NOAO SDM Document PL020

The NOAO Calibration Library System

F. Valdes[1]

National Optical Astronomy Observatories
Science Data Management

DRAFT: September 27, 2010

[1]NOAO Data Products Program, P.O. Box 26732, Tucson, AZ 85732

Table of Contents

Abstract

The NOAO Calibration Library System manages calibration information and a calibration file repository. It operates as a service to respond to requests for calibration information and files and to accept requests for adding calibrations to the library. A notable feature of the system is the built-in logic to select the ”best” calibration given information about the data to be calibrated. This document describes this key component of the NOAO pipelines.

Keywords: pipeline, calibration

Purposes of this Document

The purposes of this paper are to document the NOAO Calibration Library System as it currently exists and to provide a model for NOAO collaborators needing to develop similar capabilities. To address the former it provides some very specific details and for the latter it attempts to point out the logical functionality and alternative possibilities.

1 Introduction

Almost by definition one of the most fundamental aspects of instrument calibration pipelines is the application of the most appropriate calibration algorithms and data to the input instrument data. To encapsulate this function the concept of a calibration library component is used. As the name implies a calibration library provides a way to check-in and check-out calibration information. It needs a catalog system and a reference librarian to help identify the best resource.

The NOAO realization of this concept is as a service with a defined protocol for pipeline modules: pieces of code which request and apply or create and submit calibration. It is a system in that it provides a cataloging system, a file management system, a data transport system, and a selection system for the best calibration.

The calibration library system we describe here is designed to handle multiple pipelines it is also usable as separate independent instances. The current operational mode for the two pipelines in production is to have them run under separate accounts with separate calibration libraries instances (that is the same code but different databases and file management).

The NOAO calibration library implementation is an auxilary middleware component distributed with the NOAO High Performance Pipeline System (NHPPS). As such it is not required to run the basic pipeline framework.

2 Concept of Operation

This section outlines how the calibration library is operates as part of the the larger NOAO pipeline operations.

The calibration library server is typically part of the pipeline start-up scripts where all pipeline components, which includes a variety of servers, are started. In some cases, since the server is an independent component, the calibration library can be shut-down and started while the rest of the pipeline system is running (or not).

It is possible to have multiple calibration libraries running to distribute the load. It is then part of the pipeline configuration to include addresses for modules to contact calibration libaries appropriate for different types of calibrations. Currently separate calibration libraries are used only for calibration rules (small scripts used by modules). All other calibration information, and especially bulk calibration files, are served by a single calibration library on a node selected (though not solely) for this purpose. A distributed capability for the ”put” function would be a challenge that we have not tackled, though there is untested mirroring code included in the implementation. It would be potentially possible to have distributed file repositories organized by, say, CCD.

As the pipeline runs it is typically working on a particular instrumetal dataset consisting of one or more exposure files. A module makes queries using an interface headers of the expo makes queries, through a single interface routine, for

Table 1: Catalog Fields

CLASS	CHAR	Calibration class
MJDSTART	FLOAT	Earliest validity in MJD
MJDEND	FLOAT	Latest validity in MJD
DETECTOR	CHAR	Instrument identifier
IMAGEID	CHAR	Detector identifier
FILTER	CHAR	Filter name
EXPTIME	FLOAT	Exposure time
NCOMBINE	INT	Number combined to create calibration
QUALITY	FLOAT	Data quality ranking
MATCH	CHAR	General purpose match string
DATATYPE	CHAR	Calibration value type
VALUE	CHAR	Calibration value

3 Selection of the Best Calibration

At the conceptual level the calibration library provides some artificial and/or operator intelligence logic to select the best calibration given information from a request. Artificial intelligence logic encompass a wide range of methods and complexity and operator intelligence means interactivity and significant personnel impact. These broad statements are meant to indicate that developers and projects have to weigh how much effort to put into this function and the trade-offs in pipeline operations.

The NOAO implementation uses a fairly simple selection mechanism which makes use of, but does not require, operator support. The short summary is that selection of the calibration information consists of a fixed, moderately complex database query using the request information as the constraints and making use of the data system to rank the choices and return the highest ranked calibration.

The maximum SQL query, the one when all possibly request parameters are provided, is

As noted elsewhere, the protocol allows the requesting pipeline to provide as little or a much information as appropriate or desired. This flexibility translates into the number of SQL constraints as the calibration library service builds the SQL query. Examples of this are shown below.

4 The Catalog

The library catalog is a database table named ”catalog”. The schema for this table is shown in figure **??**. In our implementation we chose sqlite, the driver being simplicity, though any other database could have been used and NOAO may substitute another one in the future. One minor useful feature is that the database is a simple file that can be copied, backed up, and distributed simply.

Table 2: DATATYPE Values

keyword	a scalar quantity
file	a file (typical text)
image	an image file
directory	a directory

The calibration datum the catalog is designed to provide is the string-valued ”VALUE” field. Associated with this is the ”DATATYPE” field defining the kind of data resource the value represents. Note that thie data type is different than the class which is the type of calibration. The data types are listed in table **??**.

A keyword quantity is just a string. The interpretation and parsing of the string is left to the pipeline module. It may be parsed as a numeric value, identifier, or some kind of simple structure serialization. The name is indicative of the typical way the value is captured by the NOAO pipelines; it is associated with an image as a new or updated keyword with keyword name being the class. For example, the class ”gain” would have a numeric gain value that is added/updates a GAIN keyword in an image header.

The other data types are files of some kind, where a directory or image are specific kinds of files. The ”image” type is indicative of a primary class of calibration data in astronomical pipelines consisting of an image (in the image format used by the pipeline such as FITS). Note that in addition to the usual type of calibration image this class is also used for maps such as bad pixels or linearity coefficients. The file type is used for non-image types of calibrations which are often text files. Examples of this are world coordinate system default solutions and cross-talk coefficients. The directory type is used to provide a collection of files in a self-contained way. For the IRAF-based NOAO pipelines one type is a directory of default parameters known to IRAF users as the ”uparm” directory.

As with the keyword data type, the value of the file types is also typically captured in the pipeline as a keyword, whose identifier is the class, associated with a particular image or set of images. The pipeline modules will then later access the keyword to get the file or directory name. The pipeline is expected to intelligently manage the requests to the calibration library by minimizing the number of queries for the same quantity and maintaining local caches of the files for repeated use (see §).

The primary organization of the calibration data in the calibration library is by calibration ”class”. Classes, provided their values fall into one of the data types, may be added at run time as needed. The list of class used by the MOSAIC and NEWFIRM pipelines is given in table **??**.

[Rules??]

5 File Repository

File management can take a variety of forms. One is to make use of an external archive system. For the NOAO implementation a simple internal repository using the host file system was chosen.

Table 3: Table of Classes

ampnames	file
bandtype	keyword
bpm	image
calops	keyword
filtalias	keyword
filtid	keyword
gain	keyword
lamrange	keyword
magzero	keyword
magzref	keyword
photcoef	keyword
photfilt	keyword
photindx	keyword
pipedata	directory
rdnoise	keyword
saturate	keyword
sftregions	file
sftskycounts	keyword
uparm	directory
wcsdb	file
xtalkfile	file

The reasons were that at the time the first pipeline needed a calibration library system there was no archive available, the data volume was understood to be modest, and simplicity was a driving principle. Note that the NOAO pipelines also submit calibration files to a user archive so the data is indirectly available to the pipelines as well.

One feature of an archive is typically strong protection of data. Since, as just noted, the calibration data also exists in the archive, the level of protection needed in the library is not as high. However, it is a simple matter to use host tools to provide backups and the NOAO calibration library, both database and files, is automatically backed up nightly to another disk at NOAO.

Under a root directory, defined by an environment variable, are directories for each ”class” using the class name for the directory name. Under the class directories which contain files or directories as entries are subdirectories named by the modified Julian date by which they are cataloged. Therefore, the catalog fields for the class and MJD (rounded to an integer day) is sufficient to locate or place a file in this simple repository.

6 Data Transport

The files in the calibration library must, at some point, be moved in and out of the file repository. One might naively expect this to be a function of the calibration library and this could be provided. However, it is critical to efficiently handle this, particularly for the much more frequent calibration retrieval. The knowledge of how to do this resides in the pipeline framework or the data flow logic of the pipeline application. For this reason the calibration library protocol returns metadata information about the file along with the file name.

The protocol returns the last modify date and the file size. The pipeline interface tool (getcal), which is running on the node of the requester, uses this information to manage a cache. It checks the cache for the presence of a local version and also check the modify time and size as a quick check against the possiblity the local version is no longer up-to-date. Note that in practice the NOAO pipelines never update files and operators never exercise the possiblity of doing this while the pipeline is running.

When the interface tool determines that it needs to cache a copy of a calibration file it uses the appropriate interface to get the file returned by the calibration library. For the NOAO implementation with its own simple file repository this can be done using network tools from IRAF (IRAF networking) or Unix (scp or rsync).

When the interface tool checks the local cache it also takes the opportunity to remove any calibration files not accessed in a certain number of days (say 1 day).

7 Protocol

The NOAO pipeline servers use a simple standard text-based protocol. A request consists of a number of keyword=value pairs. One of these is the service desired.

Table 4: Arguments to get a calibration.

CLASS	The type of calibration we want. e.g. 'Zero'
MJD	Modified Julian Day. e.g. 52642.4
DETECTOR	The string identifying the detector e.g. 'Mosaic2'
IMAGEID	Image/Amp ID e.g. 'SITe #98164FACR10-02 (NOAO 21), lower left (Amp13)'
FILTER	Filter name e.g. '815 815_v1 k1026 '
EXPTIME	Exposure time e.g. 120.
DMJD	DMJD window e.g. 10 for +- 10 days
DEXPTIME	Exposure time window as fraction of EXPTIME e.g. 0.1 for within 10% of requested exposure t
QUALITY	Minimum desired quality value e.g. 1
MATCH	Match pattern string e.g. 'glob "DA=4 FS=[12]"'
VALUE	Destination directory path (in IRAF network notation) e.g. 'pipedevn!/pipelines/fpierfed/MarioC

8 Request for a Calibration

8.1 The Request Options

The values of many of the constraint parameters may be preceded by '=', 'LIKE', or 'GBLOB' to select how the SQL query constraint is constructed. IF absent the default is 'LIKE' for the class and filter and '=' for the others. A note about the filters is that for the NOAO Mosaic the filter names have a serial number and so filters are normally match to the serial number because the string part of the name can vary.

8.2 Calibration Selection

The heart of the calibration library is the selection of the best available calibration satisfying the request. This can be made as elaborate as desired. In our current library we use a relatively simple logic that can be represented as an SQL query to the library catalog. It is simple but still more sophisticated than just selecting the nearest in time. It also supports the ability for the operator or pipeline scientist to force a selection or contribute their knowledge of the calibrations by a numerical quality value.

After the set of request parameters has been received by the calibration library server an SQL query is dynamically built. By dyanically we mean that certain parts of the query depend on the presence of request parameters and their values. In our case it is mostly a matter of adding a constraint for each requested parameter. Figure **??** gives the schematic query. The lines in square brackets are included only if the dependent quantity was included in the request. Also, the declaration of dexptime is as shown only if an exposure time is provided, otherwise it is declared as "0.0 AS dexptime".

The selection logic is essentially the ORDER clause. The first value in the ordered list is

Figure 1: Schematic query to select a calibration.

```
SELECT value, datatype, class, mjdstart,
    <mjd> AS mjd, <mjd> - (nhdstart + mjdend) / 2 as dmjd,
    ABS (exptime - <exptime>) AS dexptime
    FROM catalog WHERE
    class <op> <class>
    [AND detector <op> <detector>]
    [AND imageid <op> <imageid>]
    [AND filter <op> <filter>]
    [AND ABS (dmjd) <= <dmjd>]
    [AND exptime <= <dexptime>]
    [AND quality >= <quality>]
    [AND match <op> <match>]
    AND mjd >= mjdstart AND mjd < mjdend
    ORDER BY quality DESC,
        class,
        MIN(2,ncombine) DESC,
        dexptime, ABS (dmjd), dmjd
```

selected. The ordering is first by quality. So the highest quality is always chosen. The class ordering is to select class ”dflat” (dome flat) before ”tflat” (twilight flat) when the request is ”LIKE to make a master calibration. The preference is for the highest value. The minimum function can be used to treat all calibrations with number combined lower than some value as equal. We experimented with five but currently feel even three is better than two or one. The next ordering implements te closed in exposure time if a target exposure time is given. If one was not supplied dexptime is zero. The last two parts of the clause select the nearest to the requested modified Julian date with preference given to the earlier one if two are equidistant.

The returned class and mjdstart are used in the server to build the path to a file in the calibration repository.

8.3 The Return from a Calibration Request

The possible return values for a calibration request in our standard protocol are shown in figure **??**. The standard protocol only requires the return a status code and a value. Obviously, in this calibration request the reutrn value is the desired calibration. If no calibration was found the status code indicates an error and the value is a description.

The optional return values are for file calibraions. The parameters are the type of file (’file’, ’image’ or ’directory’), the network path (URI) to the file in the calibration library, the last modify time, and the size. The last two are used, as described earlier, for the requesting client to decide whether they have a current local copy.

Table 5: Return values from a regquest for a calibration

STATUS	'exit_code getcal'
VALUE	The value of the catalog entry (depends on the data type) e.g. 'Mosaic2_CAL9906_6_bpm.fits[1]'
TYPE	The type of the item one of 'file', 'image', 'directory', etc.
PATH	The IRAF path to the item
MTIME	The last modification time of the item
SIZE	The size of the item in bytes

Table 6: Parameters for Entering a Calibration

R	CLASS	The type of calibration we want to ingest. e.g. 'Zero'
R	MJDSTART	Modified Julian Day. e.g. 52642.4
R	MJDEND	Modified Julian Day. e.g. 52642.4
	DETECTOR	The string identifying the detector (optional) e.g. 'Mosaic2'
	IMAGEID	Image/Amp ID (optional) e.g. 'SITe #98164FACR10-02 (NOAO 21), lower left (Amp13)'
	EXPTIME	Exposure time (optional) e.g. 120.
	NCOMBINE	Number of images combined (optional) e.g. 1
	QUALITY	Image quality (optional) e.g. 0.
	MATCH	Match string (optional) e.g. 'DA=4 FS=16'
R	VALUE	Destination directory path (in IRAF network notation) e.g. 'pipedevn!/pipelines/fpierfed/Ma
R	DATATYPE	Type of data to ingest e.g. file

As noted earlier, the calibration value is often associated with an image as a keyword value. The requires the implementation to limit the length of the filename and the supplied path to fit in the maximum length of a keyword value. This is done by using a very short logical path in the input request ('MC$') and paying attention of the file names.

9 Entering a Calibration

The server interface is primarily intended for use by the pipeline modules (through the interface tool). While operator interfaces can be built to work through the calibration library server typically the operator will use the database client or scripts written to talk directly to the database.

The parameters are obviously the fields of the catalog schema.

The required information are the calibration class, the interval of validity as modified Julian dates, the datatype of the calibration data, and the value. Optional parameters are used as constraints in later requests. The value and optional fields are inserted or replaced in the calibration catalog. For the 'keyword' datatype that is all that is done.

Figure 2: GETCAL and PUTCAL parameters.

. . .

The file datatypes copy the specified calibration to the calibration file repository. To store a file the MJDSTART value is rounded to an integer. This value is used as subdirectory in the class directory given by the value of the CLASS parameter. The directory is created if needed. Then an copy command is executed to copy the specified file given by the VALUE parameter to the library directory. The copy command can be datatype or host dependent. In the current server rsync is used in all cases.

The return from the server is a status value and a description string. The only two results are success or failure.

10 The Pipeline Interface Commands

To provide a single point of interface and to provide some useful features the pipeline modules which request or submit calibrations use the interface commands getcal and putcal. These are IRAF command because the NOAO pipeline applications are predominantly composed of IRAF modules. The IRAF parameter mechanism is also well suited to this type of application.

10.1 getcal

The calibration library server is currently part of a larger server called the Data Manager. The data manager port connects the tool to the server. The interesting point about this is that by having multiple servers, some on each node and one on a ”data manager” node, the pipeline application can optimize access based on the type of data. In current operations, a central calibration library is used.

The added features that getcal provides is

- filling in parameter values for the calibration query from an image header including parameters needed for different types of images
- storing the return value as a header keyword
- providing the return value, status, and string as both standard output and variables
- copying calibration files to a cache directory if needed
- do multiple queries for different classes in one execution or the same query over multiple images

Examples of category 1 is sending the modified Julian date of the exposure, the filter for calibration types that depend on filter, and the instrument and detector (e.g. amplifier or CCD) identifications. An example of the filter dependence is that requests for bias and dark calibrations are filter independent while flat fields are filter dependent.

Another interesting example in this category is filling in the match parameter with some arbitrary keyword that discriminates calibrations. For example, the NOAO Mosaic Imagers can be operated in single or dual amplifier modes. For instance, bias and dome flat calibration files are different. So the keyword identifying the mode is used.

While some queries are tied to specific image properties others are not. Therefore the use of images and keywords from thos images is only one mode of use. The explicit values may be given and not reference image specified.

The second category is an important feature for header driven processing. The calibration files to be used are selected by later pipeline modules from the headers. In the third category, it is sometimes desired to get the value as a variable and error branches in the modules are also simpler if a variable is filled in as opposed to capturing and parsing the standard output.

The last category was touched on previously. It is common that for many exposures being calibrated the same calibration file is used. So it is more efficient to get the calibration file once and place it in a local cache directory. In subsequent queries the routine can simple check the cache to see if it has to copy the file from the calibration repository. Each compute node will do this separately and so the same calibration may be cached multiple times though the pipeline parellization may be such that different nodes are dedicated to different CCDs over all exposures. Another aspect which can be controlled as needed is that distributed systems with shared filesystems can use the same cache (or possible work directly out of the calibration repository). The getcal tool does need locking logic to deal with parallel requests on the same node or across nodes in a shared file system implementation.

10.2 putcal

The putcal tool is similar. Given a list of calibrations to be added to the calibration library te attributes to be included in the call to the calibration library server can be set from keywords in a reference image or explicitly with static values or script variables. As with getcal the status code and string are returned as variables and as part of the standard output.

11 The Software

The calibration library server is written in python with libraries for the socket and database servives. The calibration catalog is implemented as an sqlite database. The pipeline interface tools are IRAF tasks. The auxilary tools are csh scripts using the sqlite command line client.

12 Auxillary Tools

There are some auxilary tools which are simple shell scripts that include direct interaction with the catalog database using the database client.

plrun, serverdm: Plrun is the highest level command-line tool for starting the NOAO pipeline applications. It is a script that calls various lower level tools to start up all the components needed for the pipeline application. One of this is starting up the calibration library server using the serverdm command.

mkdmcat: A script that

misc: calfiles, calquality, dmclean, mkdmcat, getdq, dbquery, mosnewfilt

13 Server-less Pipeline Situations

The NOAO pipeline applications are operated at NOAO in a server oriented mode. NHPPS is the primary execution framework which uses node manager servers on each node and RPC and resource discovery servers on specific nodes. It is therefore natural to have data manager, calibration library, and switchboard servers as part of the architecture. In this architecture, pipeline models within the work flow on disributed nodes talk to these servers which are at known addresses.

However, NOAO is also involved in collaborations which use ”grid” style architectures. In this structure use of services by individual modules is not desirable. There are ways to either use the service only on the head orchestration node or fold the functions into the pipeline interface tool. These approaches are only now being investigated. So the purpose of this section is to indicate that many of the concepts behind the calibration library can be applied in different architectures and some development is underway in this area.

makes use of various servers. The distributed nodes are part of a dedicated cluster and