An Enhanced Multi-Mission Calibration Database for High Energy Astrophysics

Ian N. Evans, Dale E. Graessle, X. Helen He, and Kenny J. Glotfelty
Smithsonian Astrophysical Observatory, 60 Garden Street, Cambridge, MA 02138

ABSTRACT

Calibration data products used by the processing pipelines and data analysis tools to calibrate observations obtained using the *Chandra X-ray Observatory* are maintained using a calibration database that complies with the CalDB standard developed by NASA's High Energy Astrophysics Science Archive Research Center. The goals of the CalDB, which was developed in the early 1990s, are to (1) allow separation of software upgrades and calibration upgrades, (2) allow multi-mission use of analysis software, and (3) facilitate the use of multiple software packages for the same data.

The instrumentation in use on the current generation of high energy astrophysics missions, such as the *Chandra X-ray Observatory*, are considerably more sophisticated than those of the preceding decades. As a result, the complexity of the requisite calibration data requires an indexing strategy that is not ideally matched to the relatively rigid structure of the existing CalDB.

We have developed, and are presently implementing, an enhanced design for the CalDB that generalizes the current structure and provides much greater flexibility in supporting sophisticated instrument models, while providing backwards compatibility with the current CalDB specification. While the explicit goal of this redesign is to support better the specific needs of the *Chandra X-ray Observatory*, the flexible structure will allow the CalDB to be adapted easily for use by future missions. Development of mission-independent analysis software is enhanced by providing a mechanism for software to structurally query the CalDB to ascertain the set of query parameters needed to completely identify a specific calibration data product.

1. INTRODUCTION

1.1. Overview of the *Chandra* CalDB

The *Chandra X-ray Observatory* is the most thoroughly calibrated space-based X-ray astronomy mission to date, and requires the most extensive and detailed calibration database of any X-ray mission, providing access to several hundred uniquely selectable calibration data products (CDPs). The CDPs are employed extensively by the *Chandra* standard data processing pipelines, and also by many user data analysis tools that are included in the *CIAO* data analysis package.

The current *Chandra* calibration database has been built in compliance with the HEASARC CalDB standard, which serves to:

- Store and archive CDPs;
- Maintain a naming convention and header structure for all CDPs;
- Index calibration data for software access, selectable based on FITS header keywords;
- Permit independent updates to calibration data and maintain configuration control;
- Provide traceable history of calibration data.

1.2. Limitations of the Current CalDB Implementation

Although calibration databases that comply with the HEASARC CalDB standard have been used successfully by many high energy astrophysics missions, certain fundamental assumptions and paradigms implicit in the existing CalDB standard have resulted in an over-complex and restrictive implementation as applied to the *Chandra* calibration database. Two fundamental specifications of the HEASARC CalDB standard are the rigid format of the calibration index file and the organization of CDPs on disk.

The HEASARC CalDB specification for the calibration index file (CIF) is a FITS format binary table that includes a defined set of columns used to select CDPs and identify their location for access. For any calibration type (*e.g.*, bad pixels, quantum efficiency, ...), selection of the appropriate CDP is performed based on the values recorded in the `TELESCOP`, `INSTRUME`, `DETNAM`, and `FILTER` columns, which record in the index the telescope (mission), instrument, detector, and filter to which the calibration applies. Choosing between time-dependent CDPs is based on the starting time when the calibration is valid, with the assumption that a "newer" CDP supersedes an "older" CDP. Any other selection criteria must be encoded in a set of up to nine 70-character string "boundary conditions."

For the *Chandra* mission, many CDPs can only be distinguished via the boundary conditions, since `FILTER` is meaningless for *Chandra*, and, for the ACIS instrument, `DETNAM` is not useful since there are numerous (thousands) of possible `DETNAM`s that share common CDPs. Selection criteria which would be meaningful for *Chandra*, such as "grating," "grating type," "mirror shell," and "observation mode" (to name a few) must be encoded as boundary conditions. The limit of 9 boundary conditions in the current CalDB specification has further complicated the indexing of some *Chandra* CDPs. Many *Chandra* CDPs (such as grating efficiencies) are either unrelated to the individual science instruments, or are relevant to data obtained using either of the science instruments. They must either be doubly indexed under each science instrument, or can be indexed via an alternative `INSTRUME` key. In the latter case, the calling software that is performing the CalDB query must be hard-coded to disregard the instrument information present in the science data products and use these alternatives when selecting specific types of CDPs.

The standard CalDB calibration data directory tree organizes CDPs according to instrument, on the assumption that each CDP applies only to a single instrument. However, for *Chandra* this is not the case, as described above. In a similar way to the CIF indexing problem, the instrument-based directory hierarchy either requires keeping multiple copies of the same CDP, or manual provision of links between directories, which complicates CalDB maintenance.

1.3. An Enhanced CalDB Implementation

Although the *Chandra* CalDB has until now been built upon the existing HEASARC standard, we have concluded that limited modifications to the CalDB specification would both significantly improve the maintainability of the *Chandra* CalDB, and provide the flexibility to support new and improved *Chandra* CDPs that will be needed for future developments. These improvements will also greatly enhance the ability of the CalDB to support new configurations and missions.

As a result, the *Chandra X-ray Center* has embarked on a new initiative to perform these enhancements, with the following goals:

- Modify the CalDB interface to allow for improved mission-independent software usage;
- Generalize the format of the CIF to be adjustable to the configuration and calibration requirements of specific missions;
- Support multiple CIFs, allowing each CIF to have a different format tuned to the configuration and calibration requirements of the indexed CDPs;
- Provide a flexible interface to query the generalized CIF;
- Allow software that is querying the CalDB to determine which calibration parameters (keywords and boundary conditions) must be specified in the query to uniquely identify a specific type of CDP;
- Provide backward compatibility with the existing missions represented in the HEASARC CalDB.

2. DESIGN OF THE ENHANCED CalDB

2.1. A Generalized Calibration Index File Format

For backwards compatibility, the generalized CIF maintains the FITS binary table format of the previous specification. However, instead of a rigidly defined set of index table columns, the enhanced CalDB splits columns into 3 categories: "mandatory," "optional," and "query."

Mandatory columns (see Table 1) must be present in any CIF, and are mostly related to the infrastructure necessary to locate the indexed CDP.

Table 1
Mandatory Index File Columns

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
`CAL_DEV`	string	`a20`		`"ONLINE"` for calibration files present in the CALDB directory tree. `"OFFLINE"` or `"<device name>"` for calibration files not present in the CALDB tree.
`CAL_DIR`	string	`a70`		The path to the directory containing the calibration file specified by `CAL_FILE`.
`CAL_FILE`	string	`a40`		The name of the calibration file.
`CAL_CNAM`	string	`a20`	`CCNM*`	The name of the type of calibration data. Used as the basis for identifying the type of data being queried.
`CAL_CBD`	string	`630a70`	`CBD*`	The calibration boundary conditions.
`CAL_XNO`	int	`i`		The FITS extension number that contains the calibration data (or zero for the primary HDU).
`CAL_QUAL`	int	`i`		The "quality" of the calibration data.
`CAL_DATE`	string	`a10`		The UTC date when this entry was added to the index file.

The remaining columns are defined in a human-readable "key configuration" file (see below for an example), which is new in the enhanced CalDB. For each non-mandatory CIF column, the key configuration file specifies the column name, data type, FITS format, equivalent FITS header keyword, null value, and whether the column is a query column or not. The first three items specify the appearance of the column in the CIF, and the remaining items are used to support query processing and building the CIF. The null value in the key configuration file is used by the CalDB query API library to determine which CIF columns are relevant when selecting a specific type of CDP.

The equivalent FITS header keyword entry in the key configuration file can be optionally used to map CIF column names to or from the corresponding FITS header keywords in user data products when performing a query, so that the calling software can deal only with header keywords if these differ from the CIF column names. For example, if the key configuration file specified that the column `TELESCOP` maps to the header keyword `MISSION`, then the CalDB query interface would indicate that the caller needs to supply the actual value of the header keyword `MISSION` whenever `TELESCOP` is referenced in the CIF. The equivalent FITS header keyword entry is similarly used to map CIF column names to CDP header keyword entries during CIF construction. During CIF construction, the equivalent FITS header keyword entry also provides a mechanism to allow a single CDP to be indexed in multiple ways.

The key configuration file supports reconfiguration of predefined CIF columns. For example, although the default format for the mandatory `CAL_CBD` column is `630a70`, which corresponds to a maximum of nine 70-character boundary conditions, the `CAL_CBD` column could be redefined in the key configuration file to have format `2450a70`, thus increasing the maximum number of boundary conditions to 35.

```
#
# Example key.config file excerpt
#
#colName   dataType   format      hdrKey     queryCol   nullVal
# ------------------------------------------------------------
...
CAL_CBD    string     2450a70     CBD*       no         “NONE”
TELESCOP   string     a10         TELESCOP   yes        ””
INSTRUME   string     a10         INSTRUME   yes        ”NONE”
DETNAM     string     a20         DETNAM     yes        ”NONE”
CCD_ID     int        i           CCD_ID     yes        -1
GRATING    string     a10         GRATING    yes        ”NONE”
GRATTYPE   string     a10         GRATTYPE   yes        ””
SHELL      string     a4          SHELL      yes        “”
...
```

All of the optional columns (see Table 2) have predefined meanings, and several are provided solely for reasons of backwards compatibility. Others support CDP selection based on the temporal validity or fidelity of the calibration data; however they may not be relevant for all CalDB configurations and are thus excluded from the mandatory set. Calibration boundary conditions also fall into this category.

Table 2
Predefined Index File Optional Columns

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
`CAL_CLAS`	string	`a3`	`CCLS*`	The calibration file class. Not interpreted by software.
`CAL_DTYP`	string	`a4`	`CDTP*`	The type of the calibration file. Not interpreted by software.
`CAL_DESC`	string	`a70`	`CDES*`	Short string description of the calibration data. Not interpreted by software.
`CAL_VSD`	string	`a10`	`CVSD*`	The beginning UTC valid date of the calibration.
`CAL_VST`	string	`a8`	`CVST*`	The beginning UTC valid time of the calibration.
`REF_TIME`	double	`d`		The MJD corresponding to `CAL_VSD`/`CAL_VST`.
`CAL_VED`	string	`a10`	`CVED*`	The ending UTC valid date of the calibration.
`CAL_VET`	string	`a8`	`CVET*`	The ending UTC valid time of the calibration.
`END_TIME`	double	`d`		The MJD corresponding to `CAL_VED`/`CAL_VET`.
`FIDELITY`	double	`d`	`FDLT*`	Numeric "fidelity" of calibration data.

The majority of the CIF columns on which CDP selection is performed are query columns. These columns can have arbitrary names and formats appropriate to the CalDB being defined, and are the primary means of distinguishing different CDPs in the enhanced CalDB.

2.2. Enhanced CalDB Query API Library

The enhanced CalDB query API library provides a two level query mechanism for selecting CDPs. The first-level query takes only the type of the CDP being searched, and optionally `TELESCOP` and `INSTRUME` to limit the search. Each CIF is read using the appropriate key configuration file information, and then the CIF is queried for all entries of the appropriate type (*i.e.*, that have matching values of `CAL_CNAM`). The contents of all query columns for the matching subset are then compared with the corresponding null values in the key configuration file. Columns that contain only null values in the matching subset are not relevant for the selection. The names of all other columns are returned to the caller to specify what information is required to perform the second level query, possibly after translation from CIF column names to equivalent FITS header keyword names. Boundary condition constraints are parsed into their components and the parameter names are returned to the caller. If the caller already knows which query parameters to specify, and how to specify them, the first level query can be skipped.

Subsequently, the caller looks up the appropriate values for the required parameters from the data undergoing calibration (typically by extracting observation meta-data from the FITS science instrument data file headers), and makes a second level query specifying all of the relevant parameters and actual values. If actual values are not specified for any required parameters, then they are populated with the null values from the key configuration file. The query interface selects all CIF records that match the actual values provided, and, if appropriate, checks calibration validity times versus the actual times provided, selects CDPs based on quality and fidelity criteria, and then returns the pointers to the matching CDPs to the caller. Actual access to the CDPs is the responsibility of the caller.

Example

An example, consider the following query to select the appropriate *Chandra* grating efficiency (`“GREFF”`) CDP using the existing CalDB implementation, and compare this to the enhanced CalDB implementation (see below).

In the *existing CalDB*, the calling software would need to specify the following parameters:

```
CAL_CNAM = “GREFF”
TELESCOP = “CHANDRA”
INSTRUME = “TEL”
DETNAM   = “GRATING”
FILTER   = “”
CAL_CBD  = “GRATING.EQ.HETG.AND.GRATTYPE.EQ.MEG.AND.SHELL.EQ.1000”
```

Although this example seems to be relatively straightforward, it highlights several complexities imposed by the existing CalDB implementation. Since the gratings on-board *Chandra* are inserted into the X-ray optical path ahead of the cameras, the grating calibration data products are independent of the choice of instrument and detector. Therefore, the calling software that is performing the query must be coded to ignore the actual contents of the `INSTRUME` and `DETNAM` header keywords in the science data products, and substitute the actual values `“TEL”` and `“GRATING”` whenever a grating calibration related query is performed. The `FILTER` keyword is irrelevant for *Chandra*, but must nevertheless be included in both the CIF and the query. The relevant selection criteria for a grating efficiency query — `GRATING`, `GRATTYPE`, and `SHELL` — can only be encoded as boundary conditions in the CIF and must be selected by constructing the appropriate query string, as shown.

Compare this to the query using the *enhanced CalDB*. The first level query would simply identify the type of CDP required:

```
CAL_CNAM = “GREFF”
```

The CalDB query API library would then search the CIFs for all rows that have `CAL_CNAM = “GREFF”`, compare the entries in each row with the null values specified in the key configuration file, and identify only those columns that have non-null entries as necessary input parameters for the second level query. If requested by the caller, the query library would translate the column names into the equivalent FITS header keywords, using the information in the key configuration file. With the sample key configuration file above, and the excerpt from the sample CIF from Table 3, the query API would return the list `TELESCOP`, `GRATING`, `GRATTYPE`, and `SHELL` to the caller. `CCD_ID` will not be included in the return, since all of the rows that match `CAL_CNAM = “GREFF”` have `CCD_ID = -1`, which is defined as the null value in the key configuration file.

Subsequently, the caller would look up the appropriate actual values for the call (for example, by querying the FITS science data product headers), and perform the second level query supplying the actual values:

```
CAL_CNAM = “GREFF”
TELESCOP = “CHANDRA”
GRATING  = “HETG”
GRATTYPE = “MEG”
SHELL    = “1000”
```

Once all of the CIF rows that correspond to the specified actual values are identified, calibration start and end valid times, quality, and fidelity (all if specified in both the CIF and the query) are checked, and matching CDPs are selected to be returned to the caller.

The flow through the two-level query to the enhanced CalDB is represented graphically in Figure 1.

Table 3
Sample Calibration Index File Excerpt

CAL_CNAM	TELESCOP	INSTRUME	DETNAM	CCD_ID	GRATING	GRATTYPE	SHELL
“QE”	“CHANDRA”	“ACIS”	“NONE”	0	“NONE”	“”	“”
“DEGAP”	“CHANDRA”	“HRC”	“HRC-I”	-1	“NONE”	“”	“”
“GREFF”	“CHANDRA”	“NONE”	“NONE”	-1	“HETG”	“HEG”	“0001”
“GREFF”	“CHANDRA”	“NONE”	“NONE”	-1	“HETG”	“MEG”	“1000”
“GREFF”	“CHANDRA”	“NONE”	“NONE”	-1	“HETG”	“MEG”	“0100”
...	...	...	...	...	...	...	...

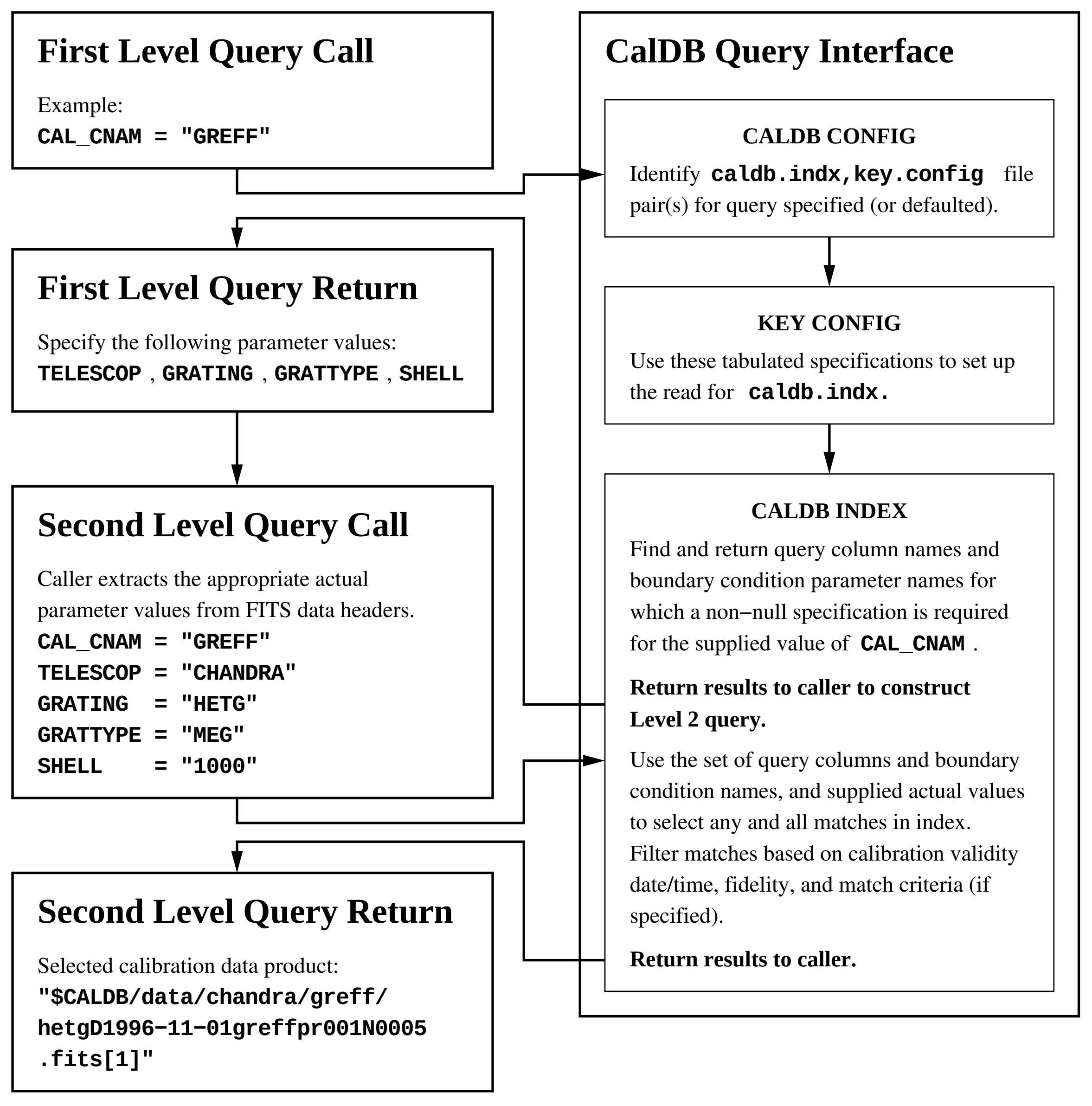
First Level Query Call
Example:
CAL_CNAM = "GREFF"
First Level Query Return
Specify the following parameter values:
TELESCOP , GRATING , GRATTYPE , SHELL
Second Level Query Call
Caller extracts the appropriate actual parameter values from FITS data headers.
CAL_CNAM = "GREFF"
TELESCOP = "CHANDRA"
GRATING = "HETG"
GRATTYPE = "MEG"
SHELL = "1000"
Second Level Query Return
Selected calibration data product:
"$CALDB/data/chandra/greff/hetgD1996-11-01greffpr001N0005.fits[1]"
CalDB Query Interface
CALDB CONFIG
Identify caldb.indx,key.config file pair(s) for query specified (or defaulted).
KEY CONFIG
Use these tabulated specifications to set up the read for caldb.indx.
CALDB INDEX
Find and return query column names and boundary condition parameter names for which a non–null specification is required for the supplied value of CAL_CNAM .
Return results to caller to construct Level 2 query.
Use the set of query columns and boundary condition names, and supplied actual values to select any and all matches in index.
Filter matches based on calibration validity date/time, fidelity, and match criteria (if specified).
Return results to caller.

Figure 1: Example schematic two-level query interface used in the enhanced CalDB.

2.3. CIF Construction

Construction of the CIF is performed automatically by a tool that creates the CIF in accordance with the specification in the key configuration file, reads the headers of all of the CDPs present in a user-specified directory (sub-)tree, and populates the rows of the CIF based on the header contents. Automatic construction simplifies CIF creation and ensures that the CIF, key configuration file, and the CDPs in the CalDB directory tree, are internally consistent.

If specified by the user, translation between FITS header keywords and column names is performed according to the contents of the key configuration file. CIF columns that do not have corresponding entries in the FITS header are populated with the null value. Population of predefined optional columns that specify calibration validity date/times is handled specially by the CIF construction tool to ensure consistency between the date/time and MJD column values.

CDPs can be indexed multiple times in the CIF, with different selection parameters, by using the "keyset" mechanism. If the key configuration file specifies that the equivalent FITS header keyword for a column is of the form `“NAME*”`, where `NAME` is a valid FITS keyword string with a maximum of 4 characters (*e.g.*, `“CVSD*”`), then the tool will look for all header keywords of the form `“NAME00`*`nn`*`”`, where *`nn`* is a two digit integer, that have corresponding `CAL_CNAM` entries (typically `“CCNM00`*`nn`*`”`), and create rows in the CIF that index the CDP with each distinct set of parameters.

2.4. Future Enhancements

Although the software implementation of the key functionality of the enhanced CalDB is well underway (see below), further enhancements to the requirements are still needed in the areas of version management, support for time systems other than UTC, and managing related files. Management of related files, for which the selection of a CDP depends on which other CDPs were previously applied during the calibration of a science data product, is the most significant enhancement that is being considered, and also the most complex.

3. SOFTWARE IMPLEMENTATION

3.1. Software Implementation

We have developed a query library API that can be called to perform the first and second level queries to the enhanced CalDB, and have developed a tool that will extract the appropriate header information from the CDPs in the CalDB directory tree, as defined by the key configuration file, to automatically build the CIF. Ancillary tools that support validating, merging, and editing CIFs will follow. The library and associated tools are written in C++ and use the HEASARC *cfitsio* package for FITS file access. They are ported to the Sun Solaris operating system, several flavors of Linux, and Mac OS-X. We plan to make available C, Python, and S-Lang wrappers to the query library API.

3.2. Query API Example

The following example demonstrates how to access the enhanced CalDB to perform a two-level query. For simplicity, the code fragment does not include any error checking. However in practice each of the query library API routines will return a NULL value or error status as appropriate if an error condition occurs. The example opens the *Chandra* CalDB, performs a first level query for all grating efficiency (`“GREFF”`) CDPs, determines the actual values corresponding to each parameter returned by the first level query from the science data products using the caller's routine `lookupKeyInScienceDataHeader`, performs the second level query with the actual parameter values, and prints the complete path to the calibration file and the FITS extension number of the CDP.

```
#include "caldb4.h"
#include "stdio.h"
int main(void) {
calCALDB* myCaldb = calInit( /*TELESCOP*/ “CHANDRA”, /*INSTRUME*/ NULL );
calSEARCH* mySearch = calSetProduct( myCaldb, /*CAL_CNAM*/ “GREFF” );
int nPars = calGetWhichParams( mySearch );      /* Perform 1st level query */
for ( int ii = 0; ii < nPars; ii++ ) {
  char* param = calGetParam( mySearch, ii, /*Translate keywords*/ calTRANSTrue );
  /* Lookup up actual value for the query in science data header */
  char* value = lookupKeyInScienceDataHeader( param );
  calSetParam( mySearch, param, value, /*unit*/ “” );
  }
calSetMatchMode( mySearch, calMATCHFirstWarn ); /* Return 1st CDP, warn if more */
int nFound = calSearch( mySearch );             /* Perform 2nd level query */
char* fileFound = calGetFile( mySearch, /*1st file*/ 0 );
printf( “%s\n”, fileFound );
calClose( myCaldb );
return 0;
}
```

ACKNOWLEDGEMENTS

Support for development of the enhanced CalDB is provided by the National Aeronautics and Space Administration through the *Chandra X-ray Center*, which is operated by the Smithsonian Astrophysical Observatory for and on behalf of the National Aeronautics and Space Administration under contract NAS 8-03060.