Document Type **LIGO-T990037-00-** **E** 5/12/1999

The LightWeight API's baseline requirements

James Kent Blackburn

Distribution of this document:

LIGO LDAS Group

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 LightWeight API's
baseline requirements

James Kent Blackburn

California Institute of Technology
LIGO Data Analysis Group
May 12, 1999

I. Introduction

A. General Description:

1. The lightweightAPI is responsible for reading, writing and customized *LIGO Light-Weight Format Documents* which are based on the Extensible Markup Language (XML) used by the LDAS distributed computing environment using an interpreted command environment. More information about the *LIGO Light-Weight Format* can be found in LIGO-T990023-00.E.
   a) The interpreted command language to be used is TCL/TK, which provides a command line, scripting and a graphical interface environments.
   b) The TCL/TK commands are extended to support low level system interfaces to the XML based *LIGO Light-Weight Format*, system I/O and parsing functions to the *LIGO Light-Weight Format* documents, as well as providing greater computational performance using C++ code that utilizes the standard TCL-C API library in the form of a TCL/TK package.
2. The lightweightAPI's TCL/TK script accesses the lightweightAPI.rsc file containing needed information and resources to extend the command set of the TCL/TK language using the lightweightAPI package, which exists as a shared object.
3. The lightweightAPI 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 lightweightAPI.tcl Script's Requirements:

1. The lightweightAPI.tcl script will provide all the functionality inherited by the genericAPI.tcl script (*i.e. help, logging, operator & emergency sockets, etc.*).
2. The lightweightAPI.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 lightweightAPI.tcl script will validate each command received on the

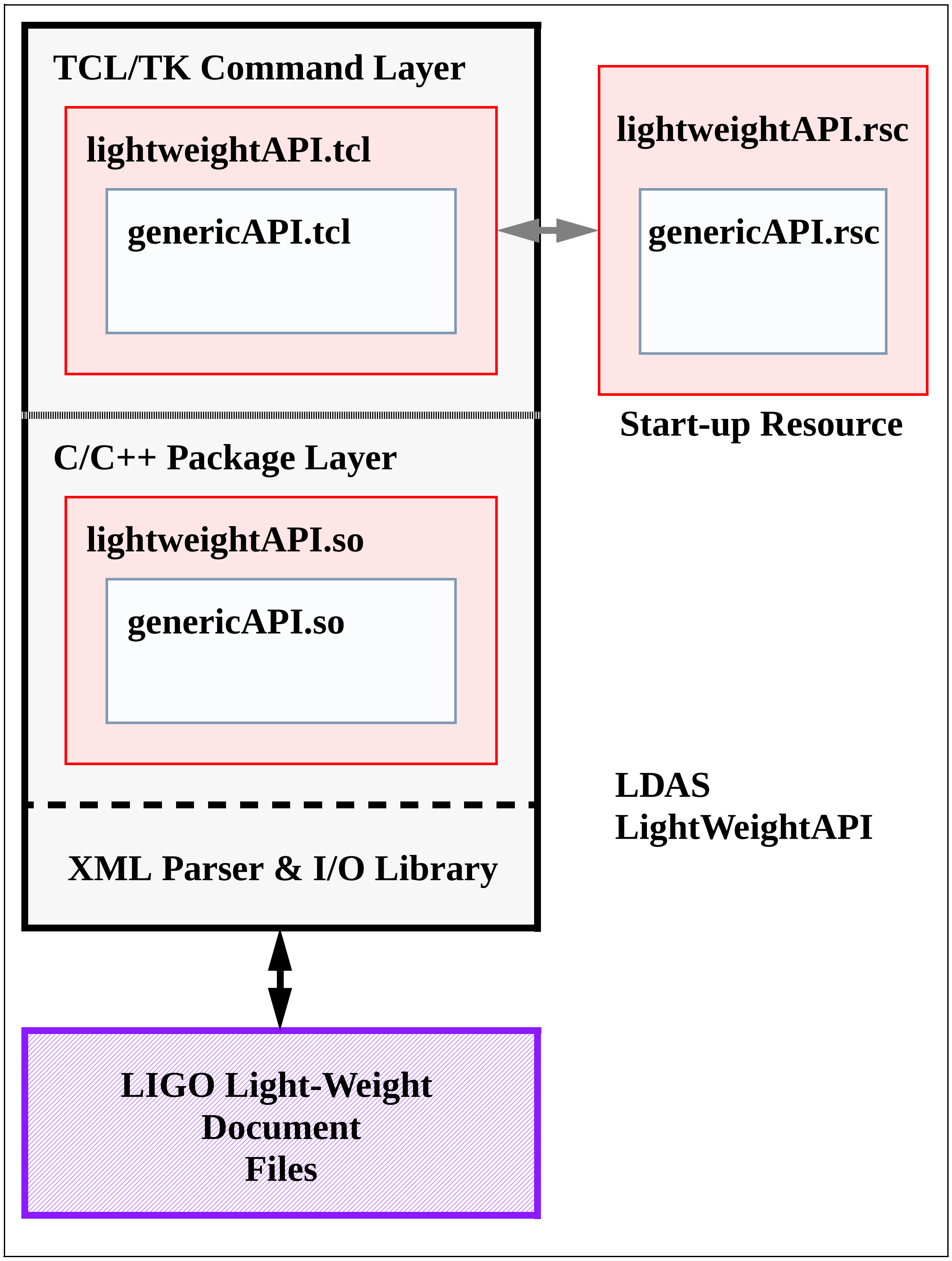
operator or emergency socket as appropriate for the lightweightAPI to evaluate. This includes validation of commands, command options, encryption keys and managerAPI identification indexes.

4. The lightweightAPI.tcl script will manage anonymous FTP connections to client machines for the purpose of transferring *LIGO Light-Weight Format Documents* constructed within the lightweightAPI.
5. In the event that an exception occurs while processing a command, the lightweightAPI.tcl layer will report the exception to the ManagerAPI's *emergency socket* along with the necessary command ***identification*** issued from the managerAPI for the lightweightAPI 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 lightweightAPI.so Package Requirements:

1. The lightweightAPI.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 generator.
2. The lightweightAPI.so package will interface and parse *LIGO Light-Weight Documents* using the *C++ XML Class Library* developed by IBM and modified by LDAS to support the Solaris and Linux OS environments. This library was originally implemented for AIX and Windows NT only.
3. The lightweightAPI.so package will inherit the functionality to communicate data through the data sockets from the genericAPI.so package.
4. The lightweightAPI.so package will support reading and writing of LIGO Light-Weight Documents as files on the local file system.
5. The lightweightAPI.so package will support streaming memory buffered versions of *LIGO Light-Weight Documents* or files out the data sockets.
6. The lightweightAPI.so package will support construction of full *LIGO Light-Weight Format Documents* from the contents of one or more ILWD formated objects.
7. The lightweightAPI.so package will support extraction (parsing) of individual elements and attributes from the LIGO Light-Weight Format and any of its internal structures.
8. The lightweightAPI.so package will provide translation of subsets of *LIGO Light-Weight Format Documents* into *Internal LDAS Light Weight Data Format* objects (*see LIGO-T980094-0x-E*).

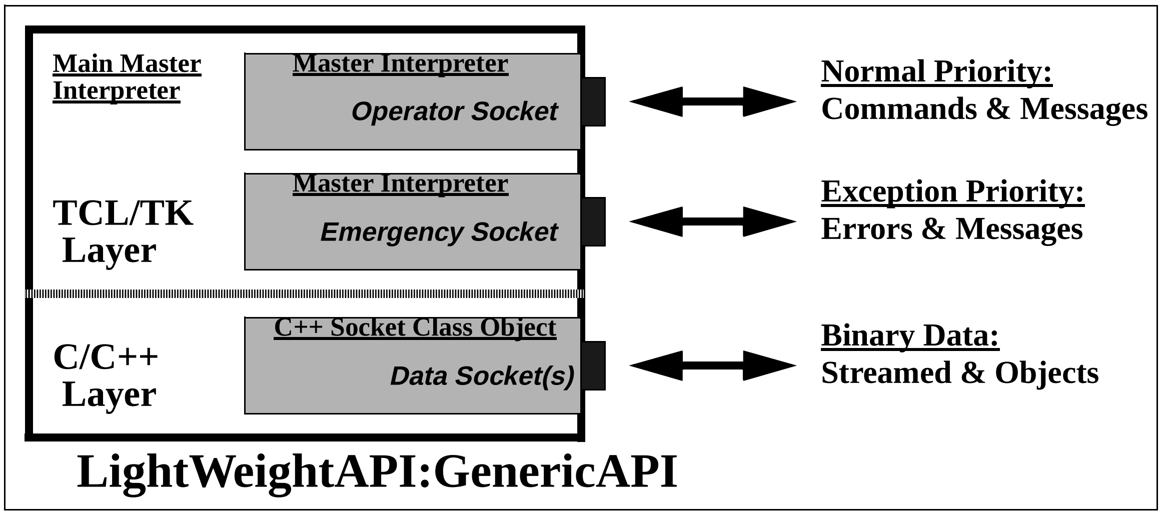
II. Component Layers of the LDAS LightWeightAPI



A. LDAS Distributed LightWeightAPI:

1. The LDAS distributed lightweightAPI 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 lightweightAPI.tcl - this TCL/TK script contains specialized TCL/TK procedures and specialized command language extensions which are particular to the lightweightAPI 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 lightweightAPI.tcl script.
    c) The lightweightAPI.rsc - this TCL/TK script contains the start-up and configuration defaults which are unique to the lightweightAPI.
    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 lightweightAPI.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 lightweightAPI.so - this shared object contains the C++ classes and C interface functions needed to extend the command language set of each lightweightAPI, allowing for more efficiently manipulation of *LIGO Light-Weight Format Documents*.
    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 lightweightAPI.so shared object directly.

III. Communications within LightWeightAPI via GenericAPI



A. Socket Based Communications in LightWeightAPI:

1. The genericAPI will provide the lightweightAPI 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. The genericAPI will also provide the lightweightAPI with the internet socket used by th TCL/TK layer to communicate abnormal conditions and error messages. This port is referred to as the *Emergency Socket* to reflect its association with exception handling. Requirements on this socket are that defined by the genericAPI.
2. The genericAPI will provide the lightweightAPI with dynamic internet sockets within the C/C++ layer that is used to communicate all data (*typically XML Document streams or ILWD 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.Now that LDAS software has become architecturally dependent, it is be necessary to supplement make with auto-configuration scripts using automake and autoconfig.

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 hyper-text linked table of contents and a hierarchical structured format.