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

Document Type **LIGO-T990002-00** - **E** 02/10/1999

The Data Conditioning 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 Data Conditioning API's *baseline requirements*

James Kent Blackburn

California Institute of Technology
LIGO Data Analysis Group
February 10, 1999

I. Introduction

A. General Description:

1. The dataConditioningAPI is responsible for cleaning, preprocessing, and statistical assessment of data from Interferometric Gravitational Wave Detectors in preparation for advanced analysis of the "conditioned" data in the LDAS distributed computing environment using an interpreted command language.
   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 "condition" 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 dataConditioningAPI's TCL/TK script accesses the dataConditioningAPI.rsc file containing needed information and configuration resources to extend the command set of the TCL/TK language using the dataConditioningAPI package, which exists as a shared object.
3. The dataConditioningAPI 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.
4. The dataConditioningAPI will receive its data in the form of the "Internal LDAS Lightweight Data Format". Data will be received at the dataConditioningAPI's data sockets (described below). Conditioned data will be sent directly to "users" of the data, e.g., metadataAPI, mpiAPI, etc., in the same format. (NOTE: "users" refers to LDAS API's in this context.)

B. The dataConditioningAPI.tcl Script's Requirements:

1. The dataConditioningAPI.tcl script will provide all the functionality inherited by the genericAPI.tcl script (*i.e. help, logging, operator & emergency sockets, etc.*).
2. The dataConditioningAPI.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 dataConditioningAPI.tcl script will validate each command received on the operator or emergency socket as appropriate for the dataConditioningAPI 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 dataConditioningAPI.tcl layer will report the exception to the ManagerAPI's *receive socket* along with the necessary command identification issued by the managerAPI with the specific dataConditioningAPI 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 dataConditioningAPI.so Package Requirements:

1. The dataConditioningAPI.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 dataConditioningAPI.so package will singularly deal with data in the form of the "Internal LDAS Lightweight Data Format".
3. The dataConditioningAPI.so package will inherit the functionality to communicate data through the data sockets from the genericAPI.so package.
4. The dataConditioningAPI.so package will support reading and writing of "Internal LDAS Lightweight Data Format files on the local file system.
5. The dataConditioningAPI.so package will provide functions for data conditioning and signal analysis based on LIGO/LSC approved numerical algorithm libraries. The functions to be supported (and used to extend the TCL/TK command language are the following:
   a) **Data Sample Rate Reduction** - This function will cleanly (without aliasing or false signal insertion) lower the sample rate and hence the bandwidth of a discretely sampled signal. The function will perform sample rate reductions in powers of two.
   b) **Heterodyning** - This function will cleanly (without aliasing or false signal insertion) frequency shift (mix) a narrow band down to DC, while at the same time lowering the sampling rate to the narrow band near DC.
   c) **Doppler Shifted Time Resampling** - This function will cleanly resample discretely sampled data to correspond to an inertial frame of reference based on a specified location on the celestial sphere using various interpolation techniques, including linear, polynomial and spline.

d) **Data Signal Bandpass Filtering** - These functions will cleanly (without aliasing or false signal insertion) lowpass, highpass and bandpass limit frequency content of a discretely sampled signal. This function will be performed using two methods:

   (1) Discrete Fourier methods in the frequency domain.

   (2) Time Domain methods such as FIR and IIR digital filters.

e) **Data Dropout Corrections** - This function will cleanly (without excess false signal insertion) correct for dropouts (dead or unrecorded stretches) in the data. Approaches for accomplishing this include, but are not limited to adding a constant value, adding Guassian noise with similar moments, adding polynomial fitted data, etc.

f) **Channel Calibration** - This function will cleanly (without aliasing or false signal insertion) amplitude and phase correct a discretely sampled signal into physical units over a specified bandwidth.

g) **Known Line Resonance Removal** - These functions will cleanly (without aliasing or false signal insertion or removal of astrophysical signals) remove known or detectable line resonances from a discretely sampled data signal. These functions will be implemented using these methods:

   (1) Kalman Filtering techniques.

   (2) Multitaper techniques.

h) **Linear Regression of Ancillary Signals** - These functions will cleanly (without aliasing or false signal insertion or removal of astrophysical signals) regress out mixed or crossover signals from a set of discretely sampled data signals with the primary purpose being to remove crossover signals from the signals of most interest such as the gravitational channel.

i) **Signal Statistical Characterizations** - These functions will calculate statistical quantities which will be useful as data descriptors to be placed in the database. Among these are the following; power spectral estimators, min, max, average, RMS, and higher order moments for a discretely sampled signal and histogram binning of discrete sample counts per ADC level bin.

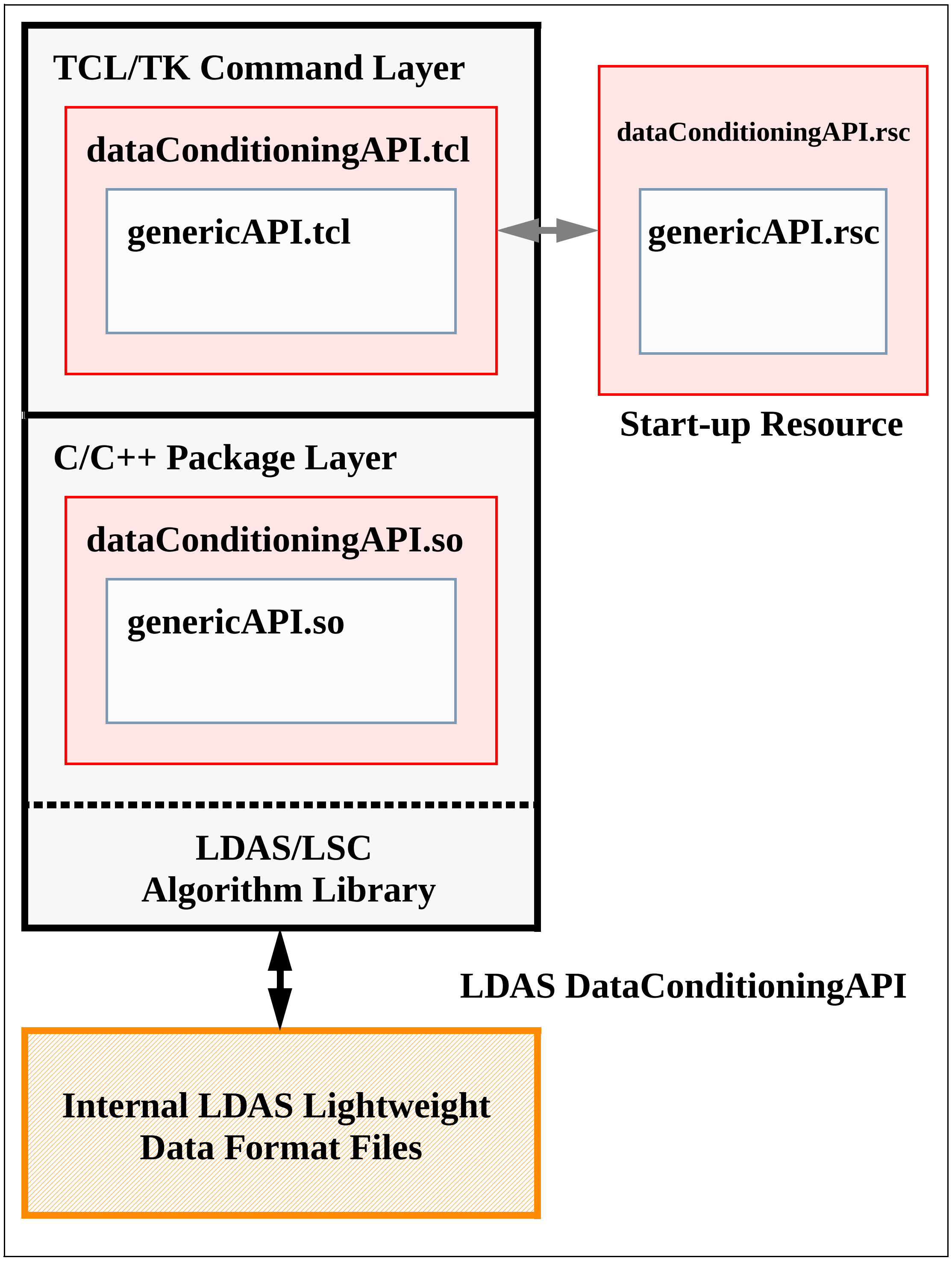
j) **Time - Frequency Transformations** - These functions will calculate two dimensional representations of a discretely sampled signal using common time-frequency transformations. Among these functions are the following:

   (1) Wavelet Analysis using a TBD set of wavelet base functions.

   (2) Moving Fourier Transforms of selected time sub-intervals.

   (3) Weighted discriminator time-frequency techniques such as is found in the GRASP package.

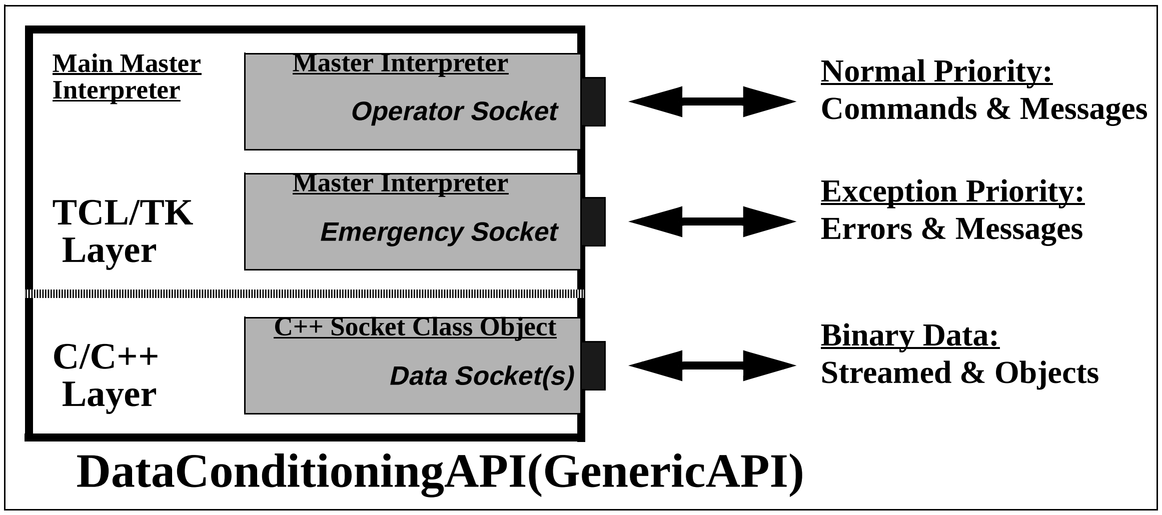
II. Component Layers of the LDAS DataConditioningAPI



A. LDAS Distributed DataConditioningAPI:

1. The LDAS distributed dataConditioningAPI 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 dataConditioningAPI.tcl - this TCL/TK script contains specialized TCL/TK procedures and specialized command language extensions which are particular to the dataConditioningAPI 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 dataConditioningAPI.rsc - this TCL/TK script contains the start-up and configuration defaults which are unique to the dataConditioningAPI.
   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 dataConditioningAPI.so - this shared object contains the C++ classes and C interface functions needed to extend the command language set of each dataConditioningAPI, allowing it to more efficiently condition LIGO data.
   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 dataConditioningAPI.so shared object directly.

III. Communications in dataConditioningAPI from GenericAPI



A. Socket Based Communications in DataConditioningAPI:

1. The genericAPI will provide the dataConditioningAPI 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 dataConditioningAPI 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 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.