CALDB 4.0 Requirements

Version of March 21, 2007

I. DEFINITIONS

(1) Index File Column Definitions

(a) The columns contained in an index file fall into one of 3 categories: *"mandatory"*, *"optional"*, or *"query"*.

(i) *Mandatory* columns have predefined meanings and must be present in any index file. They are used by CALDB software to record and extract information about calibrations from the index file.

1. Mandatory columns do not include columns used to perform CALDB queries.

2. Table 1 defines the list of mandatory columns.

3. The fully qualified path to a calibration file is determined from the contents of an index file record as follows:

a. If `CAL_DEV = "ONLINE"`, then the fully qualified path to the calibration file is `"{$CALDB}/{$CAL_DIR}/{$CAL_FILE}"`.

b. If `CAL_DEV = "OFFLINE"`, then the fully qualified path to the calibration file is `"OFFLINE {$CAL_DIR}/{$CAL_FILE}"` and the file is not available for access.

c. If `CAL_DEV = "<device>"`, then the fully qualified path to the calibration file is `"<device>/{$CAL_DIR}/{$CAL_FILE}"`.

Note

`<device>` is not *restricted to being a locally accessible disk device, but may also be a Uniform*

Resource Locator (URL; for example, "http://www.calibdata.org" or "ftp://ftp.calibdata.org") to support indexing of network-accessible calibration data.

d. In subparts a–c above, if {$CAL_DIR} translates to the empty (null) string, then the trailing "/" following {$CAL_DIR} is discarded.

4. A key configuration file [see section I.(2)] defines the column format of an index file, and may be used to redefine certain properties of mandatory columns.

a. The data type cannot be redefined between string and numeric types. The format cannot be redefined to be inconsistent with the data type (*e.g.*, numeric data type with string format). Any recognizable attempt to do so shall generate a warning and the redefinition shall be ignored.

Note

The intent of this requirement is primarily to allow the format or FITS header keyword equivalents to be modified.

(ii) *Optional* columns may be present in an index file, but are not required.

1. Optional columns do not include columns used to perform CALDB queries.

2. Optional columns may have predefined meanings, and may be interpreted by software in a predefined manner.

3. Table 2 defines the list of predefined optional columns.

a. The data type of predefined optional columns cannot be redefined between string and numeric types. The format cannot be redefined to be inconsistent with the data type (*e.g.*, numeric data type with string format). Any recognizable attempt

to do so shall generate a warning and the redefinition shall be ignored.

(iii) *Query* columns are those columns that are used to perform matches against actual query parameters supplied by the calling application.

1. Only query columns will be returned to the application in response to a first level query [see section III].

(b) Optional and query columns contained in an index file are defined in a "key configuration" file.

(i) Except in the case of redefinitions in accordance with section I.(1).(a).(i).4, mandatory columns are not included in the key configuration file since they are required to be present in any index file.

(ii) A default key configuration file shall be provided that includes the appropriate definitions to define a backwards-compatible HEASARC version 1.1 calibration index file.

Table 1

Mandatory Index File Columns

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
CAL_DEV	string	char20		"ONLINE" for calibration files present in the CALDB directory tree. "OFFLINE" or "<device name>" for calibration files not present in the CALDB tree. See below for information on constructing the fully qualified path to the calibration file from CAL_DEV, CAL_DIR, and CAL_FILE.
CAL_DIR	string	char70		The path to the directory containing the calibration file specified by CAL_FILE.
CAL_FILE	string	char40		The name of the calibration file.
CAL_CNAM	string	char20	CCNM*	The name of the type of calibration data. Used as the basis for identifying the type of data being queried.
CAL_CBD	string	char70[9]	CBD*	The calibration boundary conditions. These are additional query constraints that are either sufficiently infrequent that including a query column in the index file is inappropriate, or that are not readily represented as query columns in the index file (examples would be numeric ranges).
CAL_XNO	int	int		The FITS extension number that contains the calibration data. May be zero if the calibration data are stored in the primary HDU (for example, an image).
CAL_QUAL	int	int		The "quality" of the calibration, defined as follows: 0 = "Good. No known issues."; 1 = "Some calibration issues known. May be default for processing or analysis."; 2 = "Calibration

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
				issues identified and replacement needed or anticipated. May be used, but will be superseded in the near future."; 3 = "Former operational mode, good enough for processing or analysis of archival data taken in that mode. Significant issues known; however recalibration of this mode is unlikely."; 4 = "Strongly deprecated or superseded. Issue a warning to this effect if used. Reprocessing needed if these data were used in earlier processing; same for earlier analysis. Kept to enable selection of these data as related files."; 5 = " Bad. Never to be used again. To be retired. A better option is available in the CalDB, including related files data."; -9 = "Dataset no longer exists.".
CAL_DATE	string	char10		The UTC date when this entry was added to the index file, in `yyyy-mm-dd` format.

Table 2

Predefined Index File Optional Columns

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
CAL_CLAS	string	char3	CCLS*	The calibration file class, one of "PCF", "BCF", "CPF", or "USR". Not interpreted by software.
CAL_DTYP	string	char4	CDTP*	The type of the calibration file, one of "DATA", "TASK", or "FEF". Not interpreted by software.
CAL_DESC	string	char70	CDES*	Short string description of the calibration data. Not interpreted by software.
CAL_VSD	string	char10	CVSD*	The beginning valid date of the calibration, in yyyy-mm-dd format. This column is interpreted by software.
CAL_VST	string	char8	CVST*	The beginning UTC valid time of the calibration, in hh:mm:ss format. This column is interpreted by software.
REF_TIME	double	double		The MJD corresponding to CAL_VSD/CAL_VST. This column is interpreted by software.
CAL_VED	string	char10	CVED*	The ending UTC valid date of the calibration, in yyyy-mm-dd format. This column is interpreted by software.
CAL_VET	string	char8	CVET*	The ending UTC valid time of the calibration, in hh:mm:ss format. This column is interpreted by software.
END_TIME	double	double		The MJD corresponding to CAL_VED/CAL_VET. This column is interpreted by software.
FIDELITY	double	double	FDLT*	Numeric "fidelity" of calibration data. May be used to select amongst different calibrations of the same quality that satisfy the query requirements. The ordering of the FIDELITY is defined such

Index File Column Name	Data Type	Format	Equivalent Header Keyword	Description
				that numerically greater values correspond to higher fidelity calibration data. This column is interpreted by software.

(2) Key Configuration File Definition

(a) A key configuration file defines the column contents of an index file, and includes the information needed to build and query an index file.

(b) The file format should be human readable and provide support for inclusion of comments. A suggested implementation format is presented in the example below, but this format is not required.

(c) The following key configuration file columns are defined:

(i) Index file column name,

(ii) Data type (`int`, `long`, `float`, `double`, `string`, `bool`),

(iii) Index file column/calibration file header keyword format,

(iv) Equivalent calibration file FITS header keyword name (if applicable; used to populate the index file column),

1. If the keyword terminates with a `"*"` (*e.g.*, `"CCNM*"`), then the keyword and corresponding index file column belong to the standard *keyset*. This information is used when constructing the index file.

(v) Whether the column is a query column,

(vi) The null value.

Example key configuration file

In this example, the mandatory index file column CAL_CBD *is redefined to allow up to 35 boundary conditions each 70 characters long; the optional index file columns* CAL_CLAS, CAL_DTYP, CAL_DESC, *and* FIDELITY *are defined; and query columns* TELESCOP, INSTRUME, DETNAM, CCD_ID, GRATING, *and* GRATTYPE *are defined. The* CAL_xxx *index file columns have FITS header keyword equivalents of the form* key* *which implies that they are part of standard* keyset.

```
#
# Example key.config file
#
#colName   dataType   format       hdrKey     queryCol   nullVal
# ------------------------------------------------------------
CAL_CBD    string     2450a70      CBD*       no         ″NONE″
CAL_CLAS   string     a3           CCLS*      no         ″BCF″
CAL_DTYP   string     a4           CDTP*      no         ″DATA″
CAL_DESC   string     a70          CDES*      no         ″″
CAL_VSD    string     a10          CVSD*      no         ″″
CAL_VST    string     a8           CVST*      no         ″″
REF_TIME   double     d            ″″         no         0.0
FIDELITY   double     d            FDLT*      no         0.0
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        ″″
```

(3) CALDB Configuration File Definition

(a) The CALDB configuration file includes the information needed to locate an index file (or index files) to be queried.

(b) The file format is human readable and provides support for inclusion of comments.

(c) If the environment variable "CALDBCONFIG" is defined, then the fully qualified path to the CALDB configuration file shall be "{$CALDBCONFIG}". If the environment variable "CALDBCONFIG" is *not* defined, then the fully qualified path to the CALDB configuration file shall be "{$CALDB}/caldb.config".

(i) The environment variable "CALDB" specifies the *root* of the CALDB tree. This environment variable must exist and refer to a locally accessible disk device.

(d) The format of the CALDB configuration file is defined as follows:

(i) Each line shall be either:

1. A blank line, which shall be ignored; or
2. A comment line, which shall begin with a "#" character, and shall be ignored; or
3. A telescope/instrument specification.

(ii) A telescope/instrument specification shall consist of seven or ten tokens, separated by white space, in the following order:

```
TELESCOP INSTRUME INDXDEV INDXDIR INDXFIL
CAL_DEV CAL_DIR [KEYDEV KEYDIR KEYFIL]
```

1. TELESCOP is the UPPERCASE string specifying the telescope (mission), and which corresponds to the contents of the TELESCOP index file column (if any), or "DEFAULT".
2. INSTRUME is the UPPERCASE string specifying the telescope (mission), and which corresponds to the contents of the INSTRUME index file column (if any), or "DEFAULT".

3. INDXDEV, INDXDIR, and INDXFIL define the fully qualified path to the index file for the specified telescope/instrument combination.

   a. If INDXDEV = "CALDB", then the fully qualified path to the index file is "{$CALDB}/<INDXDIR>/<INDXFIL>", where <INDXDIR> and <INDXFIL> are the values of INDXDIR and INDXFIL extracted from the CALDB configuration file.

   b. If INDXDEV != "CALDB", then the fully qualified path to the index file is "<INDXDEV>/<INDXDIR>/<INDXFIL>".

4. CAL_DEV and CAL_DIR must be present for reasons of backwards compatibility, but are not used by the software and should be ignored. (In earlier versions of the CALDB software they were used to identify the locations of calibration data products when constructing the index file.)

5. KEYDEV, KEYDIR, and KEYFIL are optional tokens that define the fully qualified path to the key configuration file that defines the format of the index file for this telescope/ instrument combination.

   a. If KEYDEV = "CALDB", then the fully qualified path to the key configuration file is "{$CALDB}/<KEYDIR>/<KEYFIL>", where <KEYDIR> and <KEYFIL> are the values of KEYDIR and KEYFIL extracted from the CALDB configuration file.

   b. If KEYDEV != "CALDB", then the fully qualified path to the index file is "<KEYDEV>/<KEYDIR>/<KEYFIL>".

   c. If KEYDEV, KEYDIR, and KEYFIL are not specified, then the values of KEYDEV and KEYDIR default to the values of INDXDEV and INDXDIR, respectively, and the value of KEYFIL defaults to

"key.config".

(e) The list of index files that are the subject of a query is determined by matching records in the CALDB configuration file, according to the following rules:

(i) The caller may optionally provide actual values corresponding to the TELESCOP and INSTRUME index file column values to be queried.

(ii) If an input value (TELESCOP and/or INSTRUME) is specified:

1. Search for exact matches for that input in the CALDB configuration file.

2. Search for "DEFAULT" matches for that input in the CALDB configuration file.

3. The list of index files to be queried comprises the union of the set of exact matches and the set of default matches. If no exact matches or default matches are found, then return an error.

(iii) If an input value is *not* specified:

1. Search for "DEFAULT" matches for that input in the CALDB configuration file.

2. The list of index files to be queried comprises the set of default matches. If no default matches for that input are found, then *all* entries are considered as being matches for that input.

Example CALDB configuration file

In this example, the ROSAT index file locations and calibration data product directory trees are configured in a manner that is compatible with a HEASARC version 1.1 CALDB. The first Chandra line will match a query with `TELESCOP = "CHANDRA"` *and any value for* `INSTRUME`*, including the case where* `INSTRUME` *is not specified. The final line will match a query that does not specify either* `TELESCOP` *or* `INSTRUME`*.*

```
# Example caldb.config file
#
# ROSAT
#
ROSAT HRI CALDB data/rosat/hri caldb.indx CALDB data/rosat/hri
ROSAT PSPCB CALDB data/rosat/pspc caldb.indx CALDB data/rosat/pspc
ROSAT PSPCC CALDB data/rosat/pspc caldb.indx CALDB data/rosat/pspc
ROSAT XRT CALDB data/rosat/xrt caldb.indx CALDB data/rosat/xrt
#
# CHANDRA
#
CHANDRA DEFAULT CALDB data/chandra caldb.indx CALDB data/chandra
     CALDB data/chandra key.config
DEFAULT DEFAULT CALDB data/chandra caldb.indx CALDB data/chandra
     CALDB data/chandra key.config
```

Example CALDB configuration file searches

Continuing the above example, a search with `TELESCOP = "ROSAT"` *and* `INSTRUME = "HRI"` *would match the first record of the file. A search with* `TELESCOP = "ROSAT"` *and* `INSTRUME` *not specified would first search for a record beginning* `"ROSAT DEFAULT"` *(which would fail), and would ultimately return all five* `ROSAT` *records. A search with* `TELESCOP = "CHANDRA"` *and* `INSTRUME` *either set to any value* or *not specified would match the record beginning* `"CHANDRA DEFAULT"`. *If neither* `TELESCOP` *nor* `INSTRUME` *were specified, then the last record (beginning* `"DEFAULT DEFAULT"`*) would match. If the* `"DEFAULT DEFAULT"` *record were not present, however, then all CALDB configuration file records would match by virtue of I.(3).(e).(iii).2.*

(4) CALDB Aliases File Definition

(a) The CALDB aliases file is an optional file that includes the information needed to translate instrument aliases into the corresponding instrument values that will be used to query an index file (or index files).

(i) The CALDB aliases file is provided for backwards compatibility with HEASARC version 1.1 calibration database index files. Use of a CALDB aliases file with a version 2.0 calibration database index files is deprecated.

(b) The file format is FITS with one or more binary table extensions.

(c) If the environment variable `"CALDBALIAS"` is defined, then the fully qualified path to the CALDB aliases file shall be `"{$CALDBALIAS}"`. If the environment variable `"CALDBALIAS"` is *not* defined, then the CALDB aliases file is not present, and instrument alias translation shall not be performed.

(d) The format of the CALDB aliases file is defined as follows:

(i) The FITS primary HDU is not used.

(ii) Each FITS binary table extension HDU shall include the instrument aliases for a single telescope. More than one FITS binary table extension HDU may include aliases for the same telescope.

1. The telescope name (which corresponds to the contents of the `TELESCOP` index file column) is encoded as the contents of the `EXTNAME` keyword in the header of the FITS binary table extension HDU.

2. The instrument aliases for the telescope are encoded, one per row, in the data portion of the FITS binary table extension HDU.

a. The instrument alias is encoded as an ASCII string and recorded in the FITS binary table column `'ALIAS   '`.

b. The number of translation values for the alias is

encoded as an integer and recorded in the FITS binary table column 'ALIAS_NO'.

c. The translation values for the alias are encoded as ASCII strings and recorded in the FITS binary table column 'VALUES ', using the FITS binary table "substring array" convention for fixed-length substrings.

(e) If the CALDB aliases file is present, then instrument alias translation is performed whenever as part of an index file query whenever actual input values are supplied for *both* the TELESCOP and INSTRUME index file columns (after optional translation of the supplied parameter names from their FITS header keyword equivalents, if specified by the caller). The instrument alias translation is performed as follows:

(i) Identify the set of FITS binary table extension HDUs for which the EXTNAME header keyword values match the actual TELESCOP parameter value provided by the caller. Remove any trailing blanks from each value prior to performing the string comparisons.

1. If no HDUs have matching EXTNAME header keyword values, then no aliases are present and the actual TELESCOP and INSTRUME parameter value provided by the caller are used to perform the index file query.

(ii) If one or more HDUs have matching EXTNAME header keyword values, then identify the set of FITS binary table rows in those HDUs for which the contents of the string column 'ALIAS ' matches the actual INSTRUME parameter value provided by the caller. Remove any trailing blanks from each value prior to performing the string comparisons.

1. If no rows have matching 'ALIAS ' column values, then no aliases are present and the actual TELESCOP and INSTRUME parameter value provided by the caller are used to perform the index file query.

(iii) From each matching row, extract the number of translation values, *n*, for the instrument alias from the column 'ALIAS_NO'. Extract *n* translation values for the instrument alias from the string

column `'VALUES  '`. Remove any trailing blanks from each extracted translation value, and identify the set of unique extracted translation values.

(iv) Substitute the set of unique extracted translation values for the actual `INSTRUME` parameter value provided by the caller to complete the index file query.

II. QUERY INTERFACE: OVERVIEW

(1) The query interface supports two levels of query.

(a) A *first level query* takes as input a `CAL_CNAM` value, and returns a list of query parameters and boundary conditions that must be supplied to ensure a successful second level query.

(b) A *second level query* takes as input the actual parameter values (corresponding to the list of query parameters and boundary conditions returned by the corresponding first level query), and returns a list of matching calibration file/extension pairs.

(i) The caller may choose to execute only the second level query with the actual parameter values if the list of query parameters and boundary conditions is known.

(ii) The caller is not required to provide actual parameters corresponding to all query parameters and boundary conditions returned in a first level query. The appropriate null values from the key configuration file will be substituted for any actual parameters that are not supplied.

III. QUERY INTERFACE: FIRST LEVEL QUERY

(1) Input specification:

(a) The caller shall specify as input the value of `CAL_CNAM` to be queried.

(b) Optionally, the caller may specify as input values of `TELESCOP` and `INSTRUME` to be used to limit the set of index files to be searched.

(c) Optionally, the caller may specify as input that returned index file column names should be translated into their FITS header keyword equivalents using the information in the key configuration file.

(2) Return values:

(a) The query interface shall return to the caller (a) a list of the index file column names (optionally translated into their FITS header keyword equivalents), and (b) a list of calibration boundary condition names and corresponding units, that should be specified in a second level query for the given `CAL_CNAM`.

(3) Algorithm:

(a) The set of index files to be searched is determined from the CALDB configuration file by performing steps I.(3).(e), using the optional `TELESCOP` and `INSTRUME` values [see III.(1).(b)] if provided as input.

(i) If the optional `TELESCOP` and `INSTRUME` values [see III.(1).(b)] were provided as input, and the CALDB aliases file is present [see I.(4).(c)], then perform instrument alias translation as specified in I.(4).(e) above.

(b) For each index file identified in step III.(3).(a):

(i) Identify the set, *r*, of index file records that have values in the `CAL_CNAM` column that match the actual parameter value provided by the caller.

Note

When matching string values is called for in this document, any leading or trailing white space shall be ignored.

(ii) Identify the index file columns that are listed as query columns in the key configuration file, and for which at least one member of the set, *r*, of index file of records identified in III.(3).(b).(i) above has a value that is not the null value for the column (specified in the key configuration file).

1. If specified by the caller, translate the index file column names into their FITS header keyword equivalents using the translations recorded in the key configuration file.

   a. If no equivalent header keyword is defined (*i.e.*, the entry is null (`""`), or if the equivalent header keyword is of the form `"NAME*"` (implying it comprises one of the keyset keywords/columns), then the index file column name should be used without translation.

(iii) Extract unique (*name*, *units*) pairs for any non-null (*i.e.*, non-`"NONE"`) boundary conditions recorded in the `CAL_CBD` index file column specified for any members of the set, *r*, of index file of records identified in III.(3).(b).(i) above.

1. Boundary conditions are recorded using the syntactic form *name*(*condition*[,…])[*units*].

   Example boundary conditions:

   ```
   FP_TEMP(150.0-170.0)K
   CCD_ID(0)
   OBSMODE("RASTER","SCAN")
   ```

2. The maximum number and string size for boundary conditions can be changed from the default values (9 boundary conditions, each of size `char70`) though redefinition of the `CAL_CBD` column via the key configuration file.

(c) Return to the calling application the list of unique index file column names (possibly translated to FITS header keyword equivalents) concatenated from all of the index file queries from III.(3).(b).(ii) above, and the list of unique boundary condition (*name*, *units*) pairs from III.(3).(b).(iii) above.

IV. QUERY INTERFACE: SECOND LEVEL QUERY

(1) Input specification:

(a) The caller shall specify as input the value of `CAL_CNAM` to be queried.

(b) Optionally, the caller may specify as input values of `TELESCOP` and `INSTRUME` to be used to limit the set of index files to be searched.

(c) Optionally, the caller may specify as input the UTC date and time for which the calibration is requested.

(i) The date and time may be specified either as two separate arguments (strings in `yyyy-mm-dd` and `hh:mm:ss` formats), or may be specified as a single argument (string in `yyyy-mm-ddThh:mm:ss` format).

(d) Optionally, the caller may specify as input a list of parameter names and corresponding actual parameter values that define the query. The appropriate parameter names are typically returned to the caller by a preceding first level query.

(i) Each parameter name maps to either an index file column name, or is a boundary condition name.

(ii) Optionally, the caller may specify as input whether the supplied parameter names should be translated from their FITS header keyword equivalents into index file column names using the key configuration file.

(e) Optionally, the caller may specify as input which calibrations to select based on calibration fidelity, if the optional index file column `FIDELITY` is defined in the key configuration file.

(f) Optionally, the caller may specify as input the action to be taken if more than one calibration matches the query parameters.

(g) Optionally, the caller may specify as input that calibration quality not be considered when selecting calibrations.

(h) Optionally, the caller may specify as input that the calibration quality (`CAL_QUAL`) values corresponding to each matching calibration be returned to the caller.

(i) Optionally, the caller may specify as input that the calibration fidelity (FIDELITY) values corresponding to each matching calibration be returned to the caller, if the optional index file column FIDELITY is defined in the key configuration file.

(2) Return values:

(a) The query interface shall return to the caller the fully qualified paths [see section I.(1).(a).(i).3] and extension numbers for the calibrations that match the actual query, as determined by the algorithm in IV.(3) below.

(b) If specified by the caller [see IV.(1).(h)], the query interface shall return to the caller the calibration quality (CAL_QUAL) values for the calibrations that match the actual query, as determined by the algorithm in IV.(3) below.

(c) If specified by the caller [see IV.(1).(i)], the query interface shall return to the caller the calibration fidelity (FIDELITY) values for the calibrations that match the actual query, if the optional index file column FIDELITY is defined in the key configuration file, as determined by the algorithm in IV.(3) below.

(3) Algorithm:

(a) The set of index files to be searched is determined from the CALDB configuration file by performing steps I.(3).(e), using the optional TELESCOP and INSTRUME values [see IV.(1).(b)] if provided as input.

(i) If the optional TELESCOP and INSTRUME values [see III.(1).(b)] were provided as input, and the CALDB aliases file is present [see I.(4).(c)], then perform instrument alias translation as specified in I.(4).(e) above.

(b) For each index file identified in step IV.(3).(a):

(i) Identify the set, *r*, of index file records that have values in the CAL_CNAM column that match the actual parameter value provided by the caller.

(ii) Identify the subset, r_1, of index files records from the set, *r*, for which each index file query column value matches the corresponding actual query parameter value specified by the caller.

1. If specified by the caller, translate from FITS header keyword equivalents to index file column names using the translations recorded in the key configuration file.

    a. If the actual parameter name does not match any of the FITS header keyword equivalents present in the key configuration file, then the actual parameter name should be interpreted as an index file column name without any translation.

2. If an actual parameter value is not supplied by the caller for any query column, then the corresponding null value recorded in the key configuration file shall be substituted as the actual value of the query parameter.

3. If an entry in the index file for any query column is equal to the null value recorded in the key configuration file, then any actual value of the query parameter supplied by the caller shall be considered to be a match.

4. If the data type of the actual parameter value does not match the data type of the index file column, the following conversions are applied when performing the comparison:

    a. If both data types are numeric, then the actual parameter value and index file column value are converted to the highest precision data type. We *define* the data type precision of numeric items to increase in the order `int`, `long`, `float`, `double`.

    b. If one data type is of type `bool` and the other is of type `string`, then the comparison succeeds if the data of type `bool` has a value of `F` and the data of type `string` has a value of `"F"` or `"FALSE"`; or if the data of type `bool` has a value of `T` and the data of type `string` has a value of `"T"` or `"TRUE"`.

    c. If one data type is of type `bool` and the other is of type `int` or `long`, then the comparison succeeds if the data of type `bool` has a value of `F` and the data

of type `int` or `long` has a value of zero; or if the data of type `bool` has a value of `T` and the data of type `int` or `long` has a non-zero value.

(iii) Identify the subset, r_2, of index files records by excluding from the set, r_1, all records for which index file boundary condition values do not match the corresponding actual boundary condition values specified by the caller.

1. If an actual value is not specified by the caller for a given boundary condition name, then the corresponding index file record is *not* excluded in this step. This is different from the case for query parameters, where the null value is substituted for the actual parameter value if it is not specified by the caller.

2. Boundary conditions *name*(*condition*[,…])[*units*] are considered to be matches if the actual value matches *any one* of the comma-delimited set of *conditions* in the parentheses. If the data types of the actual value and the condition value(s) do not match, then the conversions described in IV.(3).(b).(ii).4 are applied. A condition may be one of:

   a. A single numeric value: a match occurs if the actual value matches exactly the condition value.

   b. A colon- or hyphen-separated numeric range of the form `low:high` or `low-high`: a match occurs if the actual value is greater than or equal to the low limit, and less than or equal to the high limit.

   c. A string, which may optionally be enclosed in double quotes (the double quotes are required if the string includes a comma or right-parenthesis, or could otherwise be interpreted as a numeric value, numeric range, or Boolean type): a match occurs if the actual value matches exactly the condition value.

   d. A Boolean, a single character either `T` or `F`: a match

occurs if the actual value matches exactly the condition value.

(iv) If a date/time was specified by the caller as input, then identify the subset, r_3, of index file records from the set, r_2, that have the largest beginning valid date/time (CAL_VSD/CAL_VST, or equivalently, REF_TIME) that is less than or equal to the date/time specified by the caller and the smallest ending valid date/time (CAL_VED/CAL_VET, or equivalently, END_TIME) that is greater than the date/time specified by the caller.

1. If no date/time was specified by the caller then the subset, r_3, of index file records shall be set equal to the set, r_2, of index file records.

2. If the CAL_VSD index file column is not defined in the key configuration file, then the beginning valid date/time of any index file record shall be set equal to the UTC date/time that is equivalent to the beginning MJD specified in the REF_TIME index file column, if the latter is defined in the key configuration file. If neither the CAL_VSD index file column nor the REF_TIME index file column are defined in the key configuration file, then all index file records shall be treated as matching the beginning date/time comparison.

3. If the CAL_VST index file column is not defined in the key configuration file, then the value of the beginning valid time shall be set to "00:00:00".

4. If the CAL_VED index file column is not defined in the key configuration file, then the ending valid date/time of any index file record shall be set equal to the UTC date/time that is equivalent to the beginning MJD specified in the END_TIME index file column, if the latter is defined in the key configuration file. If neither the CAL_VED index file column nor the END_TIME index file column are defined in the key configuration file, then all index file records shall be treated as matching the ending date/time

comparison.

5. If the CAL_VET index file column is not defined in the key configuration file, then the value of the ending valid time shall be set to "00:00:00".

(c) If the caller did *not* specify that the calibration quality not be considered [see IV.(1).(g)], then identify the subset, r_4, of index file records from the concatenation of all sets, r_3, [one for each of the index files identified in step IV.(3).(a)] that have the highest calibration quality (stored in the index file CAL_QUAL column).

(i) If the caller did specify that the calibration quality not be considered [see IV.(1).(g)], then the subset, r_4, of index file records shall be set equal to the set, r_3, of index file records.

(d) If the caller specified a fidelity criterion [see IV.(1).(e)] *and* the FIDELITY index file column is defined in the key configuration file, then identify the subset, r_5, of index file records from the set, r_4, that meet that criterion. Possible criteria are:

(i) "HIGHEST": the index file records with the numerically largest FIDELITY value are returned.

(ii) "LOWEST": the index file records with the numerically smallest FIDELITY value are returned.

(iii) Minimum value: the index file records with FIDELITY values greater than or equal to the specified value are returned.

(iv) Maximum value: the index file records with FIDELITY values less than or equal to the specified value are returned.

(v) Minimum and maximum values: the index file records with FIDELITY values greater than or equal to the specified minimum value and less than or equal to the specified maximum value are returned.

(vi) If the caller did not specify a fidelity criterion *or* the FIDELITY index file column is not defined in the key configuration file, then the subset, r_5, of index file records shall be set equal to the set, r_4, of index file records.

(e) The subset, r_5, of index file records represents the set of matching calibrations. If the caller specified a match method [see IV.(1).(f)], then that method is used to determine which calibrations to return to the caller. If no match method is specified, then the "ALL" method shall be used. Possible match methods are:

(i) "ALL": All matching calibrations are returned to the caller.

(ii) "FIRST": The matching calibration that is recorded first (smallest row number) in the set of index files (first matching index file identified in the CALDB configuration file) is returned to the caller.

(iii) "FIRSTWARN": As "FIRST", but a warning is issued that multiple matches were found.

(iv) "LAST": The matching calibration that is recorded last (largest row number) in the set of index files (last matching index file identified in the CALDB configuration file) is returned to the caller.

(v) "LASTWARN": As "LAST", but a warning is issued that multiple matches were found.

(vi) "SINGLE": If more than one matching calibration is found then an error is returned.

(f) If specified by the caller [see IV.(1).(h)], for each calibration returned to the caller in step IV.(3).(e) above return the calibration quality (CAL_QUAL) value corresponding to the calibration from the index file.

(g) If specified by the caller [see IV.(1).(i)] *and* if the optional index file column FIDELITY is defined in the key configuration file, for each calibration returned to the caller in step IV.(3).(e) above return the calibration fidelity (FIDELITY) value corresponding to the calibration from the index file.

V. INDEX FILE CONSTRUCTION

(1) Input specification:

(a) The user may specify as input the path to the index file.

(i) If the path to the index file is not specified, the default value shell be set to `"./caldb.indx"`.

(b) The user may specify as input the path to the keyword configuration file.

(i) If the path to the keyword configuration file is not specified, the default value shell be set to `"./key.config"`.

(c) The user may specify as input the directory path corresponding to the *root* of the CALDB tree.

(i) The root directory path is the prefix that is prepended to the relative path specified in the `$CAL_DIR` index file column to construct the fully qualified path to a calibration file.

(ii) If not specified by the user, the root is set to a default value of `"{$CALDB}/"`.

(d) The user may specify as input a list of directory and/or file paths that will be used to identify FITS format calibration files to be added to the index file. There are 3 possible formats:

(i) If a fully qualified path to a file or files is specified, then only the specified file or files is/are included.

(ii) If a directory path is specified *without* the recursion character (`"+"`) appended to the path, then all FITS files located in the specified directory are included.

1. For this purpose, FITS files are assumed to be all files that have as a filename extension one of `".fits"`, `".FITS"`, `".fit"`, or `".FIT"`.

(iii) If a directory path is specified *with* the recursion character (`"+"`) appended to the path, then all FITS files located in the directory tree having the specified directory as its root are included.

Examples

The specification `"/data/myfiles/mysubdir/file.fits"` *would match only the specified file in the specified subdirectory.*

The specification `"/data/myfiles/*.foo"` *would match all of the files with the extension* `".foo"` *in the* `myfiles` *directory, but would not match any files in the* `mysubdir` *subdirectory.*

Similar to the above,`"/data/myfiles/"` *would match only FITS files in the* `myfiles` *directory.*

Contrast this to `"/data/myfiles/*"` *which would match all of the files in the* `myfiles` *directory.*

Finally, `"/data/myfiles+"` *would match all of the FITS files in the directory tree whose root is* `myfiles`*, therefore including FITS files in the* `mysubdir` *subdirectory.*

(2) Return values:

(a) Index file construction shall create or update the FITS binary table format index file specified as input [see V.(1).(a)].

(i) The index file shall include a null primary HDU and a FITS binary table HDU that contains the contents of the index. The index shall be populated according to the algorithm in section V.(3).

(ii) The extension name of the FITS binary table HDU that contains the index shall be stored in the `EXTNAME` FITS header keyword and shall be set to `"CIF "`.

(iii) The version number of the file format of the FITS binary table HDU that contains the index shall be stored in the `CIFVERSN` FITS header keyword and shall be set to `"2.0 "`.

(3) Algorithm:

(a) Define the index file binary table format using Tables 1 and 2, above, and the contents of the key configuration file.

(i) Identify if any of the columns identified in Table 1 are redefined in the key configuration file, and verify that the redefinitions meet the requirements of section I.(1).(a).(i).4.

1. If a mandatory column is redefined in the key configuration file and that redefinition does not meet the requirements of section I.(1).(a).(i).4, then ignore that redefinition and issue a warning; otherwise, redefine the column using the information specified in the key configuration file.

(ii) Identify if any of the columns identified in Table 2 are defined in the key configuration file, and verify that the definitions meet the requirements of section I.(1).(a).(ii).3.a.

1. If an optional column is defined in the key configuration file and that definition does not meet the requirements of section I.(1).(a).(ii).3.a, then define the column using the information in Table 2 and issue a warning; otherwise, define the column using the information specified in the key configuration file.

(iii) Define all remaining index file optional and query columns using the information specified in the key configuration file.

(iv) `TFORM`*nn* and `TDIM`*nn* specifications for index file FITS binary table columns are determined from the key configuration file `format` definitions using the conversions given in Table 3.

(v) In the case where an existing index file is being updated, if the index file binary table format defined in this section is not consistent with the format of the existing index file binary table, then issue an error message identifying the inconsistency and exit.

1. Consistency requires that all index column names, data types, and formats match. Ordering of index file columns is *not* significant.

(b) Repeat the following steps for each HDU (including the primary HDU and all extensions) of each FITS format calibration file specified as input to be added to the index file [see V.(1).(d)]:

(i) Determine if the HDU header contains a FITS header keyword of the form `"<CAL_CNAM>00`*nn*`"`, where `<CAL_CNAM>*` is the equivalent header keyword to the `CAL_CNAM` mandatory index file column, and *nn* is a 2 digit integer (the *keyset suffix*). If not, then skip processing this HDU and continue to the next HDU.

1. Unless redefined in the key configuration file, `"<CAL_CNAM>*"` translates to the equivalent header keyword `"CCNM*"`, so the appropriate FITS header keyword to search for will be of the form `"CCNM00`*nn*`"`.

(ii) For each optional or query column defined in the key configuration file, determine the equivalent header keyword using the information in the key configuration file.

1. If no equivalent header keyword is defined (*i.e.*, the entry is null (`""`), then define the equivalent header keyword to be the name of the index file column as defined in the key configuration file.

2. If the equivalent header keyword is of the form `"NAME*"`, then it comprises one of the keyset keywords/columns and

is handled specially.

(iii) Construct a prototype index file entry for this HDU by populating the optional and query column entries with the values extracted from the equivalent header keywords in the HDU header *that are not keyset keywords* (keyset keywords will be handled in section V.(3).(b).(v)].

1. If a header equivalent keyword is not present in the HDU header, then the corresponding column of the prototype index file entry should be populated with the null value for the column defined in the key configuration file.

(iv) Populate the following mandatory columns in the prototype index file entry for this HDU as follows:

`CAL_DEV`: Set equal to `"ONLINE"`.

`CAL_DIR`: Set equal to the directory path to the current calibration file *relative* to the root of the CALDB tree [see V.(1).(c)].

`CAL_FILE`: Set equal to the name of the current calibration file.

`CAL_XNO`: Set equal to the extension number of the current HDU in the calibration file (the primary HDU has `CAL_XNO` set equal to 0 [zero]).

`CAL_QUAL`: Set equal to 0 (zero).

`CAL_DATE`: Set equal to today's date (UTC) in the format specified in Table 1.

(v) Update a copy of the prototype index file entry for this HDU for the first keyset:

1. Set the keyset suffix *nn* = `"01"`.

2. Determine if the HDU header contains a FITS header keyword of the form `"<CAL_CNAM>00`*nn*`"` (normally `"CCNM00`*nn*`"`), where `<CAL_CNAM>*` is the equivalent header keyword to the `CAL_CNAM` mandatory index file column, and *nn* is the keyset suffix. If so, keyset *nn* is

defined, otherwise keyset *nn* is not defined.

3. If keyset *nn* is *not* defined, then no further processing is performed for this HDU. Otherwise …

4. For each index file column that has an equivalent FITS header keyword that is a keyset keyword, extract the corresponding value from the header keyword formed by appending "00*nn*" to the equivalent FITS header keyword.

    a. If a FITS header keyword is not present in the HDU header, then the corresponding column of the prototype index file entry should be populated with the null value for the column defined in the key configuration file.

5. Multiple boundary conditions (index file column CAL_CBD, with equivalent FITS header keywords "CBD*m*00*nn*") may be defined for a given keyset.

    a. The allowed values of *m* range from 1 up to the maximum value specified by the repetition count <j> in the format definition char<i>[<j>] of the CAL_CBD mandatory index file column.

    b. The maximum allowable value of <j> in the format definition is 35; if a larger value than 35 is specified then 35 is assumed and a warning is issued.

    c. The values of *m* are encoded as single characters in base 36, in the order 1, …, 9, A, …, Z up to the maximum specified by the format definition.

6. Populate the copy of the prototype index file entry for this HDU with the extracted values.

7. If the optional column REF_TIME is defined in the key configuration file *and* the value of the REF_TIME column in the copy of the prototype index file entry for this HDU has not been populated by either step V.(3).(b).(iii) or step V.(3).(b).(v), then populate the value of the REF_TIME

column as follows:

a. If the optional column CAL_VSD is defined in the key configuration file, then populate the value of the REF_TIME column in the copy of the prototype index file entry for this HDU with the modified Julian date corresponding to the UTC date and time in the CAL_VSD and CAL_VST columns. If the optional column CAL_VSD is not defined in the key configuration file, then populate the value of the REF_TIME column in the copy of the prototype index file entry for this HDU with the value 0.0.

   i. If the CAL_VST index file column is not defined in the key configuration file, then the value of the beginning valid time used to convert from UTC to MJD shall be set to "00:00:00".

8. If the optional column END_TIME is defined in the key configuration file *and* the value of the END_TIME column in the copy of the prototype index file entry for this HDU has not been populated by either step V.(3).(b).(iii) or step V.(3).(b).(v), then populate the value of the END_TIME column as follows:

   a. If the optional column CAL_VED is defined in the key configuration file, then populate the value of the END_TIME column in the copy of the prototype index file entry for this HDU with the modified Julian date corresponding to the UTC date and time in the CAL_VED and CAL_VET columns. If the optional column CAL_VED is not defined in the key configuration file, then populate the value of the END_TIME column in the copy of the prototype index file entry for this HDU with the value 999999.0.

      i. If the CAL_VET index file column is not

defined in the key configuration file, then the value of the beginning valid time used to convert from UTC to MJD shall be set to `"00:00:00"`.

9. Output the completed copy of the prototype index file entry as the new last row of the index file.

10. Increment the keyset suffix and repeat steps V.(3).(b).(v).2 through III.(3).(b).(v).9 for each defined keyset.

Table 3

Conversion from key configuration file `format` to index file

FITS binary table column `TFORM`*nn*/`TDIM`*nn* specifications

Format	**TFORMnn**	**TDIMnn**
bool	"L"	
bool[<i>]	"<i>L"	
bool[<i>,<j>,...]	"<i*j*...>L"	"(<i>, <j>, ...)"
char<i>	"<i>A"	
char<i>[<j>]	"<i*j>A"	"(<i>, <j>)"
char<i>[<j>,<k>,...]	"<i*j*k*...>A"	"(<i>, <j>, <k>, ...)"
double	"D"	
double[<i>]	"<i>D"	
double[<i>,<j>,...]	"<i*j*...>D"	"(<i>, <j>, ...)"
float	"E"	
float[<i>]	"<i>E"	
float[<i>,<j>,...]	"<i*j*...>E"	"(<i>, <j>, ...)"
int	"I"	
int[<i>]	"<i>I"	
int[<i>,<j>,...]	"<i*j*...>I"	"(<i>, <j>, ...)"
long	"J"	
long[<i>]	"<i>J"	
long[<i>,<j>,...]	"<i*j*...>J"	"(<i>, <j>, ...)"

Example format conversions:

Format	***TFORMnn***	***TDIMnn***
"char4"	*"4A"*	–
"char70[9]"	*"630A"*	*"(70, 9)"*
"double[2]"	*"2D"*	–
"float"	*"E"*	–
"long[6,2]"	*"12J"*	*"(6, 2)"*

VI. INDEX FILE VALIDATION

(1) Input specification:

(a) The user may specify as input the path to the index file.

(i) If the path to the index file is not specified, the default value shell be set to `"./caldb.indx"`.

(b) The user may specify as input the path to the keyword configuration file.

(i) If the path to the keyword configuration file is not specified, the default value shell be set to `"./key.config"`.

(c) The user may specify as input the directory path corresponding to the *root* of the CALDB tree.

(i) The root directory path is the prefix that is prepended to the relative path specified in the `$CAL_DIR` index file column to construct the fully qualified path to a calibration file.

(ii) If not specified by the user, the root is set to a default value of `"{$CALDB}/"`.

(d) The user may specify whether duplicate index file entries should be deleted from the index file.

(i) The default is not to delete duplicate index file entries.

(ii) The user may specify that the tool query whether duplicate index file entries should be deleted on a case by case basis, in which case the tool shall query the user separately for each duplicate index file entry whether it should be deleted or not (the default).

(e) The user may specify whether missing calibration files should be marked `"OFFLINE"` in the index file.

(i) The default is not to mark missing calibration files as `"OFFLINE"`.

(ii) The user may specify that the tool query whether missing calibration files should be marked `"OFFLINE"` on a case by case basis, in which case the tool shall query the user separately for each missing calibration file whether it should be marked

"OFFLINE" or not (the default).

(f) The user may specify whether invalid index file entries should be deleted from the index file.

(i) The default is not to delete invalid index file entries.

(ii) The user may specify that the tool query whether invalid index file entries should be deleted on a case by case basis, in which case the tool shall query the user separately for each invalid index file entry whether it should be deleted or not (the default).

(g) The user may specify whether missing index file entries should be added to the index file.

(i) The default is not to add missing index file entries.

(ii) The user may specify that the tool query whether missing index file entries should be added on a case by case basis, in which case the tool shall query the user separately for each missing index file entry whether it should be added or not (the default).

(2) Return values:

(a) Index file validation shall report validation errors to STDOUT, and possibly update the FITS binary table format index file specified as input [see VI.(1).(a)].

(3) Algorithm:

(a) Define the index file binary table format using Tables 1 and 2, above, and the contents of the key configuration file, as specified in section V.(3).(a).

Note

Section V.(3).(a).(v) ensures that the format of the index file and the key configuration file are consistent.

(b) Identify duplicate rows in the index file. A duplicate row is a row for which all column entries match the column entries of another row.

(i) If duplicate rows are found, then report a validation warning that includes the row numbers of the original and duplicate rows in the index file.

1. If the user specified that duplicate index file entries should be deleted [see VI.(1).(d)], then update the index file by deleting the duplicate records. When selecting records to delete, the first instance of a record should be preserved, and all subsequent duplicates should be deleted. Report this action to the user.

(c) Identify the sets of index file rows that have CAL_DEV = "ONLINE", the same fully qualified calibration file path, and that reference the same HDU.

(i) The fully qualified calibration file path is determined using the information extracted from the index file, as specified in section I.(1).(a).(i).3.

(ii) The HDU is specified by the extension number CAL_XNO.

(d) Repeat the following steps for each set of index file rows identified in VI.(3).(c), above:

(i) Verify that the specified file and HDU exist using the fully qualified path and extension number computed above.

1. If the specified file and/or HDU *does not* exist, report a validation warning that includes the fully qualified path to the calibration file, and the row number(s) in the index file.

a. If the user specified that missing calibration files should be marked "OFFLINE" [see VI.(1).(e)], then update the appropriate row(s) of the index file by setting CAL_DEV = "OFFLINE". Report this action to the user.

2. If the specified file and HDU *does* exist, perform the following steps:

a. Construct prototype index file entries for each defined keyset using the contents of the HDU header, according to the specifications V.(3).(b).(i)–V.(3).(b).(v).7 and V.(3).(b).(v).9.

b. For each index file row that is part of the set of

index file rows currently being processed, verify that the index file row corresponds to one of the prototype index file entries constructed in the previous step by matching corresponding column entries.

i. If the index file row *does not* match any prototype index file entries, report a validation warning that includes the fully qualified path to the calibration file, and the row number(s) in the index file.

(1) If the user specified that invalid index file entries should be deleted [see VI.(1).(f)], then update the index file by deleting the appropriate records. Report this action to the user.

c. For each prototype index file entry that *does not* match *any* index file entry, report a validation warning that includes the contents of the prototype index file entry.

i. If the user specified that missing index file entries should be added [see VI.(1).(g)], then update the index file by outputting the prototype index file entry as the new last row of the index file. Report this action to the user.

VII. INDEX FILE MERGING

(1) Input specification:

(a) The user shall specify as input the paths to the index files to be merged.

(b) The user may specify as input the path to the resulting merged index file.

(i) If the path to the merged index file is not specified, the default value shell be set to `"./caldb.indx"`.

(c) The user may specify as input the path to the keyword configuration file.

(i) If the path to the keyword configuration file is not specified, the default value shell be set to `"./key.config"`.

(d) The user may specify whether duplicate index file entries should be deleted from the merged index file.

(i) The default is to delete duplicate index file entries.

(2) Return values:

(a) Index file merging shall merge the FITS binary table format index files specified in VII.(1).(a) into a single merged FITS binary table format index file specified in VII.(1).(b), and may report duplication messages to `STDOUT`.

(3) Algorithm:

(a) Define the index file binary table formats for each input index file to be merged [see VII.(1).(a)] using Tables 1 and 2, above, and the contents of the key configuration file, as specified in section V.(3).(a).

Note

Section V.(3).(a).(v) ensures that the format of the input index files to be merged and the key configuration file are consistent.

(b) Concatenate the rows of the input index files to be merged.

(c) Identify duplicate rows in the index file. A duplicate row is a row for which all column entries match the column entries of another row.

(i) If duplicate rows are found and the user specified that duplicate

index file entries should be deleted [see VII.(1).(d)], then update the merged index file by deleting the duplicate records. When selecting records to delete, the first instance of a record should be preserved, and all subsequent duplicates should be deleted. Report this action to the user.

(d) Copy the null primary HDU from the first input index file to the null primary HDU of the merged index file specified in VII.(1).(b), updating the DATE FITS header keyword (if present). Output the merged index file records into a single merged FITS binary table HDU, updating the DATE FITS header keyword (if present).

REQUIREMENTS TO BE SPECIFIED

The following requirements are TBD:

- Version management
  - CALDB and file level version encoding
  - Version consistency checking for index files and key configuration files
  - Automated versioning of CALDB updates
- Managing related files and file groups
  - Related file versioning and calibration quality migration
  - Identifying file groups in data products
- Support for time systems other than UTC
  - Identifying a calibration's time system for the caller
  - Converting between time systems
- Index file editor application requirements