Operated for NASA by AURA

Technical Instrument Report 2003-01

Revision to TIR98-12: Assessment and Delivery of Spectrographs Calibration Reference Files

Rosa Diaz-Miller
March 25, 2003

ABSTRACT

This TIR is a revision of TIR 98-12 which describes the standard procedures to be used by the Spectrographs group for reference file format checking and data assessment. This revision includes the new procedures and tools available to the scientist for testing and verification of reference files. As in the previous version, it establishes the Pipeline Group's responsibilities versus the instrument scientist's responsibilities.

Introduction

The generation of calibration reference files is the responsibility of the Instrument groups at STScI. As part of this process, standard preparation and assessment of the files need to be performed before they are officially "delivered" to CDBS (Calibration Data Base System). Until recently delivery signified an official handoff of the files from the Spectrographs group (STIS) to the CDBS specialists, who performed further verification and ingested the reference files and their associated tracking information into CDBS, DADS and OPUS databases. This task is now responsibility of each of the instrument groups, speeding the process of preparation and actual use of the reference files in the pipeline. The detailed steps and procedures for ingesting reference files in these databases will be described in a future TIR and here we concentrate only on their verification and assessment.

The ingest of the reference files in the OPUS database warrant their use by the OTFR pipeline. However, those are also ingested in the Data Archive Distribution Service (DADS) database which allows their retrieval using *StarView* via the option **best reference files.** The main function of the databases is to allow selection of the correct reference files for a specific set of observations. The selection is based on data file keyword values and the selection criteria outlined in ICD-47, located at ***http://www.sesd.stsci.edu/dso/icd47.pdf***

The checking and quality assessment of reference files will be performed by the person creating the files, while official delivery of the files to CDBS is handled within the Pipeline Group by the Reference File Lead Scientist and Reference File Lead DA. Spectrographs group members should send new reference files to these two people (see the **Pipeline** web page for Pipeline members in charge of various things). Checking and quality assessment are required for the following reasons: to ensure correct keyword syntax, file structure and format (specified in ICD-47); to ensure that the files will work properly in the pipeline with the calibration software; to avoid propagation of erroneous files and to ensure proper documentation of the reference files. Accordingly, we define in this document the standard procedure for checking and assessing the quality of STIS reference files. The procedures outlined here should follow creation of the reference file which is assumed to be in ICD-47 format. We first provide an outline of the steps to be performed, followed by a more detailed description of each step. Steps 2.1 through 2.8 should be performed by the person creating the reference files; steps 2.9 through 2.16 are performed by the pipeline block. If errors are encountered at any step they must be resolved before proceeding to the next step. Unresolved problems with the files or the software tasks should be referred to the Reference File Lead Scientist and DA.

Summary Checklist for Preparation and Assessment of Spectrographs Reference Files

To be done by the person creating the files

- 2.1 Configure the XSTIS and IDL packages in your environment.
- 2.2 Update all reference file headers properly (including pedigree, useafter, description, comment, nextend and history lines).
- 2.3 Rename working copies of the files to contain no capital letters.
- 2.4 Verify that the file is in standard FITS format.
- 2.5 Run the CDBS *certify* tool on the files.
- 2.6 Perform a detailed quality assessment of the reference file data. Run the Reference File Checking Tool or create and examine reference file paper products (if applicable).
- 2.7 Test the file in *calstis*
- 2.8 Fill out the delivery template and notify the Reference File Lead Scientist and Reference File Lead DA that the files are ready for delivery.

To be done by the Pipeline Group

- 2.9 Copy the files to a delivery directory.
- 2.10 Verify that the file is in standard FITS format.
- 2.11 Run the CDBS *certify* tool on the files.
- 2.12 Create the CDBS load file.
- 2.13 Populate keyword fields in the load file appropriately.
- *2.*14 *Certify* the load file.
- 2.15 Run *check_load* on the load files.
- 2.16 Run *uniqname* on the files and deliver the reference files to the OPUS, DADS and CDBS databases.
- 2.17 Verify the correct usage of the reference file in the operational environment.
- 2.18 Place a copy of the reference file in the OREF directory and update the history web page.

Detailed description of the procedure steps

2.1 Configure the IDL and XSTIS package in your environment

All of the IRAF tools referred to in the following steps are configured through the STIS package in the .envrc file. If you do not have this package set up, please refer to the information on how to do so on the STIS internal web Pipeline page under “How to Set up the xstis Package” (***http://www.stsci.edu/hst/stis/team/procedures/xstis_install.html***). After configuring the package STIS via your .envrc file, you also need to do the following to make the IRAF and IDL software tools, described below, available.

For the XSTIS package add the following lines to your iraf login file, then start your iraf session normally:

```
set xstis = /data/garnet2/xstis/
task xstis.pkg = xstis$xstis.cl
reset helpdb= (envget (“helpdb”) // “,xstis$helpdb.mip”)
```

load the xstis package (either via your login.cl or at the cl prompt by typing xstis).

For the IDL package: in your personal .setenv define the environmental variables to load the STIS IDL tools. This can be done adding the following lines:

```
setenv STIS_IDL /data/garnet2/idl
source $STIS_IDL/stis_cshrc
```

In this same file add to your IDL_PATH the directory /data/garnet2/idl. It should look like this:

```
setenv IDL_PATH +/usr/local/rsi/idl/lib:+/data/garnet2/idl
```

Now start your IDL session normally.

2.2 Update all reference file headers properly

Once the reference file has been created, it is necessary to update the primary headers with the proper useafter date, pedigree of the file, description, number of extensions, comment keyword, and history lines. These header keywords are crucial to the usefulness of the reference file and can be added using the IRAF task, **hedit**. The relevant header keywords are: PEDIGREE, USEAFTER, DESCRIP, COMMENT and NEXTEND. The USEAFTER and PEDIGREE keywords will most likely already exist in your header and need only be changed using **hedit**, DESCRIP, COMMENT and NEXTEND may need to be "added" by setting the add parameter equal to yes in **hedit**.

- PEDIGREE describes what type of data the file contains, INFLIGHT, GROUND or DUMMY. For the case of PEDIGREE = INFLIGHT the date range of the data used to create the file should be included. A typical pedigree looks something like: "INFLIGHT 21/01/2000 26/01/2000", with the dates in dd/mm/yyyy format. Note the four digit year. If the data of a reference files has more than one kind of pedigree (e.g. INFLIGHT and GROUND), set the keyword PEDIGREE to the one occurring more in the reference file, as only one kind of pedigree can be used in the header.
- USEAFTER represents the first date and time this file should be used to calibrate science data. The format for this keyword is month dd yyyy hr:min:sec, where any unique abbreviation for the month is acceptable (or write out the whole word). The time should be set equal to 00:00:00 for most of the reference files. Current exceptions are bias and dark reference files which are shouldn't be used across anneal occurrences and therefore their hr:min:sec are set equal to the anneal time. Therefore, a reference file that has a useafter date of Jan 26 2000 00:00:00, should not be used to calibrate data taken BEFORE that date and hour.
  Special care should be taken in deciding the useafter date because it is often used to assist with the fixing of problems in reference files used for specific date ranges. Typically, the useafter date was the last date of the date range of the data used to create the reference file. For instance, with a Pedigree of "INFLIGHT 21/01/2000 26/01/2000", the USEAFTER would be "Jan 26 2000 00:00:00". However this has not been always the case as certain reference files are "back dated" to launch in order to be used for all data. Also it should be noted that the USEAFTER date keyword has precedence over any other parameter in the reference file. This is, if there exist an older reference file with an USEAFTER date greater than that of the new reference file then the old file will be used instead. Therefore, if a new reference file should replace a previous one, the USEAFTER date of the new file should be identical to the old one (from year to second). Also, if the new reference file replaces more than one reference file, each with different USEAFTER date, one copy should be made for each of these files and the USEAFTER date should be populated to mach that of the old files.

Finally, it should be noted that in earlier deliveries the format of the USEAFTER keyword did not include the time specification. In those cases the time was implicitly 00:00:00 and this value should be added to the USEAFTER of the new reference file.

- The DESCRIP (Description) keyword should be a concise sentence used to describe the type of reference file; one that makes it easy for users to decipher the type of reference file. For example:

  ```
  DESCRIP = “Reference superdark created from proposal 7276”
  ```

  This provides a simple description of the file; more complete information will be provided in the HISTORY lines.

- In the COMMENT keyword, the names of the creators of the reference file should be entered, e.g. “Reference file was created by R. Diaz-Miller and P. Goudfrooij”.
- The NEXTEND keyword indicates the number of extensions in the reference file and It should be set to NEXTEND = 1 for tables and NEXTEND = 3 for image reference files.
- The HISTORY lines can be added using an IRAF task called, stfhistory. This task loads an ASCII file into your data header as history lines.

```
PACKAGE = headers
   TASK = stfhistory
input   = reference_file.fits[0]  Image template
text    =  @history_file.ascii  History text
(verbose= yes) Verbose message switch
(mode   =                  al)
```

The following information is *required* in the HISTORY lines:

1. A complete but concise description of the process used to create the reference file. Be specific--note what software was used and if it is available externally or only internally.
2. Include in the description any pertinent numbers that others may find useful, e.g.- how many crsplits or repeat observations were involved in each raw image, what the total exposure time for the raw images was; or, as in the case of a superdark, the value above which hot pixels were updated.
3. Where possible include relevant ISR/TIR numbers.
4. A listing of what raw datasets went into the creation of the reference file.
5. *No names of people* involved in the creation of the reference file should be included in the HISTORY lines, these names should only be mentioned in the COMMENT keyword; this is because the COMMENT keyword does not show up in StarView.

It can be useful to look at the headers of previously delivered reference files to see what was placed in their history lines. If a file has not changed appreciably since the last delivery (e.g.- if one or two errors in certain columns have been fixed), rather than repeat history lines from previously delivered tables/files, simply reference the last appropriate reference file and TIR/ISR number. Keep in mind that history lines are very important because they not only document what we did to create the reference file, but they serve to inform general observers who use these files as well.
An example history file looks something like this:

```
Created on July 26, 1999, using the
cl script "weekdark", which is available in the
stis package within STSDAS.

This superdark image is a combination of 2 images.
The first is a "baseline dark" image which is a
(typically) monthly average of (cosmic-ray-rejected)
dark files (in this case an average of 38 darks). The
second image is make locally within the "weekdark"
script from 14 darks that are taken from Proposals
7948/7949/8408/8437 "Dark and Bias Monitor".
These gain 1 darks are combined together
(cosmic rays are rejected in the combination)
using calstis, and normalized to a dark time of 1
second. After that, hot pixels in that normalized
dark are updated into the baseline dark. These hot
pixels have a value higher than
(baseline dark current + 5 sigma of the new dark).
Hot pixels have been updated above a level of
0.01723591 electrons/second for this particular dark.
The pixels hotter than 0.1 electron/sec in this dark
are being assigned a DQ value of 16.
Previously hot pixels that have fully annealed out
between the observing dates of the baseline and the
new dark are being assigned a value equal to that in
a median-filtered (kernel = 2x2 pixels) version of
the baseline dark.

The following input dark files were used:

o4wi7agxq
o4wi7bh5q
o4wi7cnyq
o4wi7doiq
o4wi7ew1q
```

If you should happen to make a mistake in your history lines and only notice it after you've added it to your primary header with **stfhistory**, you can "undo" what you did with a task called **eheader** or **tupar**. The IRAF stsdas.toolbox.header task **eheader** will allow to edit the primary image header using an interactive text editor (might not work with emacs). After finished the editing the changes are saved back to the original file. To erase lines with **tupar** simply **tupar** the file's primary header: tupar reference_file.fits[0]. At the colon prompt type "l" to list the header. Once you see what line numbers you'd like to delete type: "d #" where # is the line number you want to delete from the header. Typing e will exit and save the file; typing q will exit without saving the changes you just made.

There is also an known bug in the CDBS software which could prevent the file from being ingested into the CDBS, OPUS and DADS database. The information of the tracking file that accompanies each delivered file is extracted from the header keywords and history lines. Those are striped of extra spaces and blank lines before input in the tracking file (known as lod file). Therefore, if a history line has the word “go” at the beginning of a line, it would be mistaken by the “go” SQL command and the ingest will fail with no clear error. Check all the lines of the history to be sure that you do not have any line that starts with this word.

2.3 Rename files to contain no capital letters.

CDBS cannot process files with names containing capital letters. Such files should be renamed to contain only lower case letters. If a large number of files with names containing capital letters are being delivered, the following UNIX C script can be used to do the renaming automatically:

decap *filename*

2.4 Verify that the file is in standard fits format.

Reference files that are not in standard fits format cannot be ingested into the databases. To verify that the reference file is in standard fits format, run the UNIX **fitsverify** tool on the file:

fitsverify *filename*

Wildcards may be used for filenames, e.g., *filename* can be **.fits*.

A sample output from the tool looks like this:

```
===================================================================
FITS Verification for file:  lbq1211ao_bia.fits




===================================================================
Summary contents of FITS file: lbq1211ao_bia.fits

0:  Primary Array ( SHORT )
        0 bytes, 108 header lines, 3 FITS blocks
1:  Image Extension ( FLOAT ) [IMAGE,SCI,1] 2 dims [1024,1024]
        4194304 bytes, 36 header lines, 1458 FITS blocks
2:  Image Extension ( FLOAT ) [IMAGE,ERR,1] 2 dims [1024,1024]
        4194304 bytes, 36 header lines, 1458 FITS blocks
3:  Image Extension ( SHORT ) [IMAGE,DQ,1] 2 dims [1024,1024]
        2097152 bytes, 36 header lines, 730 FITS blocks
No special records.

===================================================================
No problems were encountered.
```

Examples of problems encountered with the files in fits verification include:

- extra spaces in keyword fields
- incorrect format for DATE keyword field (Dec. 18, 2000 instead of 18/12/00)
- missing PCOUNT and GCOUNT keywords in extension headers.

Note that this command checks for standard fits header keywords and does not report any missing generic STIS header keywords. These give the information necessary to define the observing mode to which these files apply and should be present in all reference files. A complete list of the standard fits header keywords can be found in section 9.4 of ICD-47. Those for a particular STIS reference file are given throughout Chapter 9 of the same document.

2.5 Run the CDBS certify tool on the file.

The CDBS *certify* tool performs further checking on the syntax and keyword values in the reference file, ensuring adherence to ICD-47 specifications. Note that most cdbs UNIX tasks can also be accessed through IRAF in the stlocal.cdbsutil package.

This tool is run by typing:

certify *filename*

Wildcards may be used for filenames, e.g., *filename* can be **.fits*.

More detailed documentation on the *certify* task is available in postscript format on the CDBS web page (***http://www.stsci.edu/instruments/observatory/cdbs/cdbs.html#CDBS***). The *certify* tool does not check all the keyword syntax and values in the reference file, but only those that are specifically used in CDBS, OPUS and DADS for selecting and tracking the reference files. The so-called *tpn* files specify explicitly what is checked by *certify.* These files are located in /data/cdbs1/tools/data, with names such as, e.g., stis_crr.tpn for the cosmic ray rejection reference file. These files are reviewed for accuracy by our group. If you find any problems with them, please notify the Reference File Lead and lead DA.

A sample *certify* output for a file that has a problem:

```
== Checking mama2_PFL.fits ==
Could not match keywords in header (mama2_PFL.fits
Cannot determine reference file type (mama2_PFL.fits)
```

If you encounter a problem at this stage, first check to see if there are any obvious problems with the file header keywords (e.g., required keywords are missing) or keyword values. If no obvious problem exists, you can look at the *tpn* file to see what keywords are being checked, and what the legal values for these keywords are. A sample *tpn* file for a PHT (photometry table) reference file looks like this:

```
# Template file used by certify to check reference files
# Some fields may be abbreviated to their first character:
#
# keytype = (Header|Group|Column)
# datatype = (Integer|Real|Logical|Double|Character)
# presence = (Optional|Required)
```

```
#
# NAME          KEYTYPE  DATATYPE  PRESENCE VALUES
#------------------------------------------------------------
INSTRUME           H        C        R    STIS
FILETYPE           H        C        R    "PHOTOMETRIC CONVERSION TABLE"
DETECTOR           H        C        R    CCD,NUV-MAMA,FUV-MAMA
OBSTYPE            H        C        R    IMAGING,SPECTROSCOPIC
OPT_ELEM           C       C        R    G140L,G140M,E140M,E140H,G230L,\
         G230M,E230M,E230H,PRISM,G230LB,G230MB,G430L,G430M,G750L,G750M,\
          MIRCUV,MIRFUV,MIRNUV,MIRVIS,X140H,X140M,X230H,X230M,N/A
CENWAVE             H       I         R
1173,1200,1218,1222,1234,1271,1272,\
1307,1321,1343,1371,1380,1387,1400,1416,1420,1425,1453,1470,1489,\
1518,1526,1540,1550,1562,1567,1575,1598,1616,1640,1665,1687,1713,1714,\
1763,1769,1813,1851,1854,1863,1884,1913,1933,1963,1978,1995,2013,\
2014,2063,2095,2113,2124,2125,2135,2163,2176,2213,2257,2263,2269,\
2276,2313,2338,2363,2375,2376,2413,2415,2416,2419,2463,2499,2513,\
2557,2561,2563,2579,2600,2613,2659,2663,2697,2707,2713,2739,2762,\
2794,2800,2812,2818,2828,2836,2862,2898,2912,2962,2976,2977,3012,\
3055,3115,3165,3305,3423,3680,3843,3936,4194,4300,4451,4706,4781,\
4961,5093,5216,5471,5734,6094,6252,6581,6768,7283,7751,7795,8311,\
          8561,8825,8975,9286,9336,9806,9851,10363,\
          1232,1269,1305,1341,1378,1414,1451,1487,1523,1560,1587,1760,\
          2010,2261,2511,2760,3010,1975,2703,-1,-999
USEAFTER            H        C         R    &SYBDATE
PEDIGREE            C        C         R    &PEDIGREE
DESCRIP             C        C         R
```

If this check does not reveal the source of the error, contact the Reference File Lead or lead DA.

2.6 Perform a detailed quality assessment of the data in the reference file or table.

The integrity of the calibration reference files is critical to producing high-quality science results. It is therefore important that before delivering them to the pipeline we thoroughly test and validate them to the best of our ability. In addition to checking for proper structure and syntax, checking of the data itself is essential.

Given the amount of information in the reference files, it is difficult to perform a thorough manual check on each file as it is prepared for delivery to CDBS. Therefore an IDL tool was developed to perform a variety of checks on the files. The "Reference File Checking Tool" (RFC tool) allows a robust and consistent testing of the reference files and makes certain that the files submitted to CDBS are both syntactically and scientifically valid. This tool can be used to check most of the existing reference files, at least for those that are delivered more frequently. The details on how to use it and the particular checks performed for each reference file can be found in ***TIR 2000-01B***, here we will describe briefly the main test performed by the tool and procedures to follow. Any problems, corrections or addendums should be addressed to the Reference Files Lead and lead DA.

The RFC tool checks are divided in three parts:

1. Check the primary header (extension [0]) of the reference file. It verifies that the minimal header keywords that describe the reference file are present. For some of them it also checks for valid unique or possible entries. It checks the format of the

USEAFTER and PEDIGREE keywords and a compare of the old and new USEAFTER values (if possible) is performed. The output looks like this

```
Primecheck status

The keyword OBSTYPE has a bad value of ANY

******************************************************
SUMMARY PRIME HEADER ERRORS

The total number of missing keywords is:        0
The total number of invalid keywords is:        1


USEAFTER has changed: old - NOV 12 2001 00:00:00   new - NOV 20 2001
00:00:00

******************************************************
```

Any missing keyword should be added and/or keyword values corrected before delivery.

2. Check data formats in the reference file. The tool compares the format, depending on the type of file, to those in a template file. Each parameter is then compared to verify that the field is of the correct size. An error output looks like this:

```
Formatcheck status

Parameter OPT_ELEM has the wrong format.
6A instead of 8A
Parameter CENWAVE has the wrong format.
I instead of 1I
Parameter SPORDER has the wrong format.
I instead of 1I
Parameter REF_APER has the wrong format.
7A instead of 12A
Parameter A2CENTER has the wrong format.
E instead of 1E
Parameter NCOEFF has the wrong format.
I instead of 1I
Parameter PEDIGREE has the wrong format.
30A instead of 67A
Parameter DESCRIP has the wrong format.
28A instead of 67A
****************************************************************
SUMMARY FORMATTING ERRORS:

The total number of bad formats is:        8
The total number of parameters not in the template is:        0
The total number of missing parameters is:        0
****************************************************************
```

Note that in this case the error flags can be ignored since the difference is only in size and not parameter type.

3. Check the data. This varies depending on the file type (table or image) and on the particular reference file (see ***TIR 2000-01B***). In general for table reference files it checks that all the detector modes are present and that these have values within a valid range. For example, the message from the DSP reference file:

```
“The cenwave  8975 in row  958 is not defined for the opt_elem G750L”
```

indicates that the given cenwave is not defined in the template files as a supported mode. If this is a new mode that should be added to the template files please notify the Reference File Lead and lead DA. There are cases where the prompted errors will have no effect on the calibration of data, cannot be corrected easily and can be neglected. In this case, the output of the RFC tool should be saved into a file and delivered with the corresponding reference file. Before delivery edit the file and indicate the reasons for which the flag(s) can be disregarded. The tool also compares the new data with the data of old reference files and prompts any changes. Verify that all the changes are flagged correctly and that these reflect the changes made to the reference file. Again, justify them in the in the output of the RFC tool and include this file in your delivery. In the case of images, it obtains the statistics of the SCI, ERR and DQ extensions and in some cases (like for DARK, and BIAS reference files), it also checks for hot pixels and pixels with large errors. The output for a BIA reference file looks like this:

```
 FILENAME= 11x1_2001_1120_1125_ref_bia.fits Statistics:

 Extension type:            SCI          ERR           DQ
 Mean:                  0.7694486    0.4760699    0.0377655
 Median:                0.7160096    0.4787739    0.0000000
 Standard Deviation:    2.6389852    0.0126393    1.3749301
 min:                 -19.6229706    0.4507987    0.0000000
 max:                2119.0820312    5.2712369  256.0000000

 The total number of hot pixels in the file is =         2123
 The number of pixels (out of 1024x1024) with large errors =  169373
(16.2%)

 Checking Data Quality extension

 *************************************************************
 Summary errors:
 There is/are       0 wrong header keywords
 There is/are       0 illegal data quality values
 *************************************************************
```

For the Bias and Dark reference files special care should be taken examining the statistics of the DQ extension. In particular the min and max value should not have decimal values after the floating point and the max value should be a valid DQ value. The tool can compare the new reference file with an old one, but the comparison is limited to obtaining statistics for the old reference file.

Any formatting problem, missing keyword or data error will be reported and should be corrected before delivery. It is recommended to include in the delivery the output of the RFC tool (check ***TIR 2000-01B*** for how to send the output to a file) with an explanation of the remaining “error” flags. Note that there are a few reference file types (LIN, MOC, SDC, WCP and scattered correction files) for which the tool cannot be used. These are , rarely updated however, and there is no need for a tool of this kind. In order to test these files other checks can be used, as mentioned below. The RFC tool has been designed to perform a considerable number of tests, but it is likely that some important checks were omitted due to the complexity involved. If considered necessary these should be performed independently. In the case in which further tests are necessary, new tests can be implemented in the tool; to discuss the implementation of a particular test in the tool contact the Reference File Lead and lead DA.

For those cases were automatic checking has not been implemented, there are many IRAF tasks that can be used to examine files. For headers: ***eheader, imhead, catfits, hedit, tlcol***; for data: ***listpix, imstat, tread, tstat, display***. As with the RFC tool, consistency checking should be employed. Do the new files look like the old ones? If not, why? Keep careful notes on how you determined that the new reference file was good. To further assist in assessment of this files, the IRAF tasks to create reference file paper products can be used There are two reference file paper products tasks, one for images and one for tables. For images, the IRAF task is ***pp_images***, located in the xstis.reffiles. An lpar of the task looks like this:

```
infile = "" Image or list of images to analyze
deliv_date = "" Date of delivery (mm/dd/yy)
(inimglist = "")
(mode = "ql")
```

To run the task type at the IRAF prompt:

pp_images *filename date*

where *date* is the date the file will be delivered, or the current date (this date is used only to help organize the printed paper products). Similarly for tables:

pp_tables *filename date*

The paper products are be used to perform a quick sanity check on the data, and should also be used to check that keywords are set appropriately. Types of problems that have been found in the past include:

- DETECTOR=MAMA-FUV instead of FUV-MAMA
- USEAFTER keyword missing HR:MIN:SEC parameter
- illegal optical element and cenwave combinations
- missing keywords (NEXTEND, COMMENT)
- size of image does not match keyword values

Where appropriate or useful the paper products postscript files should be provided along with the reference files when the files are ready for delivery.

2.7 Test the reference files in calstis

Calstis must be used to test calibration of actual data with the reference file(s) being readied for delivery. The highest level check should be a verification that the appropriate stages of *calstis* do not crash, and that the calibrated data makes sense. Make sure that the proper *calstis* switches are set to PERFORM so the file being tested is actually used. The data used to create the reference file should be calibrated with the new reference file, and the results should make sense. Real datasets should be calibrated with both the new reference file, and the old reference file, and the differences understood and thoroughly documented (see step 2.2 for information on how to document the files). It is good to run the IRAF task, **imstat** on the file and determine if the values are reasonable on all extensions: science, error and data quality. Often plotting or displaying the file will reveal obvious problems that might otherwise have been missed.

2.8 Fill out and e-mail the delivery template.

When all of the above steps have been completed with no errors, the reference files are ready for delivery. Fill the delivery template (shown below) and send it to the Pipeline Reference File lead and lead DA. Make sure that the files and directory where the files are located are group accessible. This directory should also contain any other file(s), like the output of the RFC tool or paper products, that help with the validation of the reference files. The delivery template can be found in the STIS Pipeline web page under Calibration Reference Files (***http://www.stsci.edu/hst/stis/projects/Pipeline/delivery.txt***) and contains the following information:

```
Name:
Date:
Type of file (bias,pht,etc.):
History section complete? (yes/no):
Verification complete? (fitsverify,certify,etc.):
Files run through calstis?
What is the level of change compared with the old file? (e.g. 1%,5%
etc.):
Description of how the files were "tested" for correctness:
Disk location and name of files:
```

This concludes the reference file creator/tester work.

*2.9 Copy the files to a delivery directory.-- *Pipeline group's responsibility starts here**

Place the reference files and any accompanying files in a delivery directory. Currently these files are in the directory */data/tauri7/reffiles/delivered/ (*deliveries from 1997 to 10/22/2002 can be found in */data/alberich7/reffiles/delivered/)* under a directory named for the date when the files will be delivered. The format of this directory is yyyy_mm_dd. This files will remain here for safekeeping until backups can be made. Complete steps

2.10 to 2.16 in this directory and, to save disk space, gzip the files after the files have been ingested in all the databases.

It is also desirable to keep a log file with the output of all the tests performed on the files. In this case, redirect the output of each test to a log file using the append redirection “>>&” command. Note the “&” character, this should be used as some of the CDBS scripts send the output flags to standard error, instead of standard output handled by the append redirection “>>” command.

2.10 Verify that the file is in standard FITS format

See section 2.4 for details.

2.11 Run the CDBS certify tool on the files.

See section 2.5 for details.

2.12 Create the CDBS load file

In order to correctly identify the files in CDBS, an ascii 'load' file is created for each reference file that contains information from the reference file header, and information from the database about existing reference files. The load file information is stored in CDBS for tracking, and in DADS for use in reference file access. The load file will have the same root name as the reference file, but with the extension '.lod'. The file consists of at least two sections: the header section, and the row section. For *image* reference files, there is a header section followed by one row section. For *table* reference files there is a header section followed by one or more row sections, each representing a row or group of rows in the reference table.

To create the load file type the following UNIX/IRAF command:

mkload *filename*

Wildcards may be used for filenames, e.g., *filename* can be **.fits*. More detailed documentation on the *mkload* task is available in postscript format on the CDBS web page under CDBS Operations (***http://www.stsci.edu/instruments/observatory/cdbs/cdbs.html#CDBS***).

An example of a load file for a reference file image:

```
FILE_NAME = 11x1_2001_1120_1125_ref_bia.fits
INSTRUMENT = stis
REFERENCE_FILE_TYPE = bia
USEAFTER_DATE = Nov 20 2001 00:00:00
COMPARISON_FILE = lbq12111o_bia.fits
OPUS_FLAG =
COMMENT =
ENDHEADER

CHANGE_LEVEL =
PEDIGREE = INFLIGHT
OBSERVATION_BEGIN_DATE = Nov 20 2001
OBSERVATION_END_DATE = Nov 25 2001
```

```
BINAXIS1 = 1
BINAXIS2 = 1
CCDAMP = D
CCDGAIN = 1
CCDOFFST = 3
DETECTOR = CCD
COMMENT =
ENDROW

ENDFILE
```

2.13 Populate keyword fields in the load file appropriately.

Many of the load file keyword fields are automatically populated from the reference file primary and extension headers by **mkload**. There are six keyword fields that we *must populate* in the load file. Those are: OPUS_FLAG and COMMENT in the header section, and CHANGE_LEVEL, PEDIGREE, OBSERVATION_BEGIN_DATE, and OBSERVATION_END_DATE fields in the row section. The COMMENT field in the row section can be blank if there are no relevant comments at the row level, but use of comments at the row level is strongly encouraged. We will be delivering reference file tables where only a few rows of the table have changed significantly from the last file delivery. In such cases, row-level comments may be more appropriate than table-level comments, and *they are required under such circumstances*. An IRAF task, **setlodkeywd**, has been developed for populating lod files and will be explained in detail later in this section.

header section of load file:

- OPUS_FLAG

Set this to Y or N to indicate whether the files are intended for pipeline use or not. We do expect to deliver reference files that will not be used in the pipeline, for example SYNPHOT files, PSFs, and the 2” flat. The OPUS_FLAG should be set to N for such cases and the files will not be delivered to the OPUS database.

- COMMENT

The COMMENT section of the load file is the equivalent of the HISTORY lines in the data header. It is recommended to fill this section.

***row section of load file**:*

- CHANGE_LEVEL

This keyword defines the level of change of the reference file with respect to the last delivered file (defined by the file identified in the field of the keyword COMPARISON _FILE). This field must be set to one of three values: SEVERE, MODERATE, or TRIVIAL. The criterion for each are:

SEVERE

- *initial delivery* of any file
- Change that *requires existing data to be recalibrated*
- The row-level field for a table has changed by more than 50% compared to the COMPARISON_FILE

MODERATE

- changes are significant, but do not warrant data recalibration
- the row-level field for a table has changed by ~10-50% compared to the COMPARISON_FILE

TRIVIAL

- changes are insignificant (e.g., fixing typos; removing erroneous but unused rows from a table), and do not warrant data recalibration

- PEDIGREE

This should be GROUND, DUMMY or INFLIGHT.

- OBSERVATION_BEGIN_DATE and OBSERVATION_END_DATE

These are the actual start and end date for the acquisition of the calibration data used to create the reference file. The format should be Month Day Year (e.g., April 12 2001) to be consistent with the USEAFTER date format.

An IRAF task, called ***setlodkeywd*** and located in xstis.reffiles, has been developed for use in populating keyword fields in the load files automatically. This is particularly useful if a large number of files need to have fields populated in an identical manner. An lpar of the task looks like this:

```
infile = "            "File or list of lod files to fix
comments = yes         Add comments? (yes/no/append)
change_level = "      "Change level value: SEVERE, MODERATE, TRIVIAL
opus_flag = "         "Opus flag (Y/N)
pedigree ="           " Pedigree entry: GROUND, DUMMY, IN-FLIGHT
(inlist = "           ")
(inlod = "            ")
(mode = "            q")
```

`infile` should have the load file name or list of lod files to be edited. If a list of files is used, use the '@' symbol. DO NOT use '*.lod'.
`comments = yes` will copy all DESCRIP and HISTORY lines from the reference file header, deleting what is currently present in this entry. If `comments = no`, nothing will be copied from the reference file and information present in this field will not be changed.
`change_level` is the change level value. Refer to explanation above for the appropriate value to use. If it is left blank, the current entry will remain unchanged.

`opus_flag = ""` is the opus flag value. Refer to explanation above. If it is left blank, the current entry will remain unchanged.

`pedigree = ""` is the pedigree value. Refer to above explanation. If it is left blank, the current keywordvalue will be retained.

`inlist, inlod` is a list parameters used internally by the task. Do not enter any value here.

An example of a filled load file looks like this:

```
FILE_NAME = 11x1_2001_1120_1125_ref_bia.fits
INSTRUMENT = stis
REFERENCE_FILE_TYPE = bia
USEAFTER_DATE = Nov 20 2001 00:00:00
COMPARISON_FILE = lbq12111o_bia.fits
OPUS_FLAG = Y
COMMENT = Superbias created by R. Diaz-Miller
Created on Dec 19, 2001 using the cl scripts
"refbias" and "refaver", which are available
in the (local) xstis package within STSDAS.
Superbias image, combination of 98 input bias frames
taken in CCDGAIN=1, BINAXIS1=1, BINAXIS2=1 mode.
All input frames were from Proposal(s):
8901/8903 "CCD Bias Monitor".
The following input files were used:
o6hn2b010
o6hn2c010
o6hn2d010
o6hn2e010
o6hn2f010
o6hn2g010
cl script "refbias" was run on these input files,
after having split them up into sub-lists of less
than 30 imsets each. After running "refbias" on the
individual sub-lists, script "refaver" was run to
average the reference files resulting from the
individual "refbias" runs together.
The "refbias" procedure works as follows. After
joining the files together into a multi-imset file,
overscan subtraction is performed for every individual
bias frame. The bias frames in the multi-imset file
are then combined using ocrreject, which performs a
three-step iterative cosmic-ray rejection with
rejection thresholds of 5, 4, and 3 sigma,
respectively.
After cosmic ray rejection, the combined bias was
divided by the number of frames combined (ocrreject
adds images instead of averaging them), using the
stsdas.toolbox.imgtools.mstools.msarith routine.
Subsequently, hot pixels (and columns) were identified
as follows. A median-filtered version of the averaged
bias is created (kernel = 15 x 3 pixels) and
subtracted from the averaged bias, leaving a
"residual" image containing hot pixels and columns.
The pixels hotter than 5 sigma of the effective RMS
noise in the residual bias image are then identified,
and flagged as such in the Data Quality (DQ)
extension of the output reference bias file.
ENDHEADER

CHANGE_LEVEL = SEVERE
```

```
PEDIGREE = INFLIGHT
OBSERVATION_BEGIN_DATE = Nov 20 2001
OBSERVATION_END_DATE = Nov 25 2001
BINAXIS1 = 1
BINAXIS2 = 1
CCDAMP = D
CCDGAIN = 1
CCDOFFST = 3
DETECTOR = CCD
COMMENT =
ENDROW

ENDFILE
```

Note that in some cases a new reference table may be identical to it's predecessor with the exception of some rows within the table. In these cases, use the CHANGE_LEVEL from the header section and se to TRIVIAL the unchanged row groups. Currently, the task ***set-lodkeywd*** can only change all of the lines in a lod file to the same value. Should the file require multiple values, the lod file will need to be edited "by hand" with a favorite editor. An example of such a situation would be a new PHT (photometry) table where the G230LB mode was updated while the G230MB mode was unchanged. In this case the row section of the lod file might look like the following:

```
CHANGE_LEVEL = SEVERE
PEDIGREE = INFLIGHT
OBSERVATION_BEGIN_DATE = May 21 1997
OBSERVATION_END_DATE = Jul 1 1997
CENWAVE =    -1
DETECTOR = CCD
OBSTYPE = SPECTROSCOPIC
OPT_ELEM = G230LB
COMMENT =
ENDROW

CHANGE_LEVEL = TRIVIAL
PEDIGREE = GROUND
OBSERVATION_BEGIN_DATE =
OBSERVATION_END_DATE =
CENWAVE =    -1
DETECTOR = CCD
OBSTYPE = SPECTROSCOPIC
OPT_ELEM = G230MB
COMMENT =
ENDROW
```

2.12 Certify the load files.

After creation of the lod files they also need to be certified:

certify *filename.lod*

where the filename can be replaced for a wildcard (*.lod). If the reference file, and consequently the load file, uses wildcard values (-1, -999, ANY, or N/A) for any of the keywords (see ICD-47), *certify* will report the following "error" and the load file needs to be "expanded":

```
Error in opt_elem[1]: "any" is not a legal value. May need to run expload
Error in cenwave[1]: "-1" is not a legal value. May need to run expload
```

To do this run the CDBS task **expload** (note that this task does not take wildcards);

expload *filename_in.lod filename_out.lod* ***stis.rule***

where the “filename_in.lod” file can be the same as “filename_out.lod”. Expload expands the load file to deal with the situation where a single reference file is to be used for many modes. For example, suppose we have a reference file that is applicable for ANY central wavelength (CENWAVE). Expload will “expand” the load file to cover all legal CENWAVE values, providing one row in the .lod file for each value. The expansion of the files is governed by the so-called *stis.rule* file, which is located in the directory /data/cdbs1/tools/data/. This file shows the current legal values used for wildcarded keyword fields in the expansion. For example, the rule for expanding the combination of OBSTYPE=SPECTROSCOPIC and OPT_ELEM=ANY the following rule (taken from the stis.rule file) applies:

```
OBSTYPE = SPECTROSCOPIC && OPT_ELEM = ANY =>
OPT_ELEM=G140L || OPT_ELEM=G140M || OPT_ELEM=E140M ||
OPT_ELEM=E140H || OPT_ELEM=G230L || OPT_ELEM=G230M ||
OPT_ELEM=E230M || OPT_ELEM=E230H || OPT_ELEM=PRISM ||
OPT_ELEM=G230LB || OPT_ELEM=G230MB || OPT_ELEM=G430L ||
OPT_ELEM=G430M || OPT_ELEM=G750L || OPT_ELEM=G750M ||
OPT_ELEM=X140H ||OPT_ELEM=X140M || OPT_ELEM=X230H ||
OPT_ELEM=X230M;
```

Once the file has been properly exploaded, run *certify* again until there are no errors or expansions required. Repeat this step as necessary until *certify* does not report any missing keyword information.

2.15 Run the check load task on the lod files.

The CDBS task, **check_load**, must be run on the lod files before they can be delivered. This task takes wildcards.

check_load **.lod*

The output from this task will look like the following:

```
starting check_load
database  cdbs_ops
server  CATLOG
Thu Feb 26 11:25:26 EST 1998
load file: i2916173o_drk.lod
header file: i2916173o_drk.fits
Wrote file (i2916173o_drk.lod)
no differences for file i2916173o_drk.lod
```

2.16 Run uniqname on the files and deliver the reference files to the OPUS, DADS and CDBS databases.

The reference files are ready to be ingested in the databases but before they can be handed off a unique time-stamp name should be assigned to the reference files and their associated load files. This can be done running the UNIX command called uniqname:

uniqname **.lod >>& cdbsnames*

The files will be renamed to CDBS style reference file names and send the output of the task to an ascii file called “cdbsnames”. This file will keep a record of how the original names relate to the new CDBS names.

Deliver the reference files to the OPUS, DADS and CDBS databases. The procedures to ingest reference files in these databases will be described in a future STIS TIR.

2.17 Verify the correct usage of the reference files in the operational environment.

Acknowledge that the reference files have been loaded into CDBS, OPUS and DADS databases should be received. This is done via a message from the OPUS team with the ingest date and an e-mail from Stephen Dignan informing that the files have been used in the latest bestref run. All the delivered files should be listed in these messages. To test that all the files have been ingested properly, go to StarView and open the search form for “STIS Reference Files”. Check that the new reference files are recommended as the best reference files (ssr_best columns) within the correct date range and that OPUS pipeline has used the files (if applicable) after the date they were ingested. An exhaustive checking of all the affected datasets is recommended. There have been a few cases were the files are not ingested properly and the old reference file is still being recommended for some datasets. If the new reference files replace previously delivered files, you could check it in StarView by setting the best reference file qualifier name to that of the old reference file; in this case no records should be found. To further check that the OPUS database is using the new reference correctly (i.e. observations made before the file was delivered), you could retrieve data from the archive and verify that the new reference files are being populated in the header correctly. This is however not necessary as we have had no problems with this database in recent years.

2.18 Place a copy of the Reference file in the OREF directory and update the history web page.

Once that the correct usage of the files has been verified the files should be copied to the OREF directory (data/garnet3/oref/), making sure that the files are group and other readable. Finally, update the external Reference Files History web pages. These are located in the internal STIS web page under Pipeline /Reference Files History (***http://www.stsci.edu/hst/stis/calibration/reference_files/oref.html***). Follow the format particular to each reference file using the information from the file. In these pages, the COMMENT column should not necessarily be the COMMENT keyword in the reference file. Rather it should contain a very simplified description of the changes/uses of the reference files. It should not contain any names and should be concise as it intends to give the user a general idea of the possible impact of these files on his/her data. Notify the creator and the Reference File Pipeline Lead that the files have gone through the system correctly.

Conclusions and Recommendations

Any problems encountered with the above procedure should be directed to the Pipeline Reference File lead and lead DA (see Pipeline web page). Suggestions for streamlining the procedure as we gain experience are encouraged.