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

Document Type **LIGO-T980119-01 - E** 01/18/1999

The Metadata 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 Metadata API's *baseline requirements*

James Kent Blackburn

California Institute of Technology
LIGO Data Analysis Group
January 18, 1999

I. Introduction

A. General Description:

1. The metadataAPI is the LDAS custom Database Client used to interface with the LIGO Database server in the LDAS distributed computing environment. It will use an interpreted command language for control. It will provide for all database functions such as table creation, inserts, queries, etc.
   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 metadataAPI's TCL/TK script accesses the metadataAPI.rsc file containing needed information and resources to extend the command set of the TCL/TK language using the metadataAPI package, which exists as a shared object.
3. The metadataAPI 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 metadataAPI.tcl Script's Requirements:

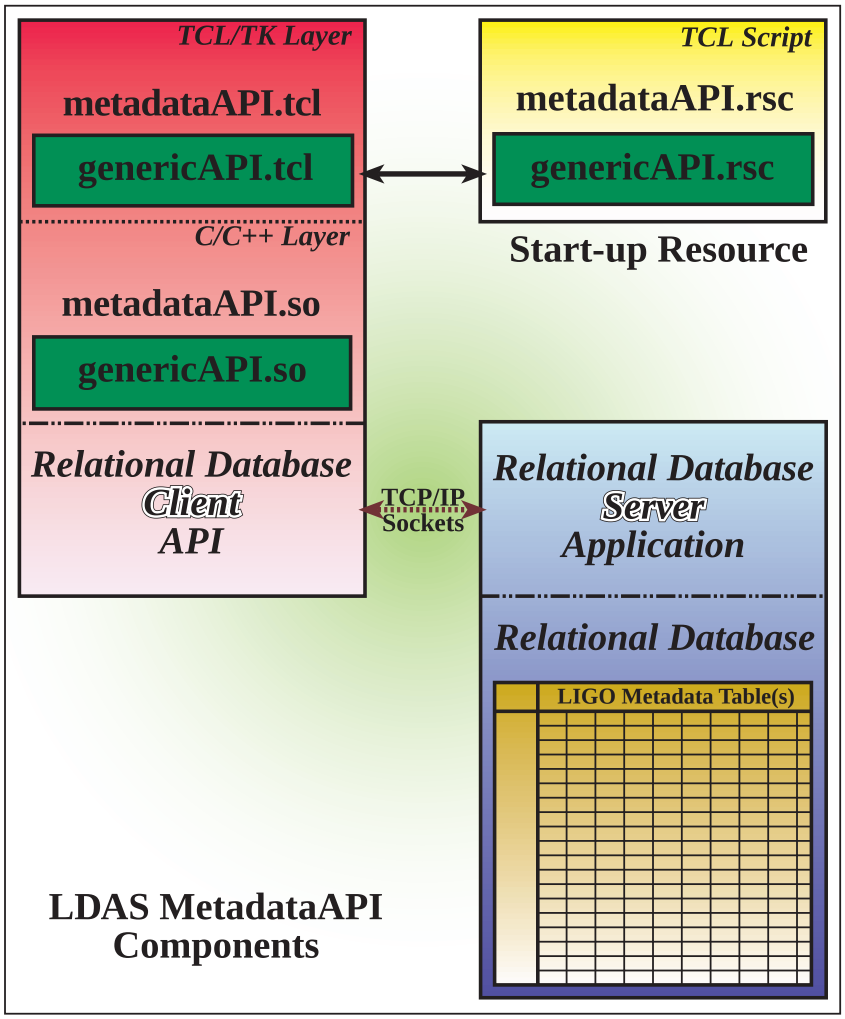
1. The metadataAPI.tcl script will provide all the functionality inherited by the genericAPI.tcl script (*i.e. help, logging, operator & emergency sockets, etc.*).
2. The metadataAPI.tcl script will report to the managerAPI's receive socket upon completion of each command issued by the managerAPI's assistant 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 metadataAPI.tcl script will validate each command received on the operator or emergency socket as appropriate for the metadataAPI to evaluate. This includes validation of commands, command options, encryption keys and managerAPI identification indexes.

4. The metadataAPI.tcl script will support casual SQL statement validation. If this validation is supported by the dynamic SQL routines in the TBD database, then this function can be moved into the metadataAPI.so package using the database client development API library.
5. In the event that an exception occurs while processing a command, the metadataAPI.tcl layer will report the exception to the ManagerAPI's *emergency socket* along with the necessary command identification issued by the managerAPI with the metadataAPI 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 metadataAPI.so Package Requirements:

1. The metadataAPI.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 metadataAPI.so package will inherit the functionality to communicate *Internal Light Weight Format Data* through the data sockets from the genericAPI.so package.
3. The metadataAPI.so package will be a LDAS custom database client, used to communicate SQL statements and data with the database server.
4. The metadataAPI.so package will provide functions for the following database management procedures:
   a) Creation of LIGO Frame metadata tables.
   b) Migration of older LIGO Frame metadata tables into upgraded LIGO Frame metadata tables
   c) Insertion and updates on entries in the LIGO Frame metadata tables.
   d) Highly repeated SQL commands will use static SQL API library calls to efficiently communicate with the database server.
   e) The metadata.so package will support dynamic SQL commands of arbitrary query content within the SQL language standard.
   f) The metadata.so package will support a superset of queries of a TBD nature which allow for more efficient interaction with LIGO metadata which doesn't directly correspond to SQL but can customized around SQL and C/C++ methods such as use of LDAS defined binary records.
5. The metadataAPI.so package will use ODBC (*Open Database Connectivity*) where possible to minimize code dependencies on vendor client API's. For those features of the database with are deemed highly useful to LIGO, the custom API calls will be acceptable.

II. Component Layers of the LDAS MetadataAPI



A. LDAS Distributed MetadataAPI components:

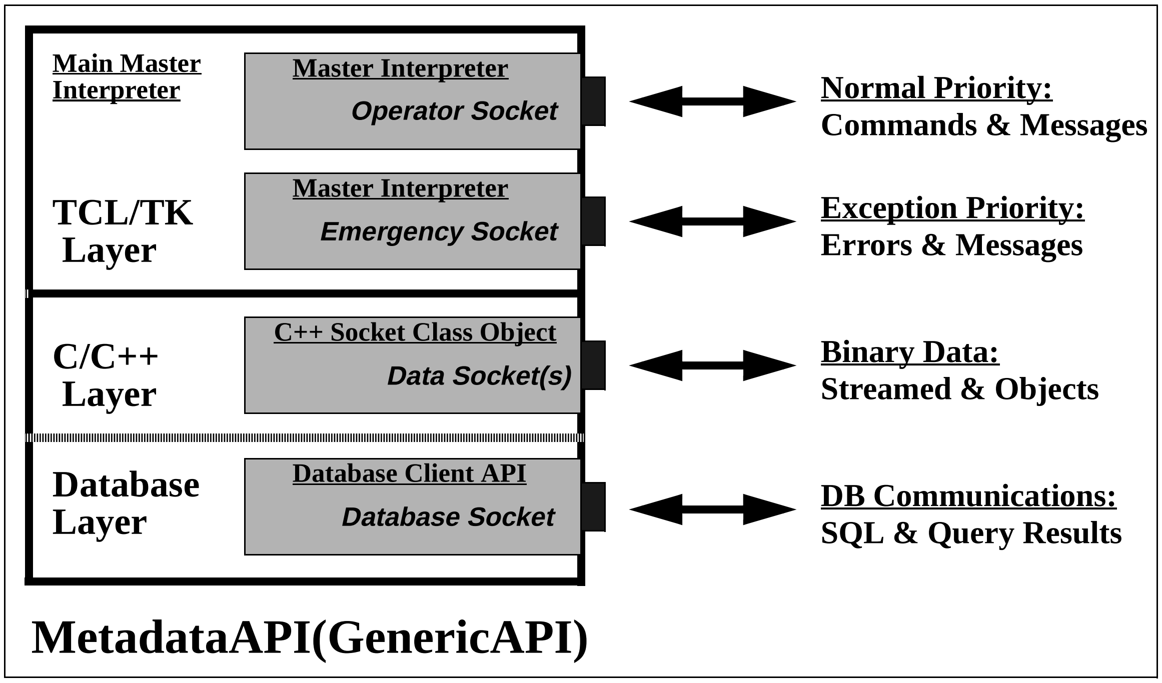
1. The LDAS distributed metadataAPI is made up of two major layers.
   a) TCL/TK Layer - this layer is the command layer and deals primarily with commands and/or messages and their attributes and/or parameters, as well as communicate with the underlying Package Layer through TCL/TK extensions.
   b) C/C++ Package Layer - this layer is the data engine layer and deals pri-

marily with the binary data and the algorithms and methods needed to manipulate LIGO's data and interface with the database server.

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 metadataAPI.tcl - this TCL/TK script contains specialized TCL/TK procedures and specialized command language extensions which are particular to the metadataAPI 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 metadataAPI.tcl script.
   c) The metadataAPI.rsc - this TCL/TK script contains the start-up and configuration defaults which are unique to the metadataAPI.
   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 metadataAPI.rsc file.
3. The C/C++ package layer consists of three 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 metadataAPI.so - this shared object contains the C++ classes and C interface functions needed to extend the command language set of each metadataAPI, allowing it to more efficiently manipulate of Internal Light Weight Format Data and provide Database Client functions.
   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 metadataAPI.so shared object directly.
   c) The database client API - this is the library used to develop a custom client, capable of communicating with the database server. This library will be linkable with the metadataAPI.so C/C++ code and will support all SQL statements associated with creating tables, inserts and updates on the tables, table migration, and queries in concert with the database server.
4. The database server will execute tasks on the LIGO Frame metadata tables as requested by the LDAS metadataAPI (which will be functioning as a custom client to the database server). The database server will manage the LIGO tables and be responsible for such routine maintenance as database table backups and other database management functions. The choice of platform for the database server and tables is not coupled to the platform being used for the metadataAPI (client). However, this platform should be able to effi-

ciently handle the order 400GB per year database expected for LIGO.

III. Communications Provided to MetadataAPI by GenericAPI



A. Socket Based Communications in MetadataAPI:

1. The genericAPI will provide the metadataAPI with an internet socket within the TCL/TK layer that is the primary communication port for commands and messages of a normal priority within the LDAS. 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 metadataAPI with dynamic internet data sockets within the C/C++ layer that are used to communicate all data (*typically binary data in the Internal Light-Weight Data Format*) 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.
3. The database client API development library will provide communications with the database server (using TCP/IP sockets) allowing the hardware/platform solutions for the LDAS metadataAPI to be decoupled from the database server hardware/platform solutions.

IV. Databases Tools:

A. Relational Database:

1. LDAS has studied the issues of database type to use. It has been decided that a relational database is sufficient for the tasks of supporting and managing LIGO’s metadata. Other types considered included object oriented databases such as ROOT (out of CERN) and the commercial Objectivity database but were found to be either inadequate or excessive for the needs of LIGO. This database will have to support of order 400GB of metadata per year.
2. In addition, LDAS has decided to pursue relational databases that support binary data types and user defined data structures in records. This will provided for additional flexibility in dealing with the dynamic understanding of the needs places on the database as understanding of LIGO’s dataset grows.

B. Commercial Database:

1. LDAS has studied the issues of commercial versus public domain databases. It has been decided to use a commercial relational database product for LIGO’s metadata. This is expected to reduce maintains requirements on the database tools in the future. As of this writing, the commercial database package being prototyped and expected to be the standard for LIGO’s metadata is IBM’s DB2 product. It is a relational database supporting binary data, and user defined data types. It is available for Windows NT, Solaris, HP and AIX at this time. The server software is much more affordable under Windows NT. Development of custom clients using the DB2 Software Developer’s Toolkit is affordable on all supported platforms, allowing the LDAS metadataAPI to be developed under Solaris while the database server runs on the less expensive Windows NT platform.
2. IBM has announced the availability of DB2 for Linux. A beta release is available to the public for testing as of this writing. Linux is expected to be a significant platform in the LDAS design, making DB2 under Linux an attractive option from IBM.

C. Open Database Connectivity (ODBC):

1. The metadataAPI.so layer will be developed using C++ classes which act as wrappers for the ODBC standardized application programmering interface. ODBC is an access methodology that enables applications, such as the metadataAPI, to seemlessly access heterogeneous databases. In addition to using ODBC, the metadataAPI.so C++ classes will support the following access handshake flow with the database server:
   a) Establish an *environment-handle*,
   b) establish a *database-connectivity-handle*,
   c) establish a *statement-handle*,

   d) perform a *validation*,
   e) carry out the necessary *binding* loop,
   f) *execute* the statement,
   g) *fetch* any results from a query,
   h) *close* all unneeded handles.

2. The metadataAPI.so layer will accept arbitrary SQL statements from the TCL/TK layer of the metadataAPI and using a single SQL interface, carry out the access handshake flow outlined above.

D. SQL Statements:

1. The arbitrary SQL statementes carried out by the metadataAPI.so layer and the database server will be enumerated as templates for creation by the TCL/TK layer of the metadataAPI. These SQL statements will be closely coupled to the following catagories of tables in the database:
   a) **Frame Description Tables** - includes frame names, locations, simple statistics such as first five moments of particular channels, spectra, and other TBD descriptions.
   b) **LDAS Event Tables** - includes descriptions of events produced by numerical filters in the LDAS, including astrophysical parameterizations, statistical confidences, etc.
   c) **GDS Trigger Tables -** includes descriptions of events produced by GDS trigger filters. These tables should have very similar design to the LDAS Event Tables.
   d) **E-Log Tables** - includes references to pictures and operator log entries from the CDS electronic log book (e-log) which are significate to LDAS.
   e) **IFO State Vector Tables** - includes references to the CDS database which contains IFO configuration, indexed by time-tags for epochs associated with each IFO state.
   f) **LDAS Log File Tables -** the log files maintained by the genericAPI layer in all LDAS API's have a TCL based database associated with them. These tables will support moving a subset of these log entries into the LDAS database server.
2. The SQL statements used to manipulate these tables will be cataloged as templates, having thier supporting fields identified as parameters to the statements. These statements could then be automatically generated by the TCL/TK command layer or entered directly by an LDAS operator/user.

V. 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.