OGIP Calibration Memo CAL/SW/95-002

OGIP CALDB Access Subroutines

(a brief overview for programmers)

Ian M. George,
Lorraine Breedon,
and
Michael F. Corcoran

Mail Code 662
NASA/GSFC,
Greenbelt, MD20771

Version: 2006 Aug 21
(Previous Versions: 1996 Jan 02)

SUMMARY

This document provides an introduction of the FORTRAN subroutines, supplied as part of the FTOOLS package, which can be called by tasks in order to access their local copy of the HEASARC's Calibration Database (Caldb). Also included is a brief review of the HEASARC Caldb, the preliminaries a user needs to have completed prior to using any task which calls the Caldb-access subroutines, along with programming suggestions/examples.
Intended Audience: Programmers wishing to make use of the their local Caldb.

Forward

This document is intended to provide a general introduction to HEASARC's calibration database system access software. Every attempt has been made to present the information in a clear, and (moderately) concise manner, and copious use of cross-referencing is made, although reference to other Calibration Memos *etc* is occasionally required.

At the current time, this document is updated as necessary, and not guaranteed complete.

All questions, comments and suggestions should be directed to:
caldbhelp@olegacy.gsfc.nasa.gov
or Caldb personnel as listed on the Caldb Personnel Page.

The latest version of this document in Portable Document Format can be obtained at
`http://heasarc.gsfc.nasa.gov/FTP/caldb/docs/memos/cal_sw_95_002/cal_sw_95_002.pdf`

LOG OF SIGNIFICANT CHANGES

Release Date	**Sections Changed**	**Brief Notes**
1994 Sep 12		Original Version
1995 Jan 11	All	Made compatible with `LaTeX2HTML` software
1995 Oct 18	All	Updated & made specific to subroutines only
1996 Jan 02	4.1	Updated to include `caldb_info` subroutine
2006 Aug 21	all	minor typo corrections & updates, including change from gtcal.f to gtcalf.f (MFC)

Contents

1 INTRODUCTION

One of the main goals of the HEASARC's Calibration Database (Caldb) system is to provide analysis software access to relevant calibration datasets, without having to have the names and locations of those datasets hard-wired into the analysis software. "Access" means the capability to identify a calibration data file based on specified instrumental properties or parameters. These parameters or properties may include (but are not limited to) mission and instrument name, detectors and filters in use, observation date and time, and other parameters like temperatures, high voltages, magnetic rigidities, orbital parameters, time since passage through the South Atlantic Anomaly, etc. This allows updates to the calibration datasets to be made immediately accessible without the need for changes to be made to the analysis software.

To achieve these goals, the HEASARC has designed and developed the Calibration Database. This is essentially a simple relational database whereby critical information concerning each and every calibration dataset within the database for a given instrument is stored in a **Calibration Index File (CIF)**, which is a FITS-formatted binary table. The format of the CIFs are described in detail in OGIP memo CAL/GEN/92-008. The Caldb access software described in this document basically interogates the relevant CIF, searching for the entry (*ie* dataset) required by the application task which matches the observational parameters.

A detailed knowledge of the CIF format is not required here. In essence a CIF consists of a FITS BINTABLE in which each row refers to a different calibration dataset, with necessary information stored in each of the many BINTABLE columns. For the purposes of this brief introduction, the CIF table columns are those containing the name and location of each calibration dataset, the instrument to which it refers, a code describing what kind of calibration information is actually stored within the dataset, and a time/date stamp indicating when the dataset should first be used. These will be explained in more detail below. The information contained within the CIF allows the Caldb access software to determine which dataset(s) are required for a given analysis task, and to return the name & location of the dataset(s) to the task. It is then the responsibility of the application to open, read and use the dataset in an appropriate manner.

This document describes the various Caldb access subroutines available from the HEASARC CALTOOLS software suite, and gives examples on their use.

2 IMPORTANT PRELIMINARIES

There are two basic requirements for a user/programmer to be able to use their local Caldb (and the access subroutines described in Section 4):

- that they can see the disk, file system *etc* containing the data files
- that they define a number of environment variables (or logicals) which allow the Caldb soft-

ware to find the Calibration Index Files (CIFs; see CAL/GEN/92-008 for further details) and hence the calibration datasets themselves. The set-up of these environment variables (or logicals) can be done by "sourcing" a single file supplied by the HEASARC Caldb and is described in detail in the Caldb Users' Guide (CAL/GEN/94-002). For c-shell users this file is called the `caldbinit.csh` file. The most important environment variables (or logicals) which must be defined are:

- `CALDB` – the path to the top-level directory of the local Caldb
- `CALDBCONFIG` – the name of the so-called Caldb configuration file, which contains the all the vital information regarding the location of the local CIFs, plus other locally-important files.

3 RECOMMENDED PROGRAMMING PRACTICES

There are a number of issues which should be born in mind when decided whether/how to include access to a local Caldb within a standalone task.

3.1 Is the Caldb-access software absolutely necessary ?

Obviously the Caldb-access software assumes that the site where the users of your task are running it have (or could have) an up-to-date Caldb set-up, and have access to it. This means that there are some overheads associated with setting up & maintaining a local Caldb. Thus in cases where your task will **always** require the **same** highly **task-specific file** (which **will not change** with time), then Caldb-access may not be 100% necessary.

However, experience with many missions has highlighted that nearly all calibration files **DO** change at some point, and thus use of the Caldb structure and access software often saves significant time in the long run.

3.2 Allow user input options

Making an analysis task **totally** reliant on having a local Caldb set-up and available to users may be as bad as having filenames & locations hard-wired into the code. It is nearly always trivial to structure analysis software so that any local Caldb can be used, **or** so that the user can manually provide the location & name of the calibration file to be used. This approach allows users to construct their own calibration datasets to be used in place of "standard" files, which is especially important during analysis tool development and testing.

FTOOLS tasks usually allow a user to specify a character-string parameter for each calibration file which is to be input. This input may be a "special string" (usually `CALDB`) which signifies that files given in the Caldb are to be used; alternatively a user can simply provide the full path to the desired calibration file as the input, bypassing the Caldb.

3.3 Report failures back to the user in some detail

One of the biggest complaints the Caldb gets is that when the Caldb-access software fails, the task from which it is called fails to deliver coherent and/or useful error messages.

The error-reporting of the Caldb-access software is being improved, but the messages will always be less coherent/detailed compared to those which could be constructed by the main task. This is particularly the case when any of the Caldb-access routines are run in "silent-mode" (see below). Assuming the error is considered fatal by your task, then at the very least, the task should report which type of calibration file the software was attempting to find, and (perhaps) suggest a course action to the user.

3.4 Include the details of any calibration used in the output file

All tasks which obtain & use a calibration dataset via the Caldb-access software should also include the name of the file (and the extension within that file if necessary) somewhere/somehow in the output file. This will provide crucial history information to users should the calibration change at some time in the future. Clearly including this information is also a good programming practice even when the Caldb-access software is not used. In the case of FITS output files, the most obvious way of doing this is via `COMMENT` or `HISTORY` cards.

4 THE CALDB ACCESS SUBROUTINES

4.1 caldb_info.f

The FORTRAN subroutine `caldb_info.f` is intended to enable tasks to check and (if desired) report back on whether a local Calibration Database (Caldb) appears to be correctly set-up and accessible to a user. The subroutine is distributed with the FTOOLS software package, and source code can be found in the ftools library as `callib/src/gen/caldb_info.f`. The `caldb_info` subroutine can be linked through the `libcaltools.a` library. Programmers may therefore wish to call `caldb_info.f` prior to calling other Caldb access routines within a given task. The level of checking and detail of the reporting depend upon the passed parameters `mode` and `chatter` as described below.

If a problem with the user's Caldb set-up/access is detected, then an error message will be written to `STDERR`, and the passed parameter `ierr` will be returned with a non-zero value.

The calling sequence is as follows:

```
call caldb_info(chatter,mode,mission,instrument,ierr)
```

where

```
        character*(*) mission,instrument
        character*(*) mode
        integer chatter, ierr
```

where the arguments are as follows:

- `chatter` (input)
  Flag to indicate how chatty (to `STDOUT`) the subroutine is during execution. A value of `chatter=9` is nominal, with lower/higher values producing quieter/verbose output (to `STDOUT`) respectively. A value of `chatter=0` gives totally silent running (to `STDOUT`), with the exception of error messages. For all values of chatter, a message will be written to `STDERR`, and `ierr` returned with a non-zero value, if any errors are encountered.
- `mode` (input)
  This parameter controls to what level the putative Caldb set-up is tested.
  - For `infomode=BASIC`, the task simply checks that the two environment variables (or logicals under VMS) required for the Caldb are defined.
  - For `infomode=INST`, in addition to performing the checks for `infomode=BASIC`, the subroutine checks it can find entries for the mission/instrument combination (defined via the passed parameters `mission` and `instrument`: below) in the Caldb Configuration file (as defined by the `CALDBCONFIG` environment variable/logical – see Section 2). At high chatter, the location of the relevant files and directories are reported upon.
  - `infomode=FULL` is reserved for future use.
- `mission` (input)
  The name of the mission or telescope for which the Caldb Configuration file is to be searched. The value of this passed parameter is only important when `infomode=INST`.
- `instrument` (input)
  The name of the instrument for which the Caldb Configuration file is to be searched. The value of this passed parameter is only important when `infomode=INST`.

4.2 gtcalf.f

The current subroutine to use for retrieving information regarding CALDB datasets is the FORTRAN subroutine `gtcalf.f`. This subroutine was previously called `gtcal.f`. This subroutine is distributed with the FTOOLS] software package, and source code can be found in the ftools source distribution in the directory `ftools/callib/src/gen`. The `gtcalf` subroutine can be linked through the `libftools.a` library.

The `gtcalf` subroutine returns the location of calibration datasets located in the Calibration Database by referring to the appropriate Calibration Index File (CIF; see CAL/GEN/92-008 for further details). Selection of the appropriate calibration data is based on the values of the arguments `mission`, `instrument`, `sub-instr`, `filter`, `codenam`, `date`, `time` and `expr`. These arguments respectively describe the mission or telescope, instrument, sub-detector (if any), filter (if any), type of dataset, date, time, and calibration boundaries for which the returned datasets should be valid (see also below). In addition to the arguments explicitly listed here, this routine also uses the values of the environment variables (logicals) `CALDB`, and `CALDBCONFIG` (as defined in Section 2). See the Caldb Users' Guide (CAL/GEN/94-002) for info on how these system variables should be set.

The maximum number of datasets matching the criteria returned is speficied by the `maxret` argument. Any datasets which meet the selection criteria are returned through the `filenam` and `extno` arrays. Each element of the `filenam` array contains the complete system dependent path (including the filename) to the file where the calibration data resides. The corresponding element of the `extno` array contains the FITS extension number of the calibration data within the file.

The calling sequence is as follows:

```
        call gtcalf(chatter,tele_str,instr_str,detnam_str,
     &  filt_str, codenam_str, strtdate, strtime, stpdate, stptime,
     &  expr_str,maxret,filenam,extno,online,
     &  nret,nfound, status)
```

where

```
        character*(*) tele_str, instr_str, detnam_str, filt_str
        character*(*) codenam_str, expr_str
        character*(*) strtdate, strtime, stpdate, stptime
        integer       maxret
        character*(*) filenam(maxret), online(maxret)
        integer       extno(maxret), nret, nfound, status
        integer       chatter
```

where the arguments are as follows:

- `tele_str` (input)
  The name of the mission or telescope for which the returned datasets should be valid. Corresponds to the `TELESCOP` keyword value found in the requested calibration file(s).

- `instr_str` (input)
  The name of the instrument for which the returned datasets should be valid. Corresponds to the `INSTRUME` keyword value found in the requested calibration file(s).

- `detnam_str` (input)
  The name of the detector for which the returned datasets should be valid. Corresponds to the DETNAM keyword value found in the requested calibration file(s). If a hyphen (`-`) is passed, no selection will be made on detector names.

- `filt_str` (input)
  The name of the filter for which the returned datasets should be valid. Corresponds to the `FILTER` keyword value found in the requested calibration file(s). If a hyphen (`-`) is passed, no selection will be made on filter values.

- `codenam_str` (input)
  The OGIP codename for the requested dataset. Corresponds to the `CCNM`*xxxx* keyword value found in the requested calibration file(s).

- `strtdate` (input)
  The date when the datasets should be valid. This value should be in `dd/mm/yy` format. If the string `now` is passed, the current system date is substituted.

- `strtime` (input)
  The time of the day when the dataset should be valid. This value should be in `hh:mm:ss.s` format. If the string `now` is passed, the current system time is substituted.

- `stpdate` (input)
  The end-date when the datasets should be valid. This value should be in dd/mm/yy format. If 'now' is passed, the current system date is substituted.

- `stptime` (input)
  The time of the day (stpdate) when the dataset should be valid. This value should be in hh:mm:ss.s format. If 'now' is passed, the current system time is substituted.

- `expr_str` (input)
  A boolean expression used to select on calibration boundary parameters. Calibration boundary parameters are part of the `CBD`*nxxxx* keyword values found in the requested calibration file(s). (e.g. In the string `ENERGY(2-12)keV`, `ENERGY` is the boundary parameter.) Currently, the expression syntax only contains the arithmetic operator `.eq.` and the logical operator `.and.`. To request a dataset which is valid for an off-axis angle of 10 arcmins and an energy of 5.0 keV, one would pass an expr value of:

  ```
  theta.eq.10.and.energy.eq.5
  ```

  If no `expr_str` selection is required, a hyphen (`-`) should be passed.
  NOTE: The `expr_str` parser cannot distinguish between certain valid and invalid expressions. For example, expressions such as:

  ```
  cor.eq.7.and.and.energy.eq.5
  ```

will be incorrectly interpreted as `cor=7` and `and.energy=5`. A `lex` and `yacc` parser is planned for this routine which will work much better.

- `maxret` (input)
  An integer defining the number of elements in the `filenam` and `extno` arguments and the number of datasets to return.

- `filenam` (returned)
  A character array containing the complete system dependent path(s) the to the file(s) in which the requested calibration data is stored. Dimensioned using the `maxret` argument.

- `extno` (returned)
  An integer array containing the FITS extension numbers of requested datasets. Each extension number is valid for the filename found in the corresonding element of the `filenam` array. Dimensioned using the `maxret` argument.

- `online` (returned)
  A character array which specifys the on-line/off-line status of the corresponding file in the filenam argument. Each element contains either `ONLINE` or a string describing the off-line status of the file. If the string `ONLINE` is not encountered for a particular `filenam` element, then this file has been moved to an off-line media (e.g. magnetic tape) and the main program will not be able to access the data. This routine will monitor the online argument and set a return status of 110 when returning if an off-line file is detected.

- `nret` (returned)
  The number of entries returned to the `filenam`, `extno` and `online` arguments.

- `nfound` (returned)
  The nfound argument reports the total number of datasets found during the search which match the selection criteria. May be larger than `maxret`.

- `status` (returned)
  The status argument returns an integer specifying the success of this routine. A non-zero status flag means that this routine was unable to return any datasets to the main program, while a zero flag means that everything executed normally.

5 FREQUENTLY ASKED QUESTIONS

5.1 What's the best introduction the Caldb ?

5.1.1 For the Programmer

The Caldb & its access software was designed with the goal of application programmers **not** having to know too much about the inner-workings of the database itself. Indeed, assuming the calibration file(s) the programmer's application needs to locate is already available in a local

Caldb, then this document is intended to contain everything the programmer needs to know in order to find the file (though certainly **not** how to read & use the file – see SectionSec:now-what).

Besides the general documents listed in Section 5.1.2, programmers wishing to know about the more technical aspects of the Caldb can check-out:

- CAL/GEN/92-008: Calibration Index Files
- CAL/GEN/92-011: Mandatory Keywords for Calibration Index Files
- CAL/SW/93-018: How to Install Callib & Caltools Software
- OGIP/92-012: The OGIP FORTRAN Programming Standard

5.1.2 For General Users

- CAL/GEN/94-002: HEASARC Caldb Users' Guide
- CAL/GEN/93-006: The Organization of the HEASARC Caldb
- CAL/GEN/91-001: The HEASARC Calibration Database (brief overview)

5.1.3 For Local Caldb Managers

- CAL/GEN/94-004: How to Install a HEASARC Caldb
- CAL/GEN/93-022: How to Install a Calibration File
- CAL/GEN/92-015: Maintanence Guide for a Caldb at a Remote Site

5.2 What's the difference between the "HEASARC Caldb" & a "local Caldb"?

There are essentially two main components to keep in mind: 1) the Caldb database system (ie the software) written and distributed by the HEASARC, and 2) the calibration data files themselves (ie the database) on-line at the HEASARC. Collectively, these are often referred to as the "HEASARC Caldb". When some/all the calibration data files have been copied to a remote site, and the Caldb database software installed (CIFs made *etc*), then this is often referred to as a "local Caldb".

5.3 What should I put in my "help file"?

The help file should include at minimum:

- A brief description of the purpose of the task
- a brief description of the algorithm used
- a list of the input parameters and a brief description
- a list and description of the outputs
- known error conditions
- samples of use.

5.4 Can I use the ftools/refdata area?

The "`refdata`" area is a directory of calibration files distributed with the FTOOLS] software package. Use of the refdata area is deprecated, since maintenance and versioning of the data files in this directory is difficult. **However**, strenuous

5.5 What if a local Caldb does not appear to be set-up?

If a local Caldb does not appear to be set-up at your site, but you anticipate writing tasks which will/can make use of a local Caldb, then we suggest you investigate setting one up. The process is described in CAL/GEN/94-004 and really isn't that difficult. It should also be noted that as a "local Caldb manager", one can choose which mission/instrument combinations you wish to include in your local Caldb, and even which subset of data files. **A local Caldb does not have to contain all the calibration files available at the HEASARC**. Indeed your local Caldb could even only contain the files ncessary for your task(s).

5.6 What do I do once I've found the Calibration filename?

Use of calibration data is highly application-specific. However, it should be noted that Caldb files conform to strict FITS format guidelines. In most of these cases, there are standard FORTRAN readers & writers distributed via the FTOOLS package (usually in the `libcaltools.a` library). Thus, it's recommended that you look for such a standard reader and determine whether it is exactly what you were after, or can be used as a template.

5.7 What if a calibration file is not found?

Assuming the calibration file is vital to your task, then clearly the task should gracefully come to a halt (with plenty of explanation to the user explaining what type of file wasn't found – see Section 3.3). There basically three explanations:

- the various inputs to the access software were passed correctly, in which case
  - **either** the users local Caldb is out-of-date, in which case the user will have to request that their local Caldb manager copy the latest calibration datasets from the HEASARC (or wereever) to their local Caldb and up-date the relevant CIF(s)
  - **or** there really is no valid calibration available, in which case the user has a real problem, and should contact the relevant GOF *etc*
- the user and/or preceeding code passed incorrect inputs to the access software. One of the most common mistakes are requests for calibration datasets valid prior to the start of the mission.
- there is a bug in the access software (clearly impossible)

In all cases, it should be obvious that a coherent explanation to the user will save everybody time.

It should be remembered that in some cases, a failure to find a certain type of calibration file might not be a fatal error. Intelligent tasks might even anticipate such a situation, and do something appropriate (like construct the required calibration dataset for itself). There is nothing wrong in writing tasks like this. However, in such cases, it is probably desirable to run the access software in "silent mode" so that users aren't confused by apparent error messages.

5.8 What if many calibration files are found (unexpectedly)?

It is certainly valid to write tasks whereby multiple calibration datasets are returned, leaving the task to decide if/how each dataset is to be used. This is achieved by deliberately passing "sloppy" inputs to the access software. Clearly, if the access software unexpectedly claims more than one calibration file satisfies the input criteria, then in most cases the task should gracefully come to a halt. Again, a coherent explanation as to why it is stopping should be supplied to the user.

5.9 Where do I deliver new calibration files to ?

Send e-mail to:
caldbhelp@olegacy.gsfc.nasa.gov
or to Caldb personne listed on the Caldb Personnel Page. with as many details as possible regarding the dataset (what it's for; where it came from; etc.) **during development of the task**. They will then discuss the format with you, perhaps suggest additional keywords and/or updates to existing keyword values. Finally they will instruct you how to formally deliver the file.

6 EXAMPLES

6.1 Using caldb_info & gtcalf

Example 1

In the following hunk of FORTRAN, it is assumed that the user has entered a character string (`calfexp`) for the calibration file to be used.

- In cases where this string has the special value `CALDB` (upper or lower case), then:
  - the `caldb_info` subroutine is first used silently to check that a local CALDB is set-up & available for the user for the required `mission/instrument` (`gtcal` will fail in any case is this is not true).
  - `gtcalf` is then used to find a specific type of calibration dataset (with a codename `COLLRESP`) valid for a given observation date/observation time along with a number of other constraints.

  Only one valid calibration file is anticipated in this example, and the code errors out whenever it gets the feeling it's not sure what's going on.
- In cases where the special string `CALDB` is **NOT** entered, then the code assumes `calfexp` contains the path, name & (potentially) extension number of the calibration file to be used.

```
C Check to see whether CALDB access software is to be used
      if((calfexp(1:5).eq.'CALDB').or.(calfexp.eq.'caldb')) then
c ....... Yes, the local CALDB is to be used, so check that it is available
          infomode = 'INST'              ! check CALDB OK this mission/instrument
          infochatter = 0                ! Run in silent mode
          call caldb_info(infochatter,infomode,mission,instrument,ierr)
          if(ierr.NE.0) then
            message = 'CALDB not defined/available'
            call wterrm(subname, version,message)
            message = 'Task requires CALDB to be both defined '//
     &          '& available in order to run'
            call wtinfo(chatter,1,1, message)
            goto 999                     ! Error out
          endif
c ....... Now call the caldb access software
          chatter = 0                      ! Run in silent mode
          tele_str= 'XTE'
          instr_str = 'PCA'                ! Mission Name
          detnam = '-'                     ! DETECTOR name
          filter = '-'                     ! Not applicable for this instrument
          codenam_str = 'COLLRESP'         ! Looking for a collimator response
          expr_str = 'energy.eq.10'        ! Want dataset valid at 10 keV
          maxret = 1                       ! Only expect 1 dataset to be returned
          call gtcalf(chatter,tele_str,instr_str,detnam_str,
       &  filt_str, codenam_str, strtdate, strtime, stpdate, stptime,
       &  expr_str,maxret,calfilename,extno,online,
       &  nret,nfound, status)
          if(status.NE.0) then
            message = 'Problem obtaining valid COLLRESP dataset '//
     &          'valid at 10 keV from the CALDB'
            call wterrm(subname, version,message)
            goto 999                      ! Error out
          elseif(nfound.gt.1) then
            message = 'More than one valid COLLRESP dataset '//
     &          'valid at 10 keV found in the CALDB'
            call wterrm(subname, version,message)
            message = 'Unable to determine which to use -- aborting'
            call wtinfo(chatter,1,1, message)
            ierr = 1
            goto 999                      ! Error out
          endif
c ....... local CALDB is not to be used, rather the user has apparently entered
c         the path/name of the calibration file themselves - so translate its
c         name (taking off any extension specified by the user).
      else
         call fcpars(calfexp, calfilename(1),calextno(1),ierr)
          if(ierr.NE.0) then
            message = 'Problem parsing calfexp experession'
            call wterrm(subname, version,message)
            goto 999                      ! Error out
          endif
      endif
c ... So, now
c       calfilename(1) contains the name of the calibration file to be opened
c       calextno(1) contains ext# of dataset (or -99 indicating search reqd)
c Open i/p file
        status = 0
        call cgetlun(iunit)
        call ftopen(iunit,calfilename(1),0,block,ierr)
        if(ierr.ne.0) then
           message = 'Problem opening '//calfilename(1)(:clenact(calfilename(1))
           call wtferr(subname, version, ierr, message)
           goto 999                      ! Error out
        endif
```

```
c                                 ..... etc etc .....
```