Redesign of CDBS

S. Lubow, C. Cox, L. Hurt, B. Simon
Space Telescope Science Institute
Baltimore, MD

September 16, 1997

1 Introduction

The calibration database system (CDBS) has served several purposes at STScI. CDBS is the central storage repository of information about calibration data, which is stored in the DADS archive. It provides critical data to the SOGS pipeline that determines which reference files are used to calibrate each observation. More recently, it has been used to provide GOs with information about which reference files are currently recommended for recalibration of particular observations through the use of Bestref in Starview. CDBS consists of a database residing on Sybase, several software components for supporting the database, and interfaces to other STScI systems. The motivations for the CDBS redesign are described in "Report on CDBS Review" by Lubow and we do not repeat them all here.

The design presented here provides the following features:

- Simplicity of database structure. The new design replaces several dozen different types of database relations with only two types. In doing so, the software that manages this information is also simplified. Calibration tables and images are treated in a similar manner. The use of versions and generations of data will be eliminated in favor of a simpler scheme which should be easier to query. Moreover, the simplified database structure is able to accommodate new data tracking capabilities.

- Improved interfaces. The new design provides cleaner and more automated interface to OPUS and DADS. CDBS automatically updates its database when response files are received.

- Extended Bestref capability. Through Bestref in Starview, GOs will be informed about the level of change for recommended reference files since the original calibration. This advice will be quite specific for the observation mode being considered. They will also be informed about critical calibration software changes that would cause them to recalibrate.

- New comment capabilities. Calibration comments about each reference file and each table entry will be stored in the database. Sybase supports comments of up to 2 GB each. These comments will be made available to users through Starview and will provide a detailed history of calibrations files down to the level of particular observation modes. Comments can be updated by instrument groups at any time after delivery through the use of new CDBS tools.

- Predelivery tools. Currently, there is a validation process before the delivery of data to OPUS. That process involves the use of paper forms with signatures from instrument

scientists. With the new system, the instrument groups will carry out all their verification of the data before submission to CDBS. This will simplify the dataflow through CDBS. The new CDBS will provide instrument groups with tools that will

- check the validity of data files
- generate a load file, which describes to data being delivered to CDBS. The load file must be subsequently edited by the instrument groups.
- expand load files to automatically include modes specified by wild-cards
- regenerate previous load files

Various other improvements are described, such as a needed optimization to the Bestref algorithm, and are found in the various sections below.

2 Configuration

CDBS will interact with OPUS, DADS, and internal science support groups (i.e., instrument teams) through several mechanisms (see Fig 1), as will be described in more detail later. The CDBS database will reside on the the same database server as the DADS archival server. Some advantage will be made of the SOGS firewall in providing communications from OPUS to CDBS. CDBS will communicate with OPUS and DADS through file transfer mechanisms. Calibration data will be copied to publicly accessible directories on the unix science network and the VMS science cluster.

3 Instrument Relations

3.1 Overview

The database schema of calibration data for instruments, exclusive of Synphot data and multi-instrument data, follows a simple set of rules.

- Each instrument has a separate set of relations. The main reason for this separation is that each instrument has different mode information. Other than differences in mode, the relations are identical for all instruments. The templates for each relation are specified in sections 3.2 and 3.3 (see also Appendix A).
- Each instrument has two associated CDBS relations. One relation, called the file-level relation, deals with general information about each calibration file. More detailed information is in the second relation, called the row-level relation.
- Every instrument calibration file delivery will add to both the file-level relation and the row-level relation. A delivery of a nondummy image reference file will add a single record to the row-level relation and to the file-level relation. Each table delivery will generally add a record to the row-level relation for each row in that table, as well as a single record to the file-level relation. Each dummy file delivery adds a file-level and a row-level record for each mode in which the file is used.

- The delivery process includes a set of information contained in the load file, which is used by CDBS for populating its tables.

3.2 File-Level Relation Template

The following fields are in the file-level relation for each instrument, together with the SQL data type, and a brief description.

- delivery_number - int not null. Set during load to CDBS, and determined by CDBS tool *cdbs_sql_gen*. A delivery is a set of files that are loaded together into CDBS. This set of files could involve several calibration tables or images. We associate each delivery with a delivery_number. Several entries in this table could have the same delivery number. There is a 1-to-1 correspondence between delivery number and general_availability_date. This field is used as a convenience to avoid the need for specifying the exact Sybase date to determine the contents of a delivery.

- reference_file_type - varchar(50) not null. Set during load to CDBS, and determined by contents of load file. It is automatically generated in the load file by CDBS tool *mkload*. This field indicates the category of the file being delivered, formerly called parm_id, for example, DRK for WFPC2.

- file_name - varchar(50) not null. Set during load to CDBS, and determined by CDBS tool *uniqname* which follows naming conventions found in ICD-47. This field specifies the name of the calibration header file or table being delivered. This field, together with field expansion_number, are used as the joining attributes with the file_name and expansion_number attributes in row-level relation, described in the next subsection. The join is 1-to-1 in the case of image reference files and 1-to-N in the case of calibration tables.

- expansion_number - int not null. Set during load to CDBS, and determined by CDBS tool *cdbs_sql_gen*. This field is useful in the case that a dummy image reference file is delivered that applies to several modes. In that case only, there will be more than one record in this relation having the same file_name. For a given dummy file, each value of the expansion_number corresponds to a different observation mode.

- useafter_date - datetime not null. Set during load to CDBS, and determined by the contents of the load file. It is automatically generated in the load file by CDBS tool *mkload*. The reference file can be used for calibration of observations made after this date. Generally, the appropriate calibration file for an observation is the one for the same instrument and mode that is not rejected (see reject_flag below) and whose useafter_date is closest in time, but earlier than the date of the observation

- opus_flag - char(1) not null. Set during load to CDBS, and determined by contents of the load file. The value is entered in the load file by instrument groups. This field has values "Y" or "N" to indicate whether the file is used in the pipeline by OPUS. The main reason for specifying this value is to determine which entries need to be made known to OPUS.

- opus_load_date - datetime null. Set in CDBS when information is received from OPUS (in the form of a feedback file) to indicate that CDBS data was loaded into the OPUS database. It is set by CDBS tool *update_opus_load_date*. Before delivery to OPUS, CDBS generates an SQL file for use by OPUS in updating their database for the entire delivery. This SQL file includes instructions to generate a load time. OPUS will process this SQL file and the output of this processing becomes the feeback file, which contains the status of the attempted updates to OPUS, as well as the opus_load_date. OPUS will send the feedback file for the entire delivery to a designated CDBS directory. CDBS software will periodically read this directory and parse the contents of the feedback file. CDBS software will inform CDBS personnel via email of the whether the update was successful. If the feedback file indicates that the update of OPUS was successful, then CDBS software will set the opus_load_date field for that calibration file. If data exists in CDBS for which the general_availability_date is more than a threshold number of days in the past and the opus_load_date is still null, then a warning will be issued to CDBS personnel.

- comment - text not null. Set during load to CDBS, and determined by the contents of load file. The comment is put in the load file by instrument groups. It is an associated description of the file being delivered. Instrument groups can update comments at any time after the data has been delivered (see section 8), by using CDBS tool *recreate_load_file*. The receated load file will be edited by instrument groups and then be processed by CDBS personnel, using CDBS tool *comments_sql_gen*. It is important that only general information about the file be stored in this field. Any mode-specific comments should be stored in the row-level relation.

- archive_date - datetime null. A date on which it is known that the calibration file has been archived in DADS. This date is set by CDBS tool *update_archive_date*. Following delivery of data to DADS, DADS will send a response file to a designated CDBS directory. CDBS software will periodically read this directory and parse the contents of these files. If the response file indicates that a calibration file was successfully archived, then CDBS software will set this archive_date field for that calibration file. In any case, CDBS software will automatically send email to CDBS personnel about the status of the attempt to archive the file and whether the CDBS update of this field was successful. If data exists in CDBS for which the archive_date is more than a threshold number of days in the past and the archive_date is still null, then a warning will be issued to CDBS personnel.

- general_availability_date - datetime null. The value for this date is set by CDBS tool *update_ga_date* just prior to the delivery of data to other systems. This date is a timestamp of the current time. All database records for a particular delivery are given the same value of this field.

- reject_flag - char(1) not null. Set initially during the load of data to CDBS to a value of "N" by CDBS tool *cdbs_sql_gen*. This field has values of "Y" or "N" to indicate whether the data for this file should not be used (i.e., rejected). Even if the data is rejected, the entry for this file remains in the database as a historical record, so that all calibration

files ever used will be tracked in CDBS. If there already exists nonrejected data for a given instrument with the same reference_file_type and useafter_date (and also mode in the case of images) as the data being loaded, then a Sybase trigger in CDBS will automatically set the reject_flag to "Y".

- rejected_by_file_name - varchar(50) null. The value is set initially to null by a Sybase default. It is the name of the file that rejected the file described by this database record. The field is updated by the same Sybase trigger that sets reject_flag.

- rejected_by_expansion_number - int null. The value is set initially to null by a Sybase default. It is expansion number of the file that rejected the file described by this database record. The field is updated by the same Sybase trigger that sets reject_flag.

- comparison_file_name - varchar(50) not null. Set during load to CDBS, and determined by the contents of the load file. A suggested value for this field is automatically entered in the load file by CDBS tool *mkload*. This field specifies the name of the file relative to which levels of change in the data described by this record are determined (see section 3.4). The levels of change are encoded in the row-level relation equivalence class fields.

Note that the datetime datatype in the above is the Sybase date/time datatype, accurate to 3.3 milliseconds.

Data can be deleted from this relation and the row-level relation, before the general_availability_date is set to a nonnull value. Only entire deliveries can be deleted from CDBS. A tool called *delete_delivery* will be provided for CDBS personnel to delete data by delivery number. After the general_availability_date is set, data cannot be deleted, but only marked as rejected, using the rejection fields described above.

3.3 Row-Level Relation Template

This relation stores detailed information about each file delivered. There is generally one record in this relation for each nondummy calibration image delivered, while in the case of table deliveries, there is one record for each table row. In the case of dummy calibration images, there will be one record in this table for each mode that applies to it. The row-level relation fields are

- file_name - varchar(50) not null. Set during load to CDBS, and determined by CDBS tool *uniqname* which follows naming conventions found in ICD-47. This field specifies the name of the calibration header file or table being delivered. This field, together with field expansion_number, are used as the joining attributes with the file_name and expansion_number attributes in the file-level relation, described in the previous subsection. The join is 1-to-1 in the case of image reference files and N-to-1 in the case of calibration tables.

- expansion_number - int not null. Set during load to CDBS, and determined by CDBS tool *cdbs_sql_gen*. This field is useful in the case that a dummy image reference file is delivered that applies to several modes. For a given dummy file, each value of the expansion_number corresponds to a different observation mode.

- observation_mode - null. Set during load to CDBS, and determined by the contents of the load file. The mode values are automatically entered in the load file by CDBS tool *mkload*. This is really a set of fields for each instrument. These fields differentiate the row-level database relations. The observation modes are described in section 3.5. Any individual reference_file_type will only use a subset of these fields. Unused fields are given special default values of -1 in the case of numerical fields or blank strings in the case of character fields.

- equivalence_class - not null. Set during load to CDBS and determined by the contents of the load file. The value of the level of change (SEVERE, MODERATE, or TRIVIAL) for each row must be specified in the load file by instrument groups. The CDBS tool *cdbs_sql_gen* uses the comparison_file_name and translates these levels of change into equivalence_class values, by applying the algorithm described in section 3.4.

- pedigree - varchar(50) null. Set during load to CDBS and determined by the contents of the load file. The value of this field must be entered in the load file by the instrument groups. This value is required for OPUS data. It indicates the source of data. Possible values are INFLIGHT, GROUND, DUMMY, and COMPUTED. INFLIGHT indicates a measurement done on HST, GROUND indicates ground-based measurements, DUMMY indicates no measurement (dummy file), and COMPUTED indicates that the data was the derived from a model, such as PSFs and LSFs.

- observation_begin_date - datetime null. Set during load to CDBS and determined by the contents of the load file. The value of this field must be entered in the load file by the instrument groups. It is the date at which this calibration data began. We leave open the option that it may not be set by permitting a null value. It is considered mandatory for inflight data.

- observation_end_date - datetime null. Set during the load to CDBS and determined by the contents of the load file. The value of this field must be entered in the load file by the instrument groups. It is the date at which this calibration data ended. We leave open the option that it may not be set by permitting a null value.

- comment - TEXT null. Set during the load to CDBS, and determined by the contents of the load file. The comment must be entered in the load file by the instrument groups. It is an associated description of the row-level data, provided by the instrument groups. Instrument groups can update comments at any time after the data has been delivered (see section 8). Row-level comments are not required by the database software. However, they are strongly recommended for each row where the level of change is MODERATE or SEVERE.

3.4 Equivalence Classes

Bestref currently determines the originally used calibration files and the currently recommended best ones. With the new system, it will inform users on whether the data that were used for the original calibration are "equivalent" to data now recommended for recalibration,

where equivalence is judged according to some criterion. The comparison for equivalence will usually be between entire files in the case of reference images, and individual table entries in the case of calibration tables. In some cases, it will not be possible to compare individual entries in the case of tables, but instead groups of entries or entire tables will be compared for the determination of equivalence. In short, the granularity (e.g., file, groups of table entries, etc.) of equivalence is determined by the mode attributes specified for each reference_file_type in the row-level relations. Discussion of modes is found in section 3.5.

Consider the case that equivalence means that the data are "nearly" identical, as judged by instrument groups. The purpose of equivalence classes is to determine whether any two calibration data values (or sets of data values) are equivalent, without needing to examine information about all intervening values. (We consider in the remainder of this section that "calibration data" is of the appropriate granularity for an equivalence class, e.g., a file in the case of reference images.) For example, suppose calibration data A, B, and C are delivered in time order. Suppose B is known to be equivalent to A, when B is delivered, and C is known to be equivalent to B upon its delivery. Bestref may need to know whether A, which was used in the original calibration, is equivalent to C which is now recommended. If the CDBS database stores the A-B equivalence in a record about B and B-C equivalence in a row-level record about C, then Bestref would need to examine CDBS data concerning intermediate data B to determine the A-C equivalence. This general transitivity problem can be solved by assigning all equivalent data to the same equivalence class, which we represent by an integer. In this scheme, suppose A belongs to equivalence class 1 in CDBS and an instrument group claims that B is equivalent to A, then B will also be assigned to equivalence class 1. Similarly, if C is deemed equivalent to the last delivery, B, it will be assigned to B's equivalence class of 1. Now when bestref compares the CDBS information on A and C, it finds that they belong to the same equivalence class, and are hence equivalent. If D is added after C, but is not equivalent to C, then we assign D to equivalence class 2.

The above scheme is appropriate for one level of equivalence, but several levels might be desired. Suppose that we accept a three level scheme such that

- Two data values (or groups of values) are equivalent unless there are severe differences between them. Nonequivalence (between the original and currently recommended data) at this level implies then the necessity of recalibration.
- Two data values are equivalent unless there are moderate or severe differences between them. Nonequivalence at this level implies a mild recommendation for recalibration.
- Two data values are equivalent provided they are nearly or exactly identical.

To support this scheme two equivalence classes are required. These are defined as

- equivalence_class_severe Any two calibration data values that differ in this quantity have severe differences between them.
- equivalence_class_moderate Any two values that differ in this quantity have moderate or severe differences between them.

Notice that if two calibration data items have the same value of equivalence_class_moderate, then they must have the same value of equivalence_class_severe. If two calibration data items

have the same values of both equivalence_class_moderate and equivalence_class_severe, then they must be nearly identical.

The load files provided by the instrument groups will indicate a level of change for the new data, designated as SEVERE, MODERATE, or TRIVIAL, respectively for the three levels described above. Indications of equivalence of reference data in CDBS is always relative to data of the same mode. The above scheme applies to data added sequentially in time with useafter_date values that increase with each delivery. However, deliveries can occur that apply with earlier values of the useafter_date than the latest value for some mode. The question then arises as to what data the indications of change refer. The answer is that indications of change are relative to data, which is termed the *comparison file*, with the closest preceding or same useafter date. If there is more than one reference file that meets this criterion, then the file with the latest general_availability_date, independent of whether the data was rejected or not, is selected.

In some cases, however, it may be desirable to choose a different comparison file. The most frequent example of that situation is when some file A has been delivered, and later file B is delivered to supersede file A (i.e., it rejects file A in CDBS) and file B has file A as a comparison file. However, file B contains a serious error and subsequently file C file is being delivered to correct that error. In that case, it can be seen that file C would best choose file A as a comparison file, rather than file B.

The algorithm for assigning equivalence classes is tabulated as follows.

Indicator	equivalence_class_severe	equivalence_class_moderate
SEVERE	+	+
MODERATE	0	+
TRIVIAL	0	0

In the above, + means to store the value of a class as the incremented value from the last used value, while 0 means to store the same value as before.

Upon reading values of these two equivalence classes for two data items, the determination of the level of severity of the differences between them is as follows.

Level	Δ_S	Δ_M
SEVERE	nonzero	nonzero
MODERATE	0	nonzero
TRIVIAL	0	0

In the above, Δ_S and Δ_M are the differences in the values of equivalence_class_severe and equivalence_class_moderate, respectively between the two data items. So that for example, if the values of equivalence_class_severe are equal, while the values of equivalence_class_moderate differ, then the two data items are moderately (M) different.

In this scheme then the row-level relations would contain two attributes for equivalence class.

- equivalence_class_severe - integer not null, with meaning described above. Set upon data load by CDBS tool *cdbs_sql_gen*, based on value of level of change indicated in the load file (SEVERE, MODERATE, or TRIVIAL).

- equivalence_class_moderate - integer not null, with meaning described above. Set upon data load by CDBS tool *cdbs_sql_gen*, based on value of level of change indicated in the load file.

3.5 Modes

We define a full-mode of a reference_file_type for an instrument to be a minimal set of parameters that are needed to uniquely specify a table row, in the case of reference tables, or to uniquely specify a reference file, in the case of reference images. For example, optcrly and pxfmt constitute the mode for FOC files of reference_file_type bac. The mode used within CDBS determines the granularity of the information tracked by CDBS for each reference_file_type. In some cases, the full-mode will not be used by CDBS for some reference_file_type(s) of calibration table data, due to practical limitations. Instead, a portion of the full-mode (partial-mode) will be specified, which can result in a CDBS record of the row-level relation referring to more than one calibration table data row. We use the term "mode" to refer to the actual mode used by CDBS, in contrast to the full-mode described above.

The row-level relation contains information about the appropriate mode. The mode information is used to select row-level information, based on data in the header. For example, with this mode information the appropriate equivalence classes for the recommended best data and previously used data can be determined, so that the level of need to recalibrate can be determined, as described in the last subsection. Each instrument row-level relation uses a different set of mode attributes. For each row-level record, only a subset of the mode attributes are used. This subset depends upon the reference_file_type for the file. The remaining attributes for the instrument are set to special default values.

For various reasons, in some cases CDBS will not track all attributes that define a full-mode. However, even partial mode information can be used effectively to determine whether recalibration is required. As a result, partial modes lead to a less precise determination of whether to recalibrate. In general, CDBS will not track data using parameters that take on a very large number of possible values, such as unrestricted floating point numbers. Therefore, parameters that are floating point numbers are generally not tracked.

In Appendix B, we list tables of modes as a table for each instrument and the corresponding datatype for each mode parameter. Each row of the table corresponds to a single reference_file_type, which is listed in the first column of each mode table. The next column labeled as "t/i", indicates whether the reference file is a table (t) or image (i). The remaining columns list the full-mode attributes for each instrument. The attributes actually used in CDBS are listed with a plus sign. Each attribute that is in the full-mode, but not used in CDBS is marked by a minus sign. Blank column values are not part of the full-mode.

Bestref uses data in DADS (originally in the science header files) to determine the mode for a given observation. That mode information is used to determine the appropriate CDBS data. Group parameters, that is parameters that indicate properties of subsets of data within a file, require special treatment because their values are not generally stored in the data header. As a result, Bestref must make inferences about the group parameter values for a particular observation. For some parameters, Bestref will determine their values by using

the file name of the original calibration file. Using this file name, Bestref issues a query to the CDBS database to determine the values of these parameters. For other parameters, it is known that all possible values are used.

3.5.1 FOC

All FOC data will be tracked using full-modes.

The uni table mode includes a wavelength field found in the science header file with keyword PHOTLAM. That wavelength value is determined by Synphot software. We will assume for now that this software does not change and so CDBS does not recompute its value. A later enhancement to this system might include this recomputation. Bestref will use the proper algorithm to select the nearest wavelength value used as a mode for uni. The table unitab is used to select an appropriate uni file.

3.5.2 HSP

For each observation, calhsp uses the real aperture name in ccp9 first to get translated into a "dark aperture" name. That name is used in table ccp5 to get the dark counts to be used in the pipeline. Bestref will not perform the latter step, so tracking is done at the full table level only by ignoring modes. All other HSP data will be tracked using full-modes.

3.5.3 FOS

Fields aper_pos and pass_dir are group parameters which will be used in CDBS modes, since separate files are delivered for different values of these fields. Bestref will query CDBS using the original calibration file name to determine the aper_pos. Also, Bestref will report on both pass_dir values, since both are generally used, when needed. Fields gm_lat and gm_long are also group parameters which generally span a large range of values within an observation. They will not be tracked by CDBS. As a result, table cyccs8r uses only the detector field as its CDBS mode. The focus field will not be tracked because it is only used on pre-costar data. Also the wavelength field is not tracked because it is not in the header and is determined by software.

3.5.4 HRS

Field carpos, the carrousel position, takes on many different values that vary from delivery to delivery and so will not be tracked by CDBS. Fields line1, line2, sample1, sample2, and ydef are floating point values that will not be tracked by CDBS.

3.5.5 WFPC

Field detector will not be tracked because it is a group parameter. All other mode fields will be used by CDBS. Field FLATFIELD is not in the header, but its value can be inferred from the value of header keyword FLATCORR.

3.5.6 WFPC2

All WFPC2 data will be tracked with full-modes.

3.5.7 STIS

All STIS data will be tracked with full-modes. In the case that the mode contains continuous variables, such as wavelength, STIS will determine a set of standard discreet values for calibration purposes. New values can be added to this set at any time, but values cannot be deleted from this set. Observational data for STIS will provide a header value that corresponds to the calibration data actually used (obtained by some nearest value algorithm).

3.5.8 NICMOS

NICMOS reference data will be tracked by full-modes, except possibly for BACKTAB.

4 Load File

Each delivery of a reference image or table contains a whole set of data. In particular, each table delivery contains all data in that table, independent of whether individual rows in that table are changed by the delivery. To accompany each delivery of a reference image file or table, there will be a load file, which describes the data file being delivered. This load file is used by CDBS to populate its tables. The definitions of the quantities listed below are found in sections 3.2 and 3.3 or are described below (see also Table 3 of Appendix A). The load file contains in a header the following information

- file_name - name of the reference file being delivered
- reference_file_type
- instrument - name of instrument (foc, hrs, etc., and also synphot and multi).
- useafter_date
- opus_flag
- comparison_file_name - the name of the reference file relative to which the level of change is determined.
- comment on the entire file being delivered

All of the above is mandatory information.

The load file entries contain for each CDBS mode, appropriate for the reference_file_type, the following information.

- observation_mode (required)
- pedigree (required for OPUS data)
- observation_begin_date (mandatory for inflight data, otherwise optional)

- observation_end_date (mandatory for inflight data, otherwise optional)
- comment on the data for that mode (optional)
- level of change - SEVERE, MODERATE, or TRIVIAL (required).

Notice that in the event that the mode does not specify a unique row in a table, but rather a group of rows, then the above information must apply to that group. For example, the level of change needs to be assessed for the group of table entries that correspond to the particular mode. Some values, such as pedigree, may not be easily stated in this scheme.

The *mkload* tool has been developed to create a template for the load file. The tool takes as input the header file name and provides as output a partially complete load file. The load file will automatically contain the following data

- reference_file_type
- instrument - name of instrument
- useafter_date
- comparison_file_name - the name of the reference file relative to which the level change is determined. In some cases, it may be desirable to change this value (see discussion in section 3.4).
- observation_modes

The instrument scientist must then determine

- opus_flag (required)
- comments on the file (required)
- comments on the individual row-level entries (optional). These are strongly recommended for rows whose changes are not TRIVIAL.
- pedigree (required for OPUS data)
- observation_begin_date (mandatory for inflight data, otherwise optional).
- observation_end_date (optional)
- level of change - SEVERE, MODERATE, or TRIVIAL (required).

The load file will not be archived in DADS, but can be regenerated by CDBS tool *recreate_load_file*, which takes the file name as a parameter. The load files are not being archived, so that their comments can be easily edited (see section 8) and stored in the CDBS database.

5 Synphot

The tables for the Synphot system will be tracked by CDBS. The CDBS information about the graphtab, comptab, and throughput tables will be stored in a single set of two database relations, of the same form as the file-level and row-level relations used for the instrument relations. Comptab files are created automatically by CDBS through task mkcomptab with each delivery of a throughput file and will not be tracked for level of change. The graphtab and component throughput tables will be tracked for level of change.

Tracking of all three types of tables will be at the file-level only. Therefore, there will be a single row added to the file-level and row-level relations for each file delivered, just as the case for reference images. Differences between the instrument and Synphot relations are

- reference_file_type - possible values are GRAPHTAB, COMPTAB, and THROUGHPUT.
- file_name - The throughput file name will continue to follow the Synphot conventions. Graphtab and comptab files will follow naming conventions found in ICD-47, as is used by the other CDBS files. Synphot software will be modified to accept either convention, when searching for the most recent comptab and graphtab files.
- compname - The only mode field for Synphot data. Only the throughput files specify a mode value. The mode consists of the component mnemonic, as known to comptab, e.g., FOS_G160L.
- equivalence_classes - required for throughput and graphtab files. The comptab files are not reported upon by Bestref and equivalence class data will not be tracked for them.

Recall that the mode information determines the level of granularity for equivalence classes. So the mode information plays an important role in this scheme.

CDBS needs to run Synphot software to determine the throughput file names to calibrate a given observation. To make this determination, Synphot reads through the contents of the appropriate graphtab and comptab tables. We do not want to reproduce the throughput file selection method within CDBS and will therefore use the Synphot software. A Synphot function exists which accepts as parameters the graphtab and comptab file names, as well as the observation mode. The function returns the names of the appropriate throughput files.

For a given observation, Bestref will first obtain the originally used comptab, graphtab names, which are now stored in DADS, together with the observation mode in order to make a Synphot function call that determines the original throughput file names (unfortunately not now stored in the header files or DADS). Using this file name, the row-level Synphot relation will be queried to determine the appropriate equivalence class values and corresponding CDBS modes (component names).

The currently recommended throughput file names must also be obtained by Bestref. To do this, the row-level Synphot relation will be queried to determine the comptab and graphtab table names having the closest useafter_date that precedes the date of the observation, which were not rejected. Using the newly determined comptab and graphtab table names, a Synphot function call will again be made to determine the recommended throughput files.

The equivalence classes of these throughput files can then be compared with the equivalence classes of the originally used throughput files of the same mode to determine the level of severity of changes, as described in section 3.4. If the modes (component mnemonics) do not match from the original to recommended throughput files, then Bestref will need to issue a warning and/or consider the change as severe.

In practice, it is advantageous to store both the original and the recommended file names within a database for each throughput file mode of each observation to speed Bestref queries. The original and recommended throughput file names and modes (component names) can be determined when the new CDBS system is initialized. These file names and modes will then be stored in a database for each observation. (The mode can be inferred from the throughput file name.) Any further needed changes to the names of recommended throughput files can be accomplished as new throughput files are loaded into CDBS. Using both the mode information and the useafter_date, the set of possibly effected observations in DADS can be determined. A time interval of possibly effected observations can be determined by looking for the next future useafter_date for reference files with that same mode. (Notice that if the mode no longer exists, then the endpoint of the interval becomes the present. Here we assume that a change in a component mnemonic means that the old mnemonic is no longer used at all, i.e., not used for a subset of the data for which it was originally used.) From that restricted list of observations, the above-described process of determining the newly recommended throughput file names via Synphot can be run with modest effort as a daily batch job. The recommended throughput file names can then be updated and used in Bestref.

6 MULTI Data

Some categories of data are used by several instruments and are never used in the OPUS pipeline. This MULTI data contains information such as spectra of standard stars. In Appendix B, we list the current MULTI table names (stored as reference_file_type in CDBS) and corresponding modes. For the new CDBS scheme, we will maintain two database relations that track all the MULTI data. These MULTI relations are of the same form as the instrument relations described earlier. Only file-level tracking will be performed. Therefore, there will be a single row added to the file-level and row-level relations for each file delivered, just as the case for reference images. The points of difference are that

- useafter_date - not required
- opus_flag - should always be "N"
- equivalence_class - not supported
- pedigree - not required
- mode - (still required) is defined for this data in Appendix B.

A drawback to this scheme is that adding new modes will require creating a new row-level relation and copying the old data into that relation. However, this need should not arise very often and the database relations should not be very large.

The load file is created by CDBS tool *mkload* and is similar to that used by the instrument relations, except that

- useafter_date - not required
- opus_flag - not allowed, since assumed to be set to "N".
- comparison_file_name - not allowed
- level of change - not allowed

7 Data File Packaging

CDBS must determine whether a file being delivered replaces an existing file. To make this determination, CDBS requires some additional knowledge of how files are packaged. To determine whether *image* file being delivered replaces an existing image reference file, CDBS uses the instrument name, the reference_file_type of the file, the useafter_date and the mode (keyword) values for that image. CDBS software searches the CDBS database for an existing file with these properties. If it finds one, it marks the existing file as rejected. The CDBS software then inserts into the database the data for the new file as being the currently recommended file (for that instrument, reference_file_type, mode, and useafter_date). In the case of tables, CDBS applies the same algorithm, except that generally the mode information is not used because files of a given type are assumed to contain all possible mode values for that reference_file_type. The only exception to this rule has been for STIS.

From the point of view of CDBS, instrument calibration files should be packaged as described below.

1. The same modes (keywords) are used to describe all files of a given reference_file_type. For example, all FOC files of type BAC depend on both OPTCRLY and PXFORMT in all cases. Less desirable would be, for example, the case that BAC files sometimes depend on OPTCRLY only. In that case, we recommend that a new reference_file_type be defined that depends only on OPTCRLY.

2. All tables of a given reference_file_type contain all possible rows for that type. For example, the ccr1 table for the HRS contains all rows and so all possible mode values for that type of table. Less desirable would, for example, be the case that the ccr1 tables are not complete, but are delivered separately for each detector. In that event, we recommend that new ccr1 reference_file_types be defined, one for each detector.

3. The mode values should preferably not take on a nearly continuous range of values. The use of REAL or FLOAT values for modes should be avoided, if possible. The reason is that CDBS will not use such fields to track the level of change of reference files (see section 3.4).

4. Values of individual mode fields (keywords) should represent a single, simple value and not a concatenation of several lower level values.

5. The modes (keywords) that characterize a file of a given reference_file_type remain fixed in time.

We urge that these rules be obeyed for new or existing instruments. If it is impractical, then CDBS can be modified for special situations.

In the case of STIS, rules 1 and 2 are not always followed. Regarding rule 1, the STIS header always supplies a special default value for modes that are not used in some file of a particular reference_file_type, but are sometimes used. For example, in the case of PFL files, only APERTURE or CENWAVE (but not both) is meaningful in any given file. STIS supplies a special default value for the other field, so that CDBS software can process this data properly. Regarding rule 2, for some tables, STIS uses some mode fields to select the table, in addition to instrument, reference_file_type, and useafter_date. Such fields are marked as "T" in the STIS dependency table of Appendix B.

8 Software Tracking

In addition to tracking changes to calibration data, we propose tracking serious changes to calibration software. Changes to both are important in determining the need to recalibrate. Of course, software is inherently more complex than data and so a software tracking scheme cannot provide the same level of completeness as for data. In the case of software, the latest version is always the most appropriate, unlike the case of data. The level of effort to track software in our proposed scheme should be minor.

8.1 Software Relation

The software relation will track only information about serious changes to calibration software. The relation will contain the following

- instrument - varchar(50) not null.
- public_availability_date - datetime null. The date that the software was made available for external use as an STSDAS release.
- opus_installation_date - datetime not null. The date that the software was installed in the OPUS pipeline.
- public_version - numeric(10, 6) null. The version number of the software corresponding to the public_availability_date.
- opus_version - numeric(10, 6) not null. The version number of this software in the OPUS pipeline.
- comment - text not null. Description of change to software. This should include the calibration steps effected.

When serious calibration software changes are made to the OPUS pipeline, a record is entered into this relation, with all fields populated, except public_availability_date and

public_version. Those fields are set once the software changes are made public by an STSDAS release. Populating the relation could be done through a load file or through a forms interface. Based on previous experience, we expect only a few new entries per year. Examples of serious changes are FOS scattered light correction, FOC geometric correction, and the WFPC shadow correction.

We envision Bestref as reporting on software changes. For each observation, Bestref will provide a report on serious software changes based on the instrument and observation date. It will report on all changes for that instrument having an opus_installation_date that occurs after the date of OPUS processing. If the public_version field is null (because the software has not yet been released via STSDAS), Bestref will not report on the existence of the change. Otherwise, it will provide the version number for which (and after which) the data should be recalibrated. If new software significant software changes occur in the OPUS pipeline, but are not yet public, then we will warn users of this condition because they might recalibrate and get worse results.

We recommend that the opus_version, the version number used in the pipeline, be added to the header information of each observation. This additional information will make it easier for Bestref to determine whether a given observation was processed with a particular improvement of the software, rather than relying on dates. Better versioning conventions would help simplify tracking. The public versions should be simply related numerically to the OPUS versions. It would also be better if the calxxx software had its own versioning for each instrument, which users could determine.

9 Comment Updates

Instrument groups can update comments at any time using the CDBS tool *recreate_load_file*. This tool takes as an input parameter the file name for the comment(s) which will be updated. The tool recreates the load file for that data file, based on information contained in CDBS. Instrument groups should then update only the comment fields in the load file. The updated load files are then returned to CDBS personnel, who will run a CDBS task called *comments_sql_gen* to update the comments in the database.

10 CDBS Data Flow

We describe below how data will flow through CDBS (see Fig. 2)

- Delivery Preparation Step. Instrument groups prepare data for delivery to CDBS by carrying out the following steps:
  - The validity of the data files are checked by running CDBS tool *certify*.
  - A load file is created for each data file by running tool *mkload*
  - The load files are edited to provide additional information (see section 4).
  - Load files containing modes specified by wild-cards are expanded by running CDBS tool *explode*

  - The validity of the load files is checked by running *certify* on them.
  - The files are delivered to CDBS

- File Rename Step. Files are renamed by running tool *uniqname*.

- Comptab Step. If the file is a Synphot throughput file, then a comptab file is created, through CDBS task *mkcomptab*. A load file for the comptab file is created through CDBS task *mkload*. These files are verified by running *certify*.

- CDBS Install Step. CDBS task *cdbs_sql_gen* is run with the load files as input to create SQL scripts for updating the CDBS data. These scripts are then run through the Sybase isql tool to update the database. The output from isql is checked to verify that the database updates were successful.

- OPUS Preparation Step. CDBS task *opus_sql_gen* is run to create an SQL script for the entire delivery. This script will be run by OPUS to update their database. CDBS task *opus_catalog* is run to create the catalog file. This file provides a list of reference files to be added to or deleted from OPUS and some status information.

- FITS Conversion Step. All calibration data is converted to FITS format.

- Database Check Step. A tool, *check_cdbs*, will be run to verify that the database satisfies certain consistency requirements. For example, the tool will check that no two files were entered in the same delivery in which one rejects the other. If there are problems here, the delivery should be deleted from the database by using tool delete_delivery. The corrected data should then be rerun through CDBS as a new delivery.

- Setting of general_availability_date Step. This date is set in CDBS by running tool *update_ga_date*. If a problem is encountered with the above processing before the general_availability_date is set, then the entire delivery will be deleted from CDBS using the *delete_delivery* task in CDBS. Once this date is set, the data is permanently recorded in the CDBS database, i.e., no deletions are permitted. Thereafter, problem data is marked as rejected (see section 3.2) once a new delivery is made to supersede the problem data. At this point all appropriate fields CDBS database for this delivery are set, except for archive_date and opus_load_date.

- These steps go on in parallel

  - DADS Installation and Bestref Update

    * DADS Installation Step. Data is delivered to DADS. CDBS tool *update_archive_date* will determine whether data was successfully archived by parsing the DADS response files (see discussion in section 3.2). If the response file indicates that the data was successfully archived, then field archive_date is automatically set in the CDBS database. Email will be automatically sent to CDBS personnel to inform them of whether archive request and the CDBS database update was successful. The routine unix science data backups onto tape will serve as a source of data in the event that it is determined that the DADS copy of the

data is corrupt. Currently DADS checks that all archived CDBS data is identical (as a FITS file) to what was delivered. This check is typically carried out a few weeks after delivery. The unix science data backups will be retained for one year. If needed, the relevant data will then be loaded back into DADS by CDBS personnel.

  * Bestref Update Step. Once the DADS archive_date been been set, Bestref will update its internal data to reflect the presence of the new calibration data as being recommended. We expect these updates to occur on a daily basis. If new calibration data has been used in the pipeline, but the calibration data is not yet available in DADS, then Bestref may report that condition. This situation should not usually be a problem for users, since there should not usually be a need to recalibrate, in any case. For new calibration data that effects old observations, we will inform the users of the existence of the newly recommended data only after the archive_date is set (except for data newly processed in the pipeline).

- OPUS Installation
  * OPUS runs the SQL script to update their database and OPUS sends the SQL output file (the feedback file) to a known CDBS directory.
  * CDBS periodically polls this directory for the presence of new OPUS output files. CDBS task *update_opus_load_date* checks the on whether the SQL was processed properly by OPUS and if so, it updates the CDBS field opus_load_date for the delivery. Email is automatically sent to CDBS personnel on the results of the OPUS update and the CDBS update.
- Unix Science Network and VMS Science Cluster Installation Step. Data is sent to these systems and possibly checked. Once this step and the OPUS File Copy Step are complete, CDBS can delete its copy of the data.

11 Bestref

The primary goal of Bestref is to inform GOs of whether recalibration of the archived data in DADS is necessary and tell them which reference files are currently recommended. Bestref will continue to run as part of Starview. The knowledge of the originally used reference files for each observation by the OPUS pipeline comes from data headers and is stored in DADS.

11.1 Push-Pull Algorithms

Bestref must obtain information about the currently recommended reference files for each observation. Much of the work required to keep Bestref up to date will be precomputed daily, as is currently the case, rather than computed at the time of the user's request. However, the current scheme involves recomputing the recommended reference files for *all* HST observations on a daily basis. Bestref is then *pulling* data from CDBS. This scheme cannot continue to be used because of the CPU resources required, particularly in view of the added requirements for STIS and NICMOS. Instead, Bestref will determine a possible subset

of effected observations as new CDBS reference data is installed. This systems *pushes* changes onto Bestref. The set possibly effected observations is characterized by the instrument, reference_file_type, and time window for each CDBS reference file. The time window is defined as starting at the useafter_date for the file being delivered and ending at the next successive useafter_date for that instrument and reference_file_type. (A more complex variant of this scheme for Synphot data is described in section 5.) Bestref will be informed of the need to update this information by CDBS. All observational data that match these properties will have their recommended reference file information modified. These changes can be done daily with little overhead.

For newly calibrated observations, this push algorithm needs to be modified, since the changes to the Bestref's recommended data files may have been made (pushed by CDBS) during the calibration. Instead, new observations will use the old pull algorithm for determining the recommended calibration data. This process could be done daily with little overhead.

Bestref will maintain a set of tables which contains for each observation the necessary information about reference files to determine their level of change. The Getref tool can be made consistent with Bestref by using the same tables. In this way, Getref will always give the same result as Bestref and will not need separate code, apart from SQL.

It is not possible to share code between Bestref and Getref to guarantee automatic consistency. The reason is that Bestref changes require a recompilation of the Starview schema in QUICK and the modification of the Starview forms. In addition, Starview cannot issue queries through stored procedures. However, a manual procedure can be put into place to ensure consistency. The keyword committee will consider the impact of each proposed change on Bestref and Getref and will issue a request to both systems if modifications are needed.

With this new CDBS design, Bestref capabilities are considerably enhanced. Bestref will provide to users the following report on each observation.

- Reference file status - Bestref will list the originally used reference files and the currently recommended ones, as is now the case. In addition, for each such file it will report on the level of change between the files. This level of change will be reported with a fine level of granularity. For data tables, the level of change will be based on the individual table entry (in most cases) used by the observation rather than the table as a whole.

- Synphot file status - Bestref will give an overall recommendation on the need to rerun Synphot. This recommendation is based on the modes and corresponding throughput files actually used for the observation, but not the exact wavelengths of greatest importance to the observation within a throughput file. Bestref will not report on individual component throughput files to the user. Since Synphot internally selects the component throughput file names to be used, the names of these files are not of interest to the user. (No recommendations based on Synphot throughput tables is done now.) Bestref will also list the original and currently recommended graphtab and comptab table names. Only changes to throughput tables can drive recalibration. But if recalibration is done, then the recommended graphtab and comptab tables should be used by Synphot.

- Calibration software status - Bestref will report on serious changes to the calibration software that would drive recalibration. Only software changes that effect the instrument appropriate for the observation and which were not applied to the original calibration will be reported. The recommended version of the software which should be used will be reported.

Text comments will also be available with each of the above status information points.

It may be possible to extend Bestref to determine whether recalibrated data should be recalibrated again. This capability may be desired for data which is recalibrated on the fly. It should be possible to determine which calibration files were used to perform the last calibration by using the date and time on which the calibration was performed. CDBS and Synphot can then infer the set of reference files which were used. Alternatively, we believe a better method is to obtain the set of reference file names from the header file of the observation. We recommend that the throughput file names be put in the header of each observation for this purpose, otherwise Synphot software will need to be run to determine these file names. Once that set of files is known, the calculation of the levels of change between the recalibrated and the currently recommended reference files could be derived at run-time. This Bestref process could run remotely at the user's site and make appropriate database connections to STScI. The details have not yet been explored.

12 Interfaces

CDBS interacts with several STScI data systems. We list below the interfaces which will be required.

- OPUS - The OPUS Installation Step proceeds as described in section 9. CDBS will place the catalog file and the SQL file, along with the data files in a known CDBS directory. OPUS will be informed of this delivery and will copy these files across the SOGS firewall to their system. After OPUS processes the SQL file, OPUS will copy the output file (called the feedback file) to a known CDBS directory. The feedback file will contain the opus_load_date for use in updating the CDBS file-level relations (see section 3.2). In Appendix C, we list the current structure of the OPUS database tables and show its relationship to CDBS. In most cases the correspondence between CDBS and OPUS data fields is obvious.

- IRAF/Synphot - CDBS will need to run Synphot software to determine the Synphot files recommended for each observation. The dataflow steps in section 9 which open and read SDAS files will likely use STSDAS software.

- Unix science network - Instrument scientists will provide the load files to CDBS, based on a CDBS template tool. CDBS will deliver calibration data to the unix science network. Tape backups of this system serve as a source of data for redelivery to DADS in the event of data corruption in the archive.

- SDAS and CNS - Software tracking will be carried out through input from the SDAS group whenever major changes are made to calibration software. CDBS will also obtain

information currently stored by CNS on the delivery dates of calibration software to the OPUS pipeline. CDBS must know which date corresponds to each serious software modification to the OPUS pipeline and which instrument is effected. It must also know when that software was made publicly available and the corresponding STSDAS release number (see also section 7).

- DADS - The DADS interface is similar to the OPUS interface. CDBS will deliver data to DADS (which does not include the load file, opus_catalog or the SQL file). After DADS archives the files, DADS response files will be sent to a known CDBS directory. CDBS will determine that the files have been archived by an automatic mechanism which sets the archive_date field in the file-level relations (see section 3.2). DADS will verify that CDBS data was accurately archived and will inform CDBS in the event of data corruption. In that event, CDBS will redeliver the data using the unix science tape backup. Also Bestref/Getref will store the original and newly recommended calibration files in the DADS database, along with other information DADS stores for each observation.