWIG* API code writer.
2. The frameAPI.so package will interface with Frames using the *C++ Frame Class Library* developed by LIGO and VIRGO. This library is a C++ implementation of the LIGO-970130-B-E specification.
3. The frameAPI.so package will inherit the functionality to communicate data through the data sockets from the genericAPI.so package.
4. The frameAPI.so package will support reading and writing of frame files on the local file system.
5. The frameAPI.so package with extend the genericAPI.so package's object oriented data socket communications to support Frame Classes, in addition to the objects handled immediately by the genericAPI.so package.
6. The frameAPI.so package will support streaming memory buffered versions of frame or frame files out the data sockets.
7. The frameAPI.so package will support simplifying full LIGO frames into reduced frames containing a subset of the original frames data sets.

8. The frameAPI.so package will support concatenation of frames of similar content into a single from of longer time duration.
9. The frameAPI.so package will support breaking up a long duration frame into shorter time duration frames of specified time lengths.
10. The frameAPI.so package will support extraction of individual frame attributes from the Frame or any of its internal structures.
11. The frameAPI.so package will provide translation of subsets of Frames into Internal LDAS Light Weight Data Format objects (*see LIGO-T980094-0x-E*).

II. Component Layers of the LDAS FrameAPI



A. LDAS Distributed FrameAPI:

1. The LDAS distributed frameAPI is made up of two major layers.
   a) TCL/TK Layer - this layer is the command layer and deals primarily with commands and/or messages and their attributes and/or parameters, as

well as communicate with the underlying Package Layer through TCL/TK extensions.

b) C/C++ Package Layer - this layer is the data engine layer and deals primarily with the binary data and the algorithms and methods needed to manipulate LIGO's data

2. The TCL/TK layer consists of two internal and two external components, designed to optimize code reuse at the level of the command language used in all LDAS API's.

   a) The frameAPI.tcl - this TCL/TK script contains specialized TCL/TK procedures and specialized command language extensions which are particular to the frameAPI in the LDAS architecture.

   b) The genericAPI.tcl - this TCL/TK script contains the common TCL/TK procedures and command language extensions found in all LDAS API's. the genericAPI.tcl code will be sourced in the frameAPI.tcl script.

   c) The frameAPI.rsc - this TCL/TK script contains the start-up and configuration defaults which are unique to the frameAPI.

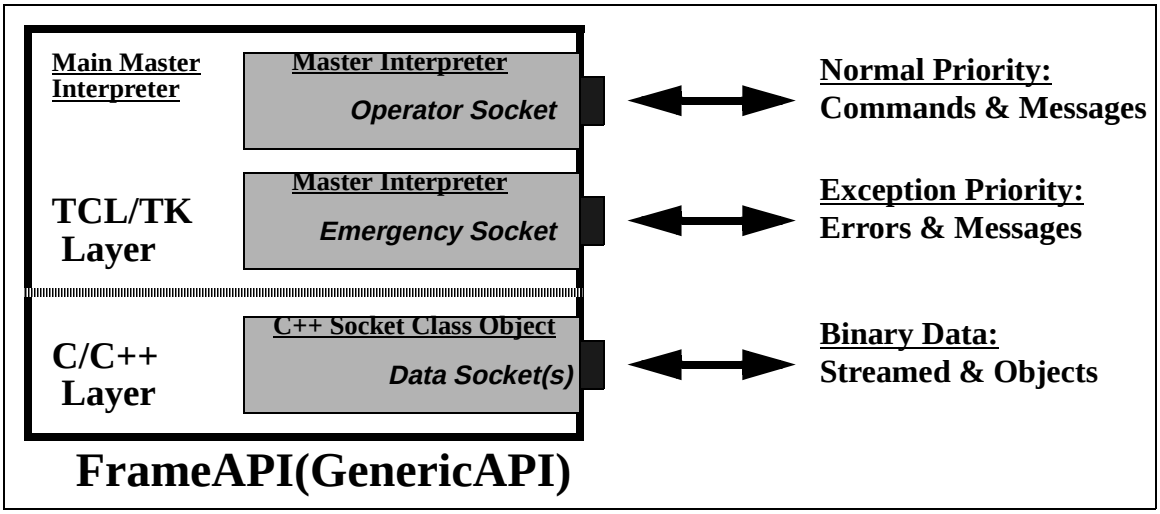
   d) The genericAPI.rsc - this TCL/TK script contains the start-up and configuration defaults which are common to each LDAS API. The genericAPI.rsc will be embedded in the frameAPI.rsc file.

3. The C/C++ package layer consists of two internal components, each developed in C++ and C to take advantage of the higher performance associated with compiled languages which is needed for the types of activities that are being carried out in this layer and loaded as shared objects.

   a) The frameAPI.so - this shared object contains the C++ classes and C interface functions needed to extend the command language set of each frameAPI, allowing it to more efficiently manipulate Frames.

   b) The genericAPI.so - this shared object contains the C++ classes and C interface functions needed to extend the command language set of all API's in LDAS, allowing efficiency and optimal code reuse. It will be linked into the frameAPI.so shared object directly.

III. Communications Provided to FrameAPI by GenericAPI



A. Socket Based Communications in FrameAPI:

1. The genericAPI will provide the frameAPI with an internet socket within the TCL/TK layer that is the primary communication port for commands and messages of a normal priority. This port is commonly referred to as the *Operator Socket* to reflect its association with normal operations. Requirements on this socket are that defined by the genericAPI.
2. The genericAPI will provide the frameAPI with dynamic internet sockets within the C/C++ layer that is used to communicate all data (*typically binary data*) in the form of streamed binary data or distributed C++ class objects using the ObjectSpace C++ Component Series Socket Library. This port is commonly referred to as the *Data Socket* to reflect its primary duty in communicating data sets. Requirements on this socket are defined by the genericAPI.

IV. Software Development Tools

A. TCL/TK:

1. TCL is a string based command language. The language has only a few fundamental constructs and relatively little syntax making it easy to learn. TCL is designed to be the glue that assembles software building blocks into applications. It is an interpreted language, but provides run-time tokenization of commands to achieve near to compiled performance in some cases. TK is an TCL integrated (as of release 8.x) tool-kit for building graphical user interfaces. Using the TCL command interface to TK, it is quick and easy to build powerful user interfaces which are portable between Unix, Windows and Macintosh computers. As of release 8.x of TCL/TK, the language has native

support for binary data.

B. C and C++:

1. The C and C++ languages are ANSI standard compiled languages. C has been in use since 1972 and has become one of the most popular and powerful compiled languages in use today. C++ is an object oriented super-set of C which only just became an ANSI/ISO standard in November of 1997. It provided facilities for greater code reuse, software reliability and maintainability than is possible in traditional procedural languages like C and FORTRAN. LIGO's data analysis software development will be dominated by C++ source code.

C. SWIG:

1. SWIG is a utility to automate the process of building wrappers to C and C++ declarations found in C and C++ source files or a special *interface file* for API's to such languages as TCL, PERL, PYTHON and GUIDE. LDAS will use the TCL interface wrappers to the TCL extension API's.

D. Make:

1. Make is a standard Unix utility for customizing the build process for executables, objects, shared objects, libraries, etc. in an efficient manor which detects the files that have changed and only rebuilds components that depend on the changed files.If/when LDAS software becomes architecturally dependent, it will be necessary to supplement make with auto-configuration scripts.

E. CVS:

1. CVS is the Concurrent Version System. It is based on the public domain (and is public domain itself) software version management utility RSC. CVS is based on the concept of a software source code repository from which multiple software developers can check in and out components of a software from any point in the development history.

F. Documentation:

1. DOC++ is a documentation system for C/C++ and Java. It generates LaTeX or HTML documents, providing for sophisticated online browsing. The documents are extracted directly from the source code files. Documents are hierarchical and structured with formatting and references.
2. TclDOC is a documentation system for TCL/TK. It generates structured HTML documents directly from the source code, providing for a similar online browsing system to the LDAS help files. Documents include a hypertext linked table of contents and a hierarchical structured format.