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

Document Type	**LIGO-T000026-00** -	**E**	03/02/2000

The Control & Monitor 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 18-34
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 Control & Monitor API's
baseline requirements

James Kent Blackburn

California Institute of Technology
LIGO Data Analysis Group
March 02, 2000

I.   Introduction

A.   General Description:

1. The controlMonitorAPI is responsible for monitoring the advanced analysis processes which are based on MPI and execute in the LDAS distributed computing parallel cluster of nodes (Beowulf). It will be split up into a separate server component and client component. This is to allow the graphical user interface driven client component to be run and displayed anywhere (and with multiple instances) while the server with carries out control and monitoring of the LDAS acts as the singular point of control and monitoring over the LDAS system's operations. It will have the following high level language elements:

    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 algorithms used to "communicate" the data, as well as provide 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 controlMonitorAPI's TCL/TK script accesses the controlMonitorAPI.rsc file containing necessary information and configuration resources to extend the command set of the TCL/TK language using the controlMonitorAPI package, which exists as a shared object.

3. The controlMonitorAPI will receive its commands and messages from the managerAPI, from a graphical user interface and from the mpiAPI.

4. When carrying out commands from the managerAPI it will report 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.

5. The controlMonitorAPI will control (customize) MPI jobs queues (machine-files) on the LDAS system. These job queues will be used by the mpiAPI when starting parallel processes. The exact configuration of these queues will be initialized by defaults found in the controlmonitorAPI's resource file. These job queues can be configured through user inputs, LDAS commands or by algorithms.

6. The controlMonitorAPI will monitor the state of processes running under the control of the mpiAPI. This will be accomplished by commands sent to the controlmonitorAPI from the mpiAPI using the LDAS "*Internal Lightweight Data Format*" ILWD format.

B. **The controlMonitorAPI.tcl Script's Requirements:**

1. The controlMonitorAPI.tcl script will provide all the functionality inherited by the genericAPI.tcl script (*i.e. help, logging, operator, jobstate & emergency sockets, etc.*).

2. The controlMonitorAPI.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-E for details*).

3. The controlMonitorAPI.tcl script will validate each command received on the operator, jobstate or emergency socket as appropriate for the controlMonitorAPI to evaluate. This includes validation of commands, command options, encryption keys and managerAPI identification indices.

4. In the event that an exception occurs while processing a command, the controlMonitorAPI.tcl layer will report the exception to the ManagerAPI's *emerency socket* along with the necessary command identification issued by the managerAPI for the specific controlMonitorAPI 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.

5. The controlMonitorAPI.tcl script will be responsible for managing (customizing) the MPI job queues (machinefiles) used by the parallel processes.

6. The controlMonitorAPI.tcl script will be able to change the job priority level of individual jobs used by the mpiAPI to carry out dynamic load balancing by the mpiAPI. This will be performed by sending change of priority commands to the mpiAPI for specific parallel jobs currently executing. These change of job priority may be initiated by user interfaces or by commands issued by the managerAPI.

7. The controlMonitorAPI.tcl script will initialize, monitor and modify the lists of node names and job queues (machinefiles) which are used to identify the various allocations of compute nodes that can be used by each MPI parallel process. **Note:** This is intended to allow various job sizes and guarantee uniqueness in the queue assignments for MPI jobs.

8. The controlMonitorAPI.tcl script will track the node allocation needs of MPI jobs. As the mpiAPI report a reallocation of nodes usage to carry out an active parallel task the mpiAPI.tcl script will report new job queues configurations to the controlMonitorAPI (*see the mpiAPI documentation T990086-E*

for more details).

9. The controlMonitorAPI.tcl script will provide graphics and user interfaces using the TK widget sets.

10. The controlMonitorAPI.tcl script will provide a graphical user interface which allows users to interactively select view all LDAS API log files. The graphical user interface will provide a selection tools so that users can customize the subset of the logs viewed.

11. The controlMonitorAPI.tcl script will provide a graphical user interface which allows users to view the current configuration of parallel computing job queues which are stored in the MPI machinefiles used by the mpiAPI.

    a) The controlMonitorAPI.tcl script will provide a graphical user interface which allows authorized users to view and modify (after entering the appropriate password) the configuration of the current MPI machinefiles associated with the parallel computing job queues. Once these MPI machinfiles have been modified, the controlMonitorAPI.tcl script will make them available to the mpiAPI and notify the mpiAPI to the changes. The act of modifying and the details of the configuration changes will be recorded in the controlMonitorAPI's log file.

12. The controlMonitorAPI.tcl script will provide a graphical user interface which allows users to view the status of all MPI parallel processing jobs currently active on the LDAS parallel cluster of computer nodes. This will include

    a) the start time of any parallel computing job currently running,

    b) current percent completion,

    c) the estimated completion time,

    d) priority level,

    e) total number of nodes available to the job through the job queue being used,

    f) actual number of nodes actively being used as a result of dynamic load balancing,

    g) an optional popup graphic of the specific nodes actively being used,

    h) an optional popup graphic showing all communications between the mpiAPI and the wrapperAPI associated with each individual parallel process,

    i) an option to change the priority of any job currently running (requiring the authorization password be given),

    j) an option to kill any job currently running (requiring the authorization password be given).

13. The controlMonitorAPI.tcl script will provide a graphical user interface to view the log file containing the

          a) start time,

          b) stop time,

          c) delta wall time,

          d) job queue name,

          e) maximum number of nodes used and

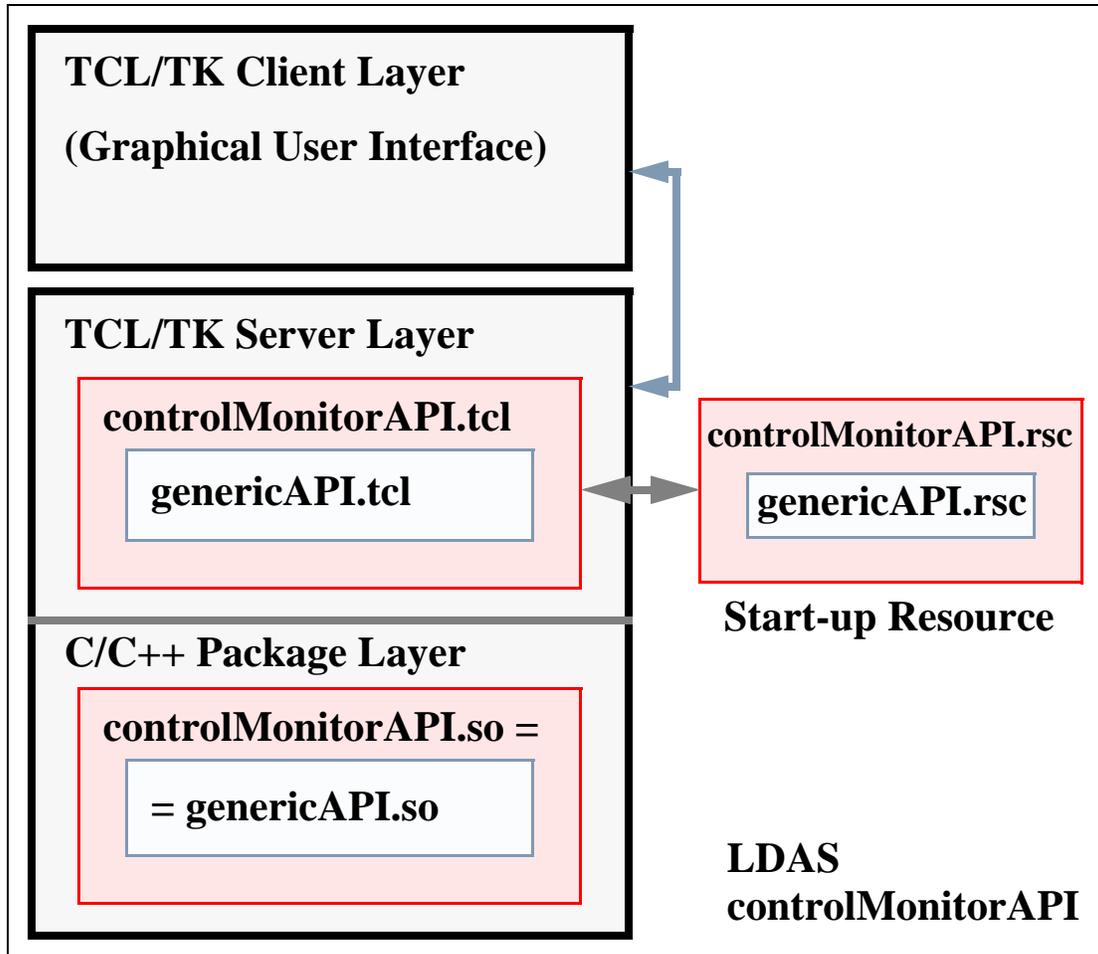   f) parameter set associated with the command line options for the wrapper-API

for all completed parallel jobs started by the mpiAPI. The actual logging of this information will be performed by the mpiAPI. The graphical user interface will provide a selection tools so that users can customize the subset of the logs viewed.

14. The controlMonitorAPI.tcl script will provide a user interface which allows users to log into any node of the Beowulf Cluster as the ldas user after providing the appropriate authorization password. This will be for the purpose of carrying out system administration related tasks on the Beowulf Cluster from the controlMonitorAPI's GUI. It will require the ability to tunnel throught the Beowulf Cluster's gateway and should be handled in a transparent way to the user.

15. The controlMonitorAPI.tcl script will provide a user interface which allows users to visualize the process load level on any of the nodes of the Beowulf Cluster either using or having a format similar to the unix top utility.

16. The controlMonitorAPI.tcl script will provide a user interface which allows users to visualize the process load level on any of the LDAS servers (dataserver, metaserver, etc.) found on the local LDAS network either using or having a format similar to the unix top utility.

C. The controlMonitorAPI.so Package Requirements:

1. The controlMonitorAPI.so package will not have any functionality not already provided in the genericAPI.so package, i.e., it will only need to send and receive "Internal LDAS Lightweight Data Formats" on command from the TCL/TK layer. That is to say the contolMonitorAPI.so package is equilvalent to the genericAPI.so package.

II. Component Layers of the LDAS controlMonitorAPI

```
┌──────────────────────────────────────────────────────────────┐
│  ┌────────────────────────────────┐                          │
│  │ TCL/TK Client Layer            │                          │
│  │                                │                          │
│  │ (Graphical User Interface)     │                          │
│  │                                │                          │
│  └────────────────────────────────┘                          │
│  ┌────────────────────────────────┐                          │
│  │ TCL/TK Server Layer            │                          │
│  │  ┌──────────────────────────┐  │   ┌─────────────────────┐│
│  │  │ controlMonitorAPI.tcl    │  │   │ controlMonitorAPI.rsc││
│  │  │  ┌────────────────────┐  │  │   │  ┌─────────────────┐ ││
│  │  │  │ genericAPI.tcl     │  │  │◄─►│  │ genericAPI.rsc  │ ││
│  │  │  └────────────────────┘  │  │   │  └─────────────────┘ ││
│  │  └──────────────────────────┘  │   └─────────────────────┘│
│  ├────────────────────────────────┤     Start-up Resource    │
│  │ C/C++ Package Layer            │                          │
│  │  ┌──────────────────────────┐  │                          │
│  │  │ controlMonitorAPI.so =   │  │                          │
│  │  │  ┌────────────────────┐  │  │                          │
│  │  │  │ = genericAPI.so    │  │  │      LDAS                │
│  │  │  └────────────────────┘  │  │      controlMonitorAPI   │
│  │  └──────────────────────────┘  │                          │
│  └────────────────────────────────┘                          │
└──────────────────────────────────────────────────────────────┘
```

A. **LDAS Distributed controlMonitorAPI:**

1. The LDAS distributed controlMonitorAPI is made up of three major layers.

   a) TCL/TK Client Layer - this layer is a separate client to the controlMonitorAPI which is the Graphical User Interface Layer. It manages all the TK widgets, receives information from the TCL/TK Server Layer and send information to the TCL/TK Server Layer based on mouse clicks and keystrokes.

   b) TCL/TK Server 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, and communicating with the TCL/TK Client Layer's GUI's widget set.

   c) C/C++ Package Layer - this layer is the data engine layer and deals pri-

marily with the binary data and the algorithms and methods needed to manipulate LIGO's data

2. The TCL/TK Client layer consists of GUI components such as menus, windows, views, buttons used as a user interface to allow the controlMonitorAPI's functions to be carried out by operators.

3. The TCL/TK Server 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 controlMonitorAPI.tcl - this TCL/TK script contains specialized TCL/TK procedures and specialized command language extensions which are particular to the controlMonitorAPI 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 controlMonitorAPI.tcl script.

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

   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 controlMonitorAPI.rsc file.

4. The C/C++ package layer consists of one internal components, 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 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 controlMonitorAPI.so shared object directly.

III.   Communications in controlMonitorAPI from GenericAPI
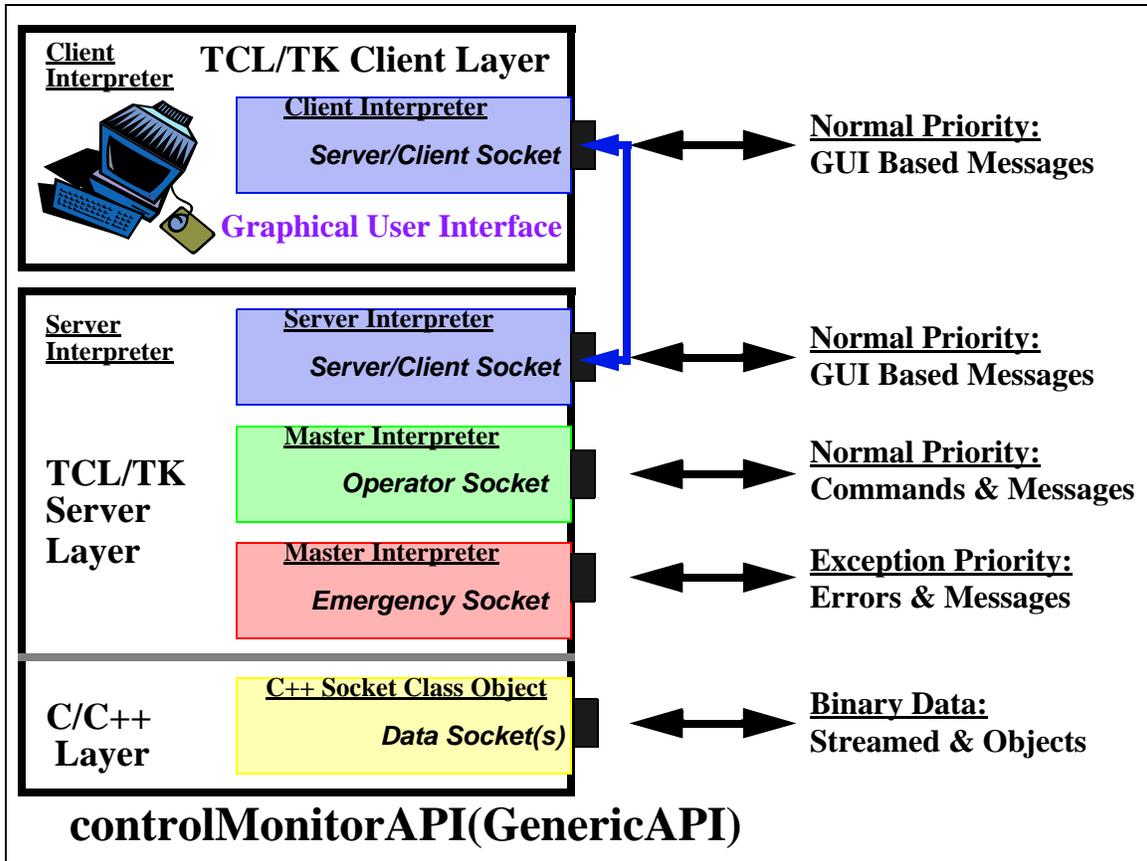
A.   Socket Based Communications in controlMonitorAPI:

1. The TCL/TK Client Layer will maintain a TCL socket connection with the TCL/TK Server which will be used to communicate information to and from the Graphical User Interface of the TCL/TK Client Layer. Through the TK widget sets available on the client side users can control and monitor the centralized server side. The TCL/TK Server Layer must support multiple connections to multiple clients. Update information from the server must be broadcast to allow all clients to remain current, while also allowing requests for changes form one client to be reflected in the displays of other clients.

2. The genericAPI will provide the mpiAPI with an internet socket within the TCL/TK Server 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. The genericAPI will also provide the controlMonitorAPI with an internel socket within the TCL/TK Server Layer for exception priority messages. This port is commonly

referred to as the *Emergency Socket* to reflect its association with exception handling.

3. The genericAPI will provide the controlMonitorAPI with dynamic TCP/IP 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 these sockets 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.   MPI:

1. The parallel software components of the LDAS will use the public domain version of MPI from MPICH, release 1.2 or greater.

2. The use of MPI code within LDAS will be restricted to the C++ interface bindings and the use of object oriented design technologies whenever possible. The templated analysis filters and associated functions are not required to be developed using C++ and object oriented design techniques. However, they must support bindings to the core C++ slave processes.

D.   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.

E.   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. The Make facility is being extended using AutoConfig, AutoMake and LibTools, all from the public domain.

F.   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.

G.   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.