Technical Instrument Report CDBS 2008-01

Assessment and Delivery of Reference Files

R. I. Diaz, M. Cracraft
rmiller@stsci.edu, cracraft@stsci.edu

June 24, 2008

Abstract

This TIR defines the standard procedures for the delivery of calibration pipeline reference and SYNPHOT data files. It clearly establishes the instrument and OTA teams' responsibilities and provides guidelines for testing and verification of reference files. This is a revision to TIR 2005-01A. In this revision, we update information to account for changes in the testing procedures for SYNPHOT files.

Introduction

The generation of calibration reference files and SYNPHOT data files is currently the responsibility of the instrument and OTA teams at STScI. As part of this process, standard testing and assessment of the files need to be performed before they are "delivered" officially to the INS/CDBS (INS Calibration Data Base System) Team for their ingest into the CDBS, DADS, and OPUS databases. In order for the reference files to be used by the OTFR pipeline, those files have to be ingested in the OPUS database, while their ingestion into the Data Archive Distribution Services (DADS) database allows users to retrieve them from the archive and mirror sites. The main function of the CDBS database is to allow the selection of the correct reference files for a specific set of observations. The selection is based on data file keyword values outlined in ICD-47, located at http://www.stsci.edu/instruments/observatory/cdbs/documents/icd-47_RevF.pdf or http://www.ess.stsci.edu/projects/distribution/ICD47/ICD-47_RevF.pdf.

There is another set of Synphot files that are not ingested into the CDBS database, but they are considered part of the CDBS system because they are used by SYNPHOT. These files are the stellar atlases and libraries and HST calibration spectra. Although these files do not go to the database, they should also be validated and tested.

The checking and quality assessment of reference files will be performed by the person creating the files, while the official delivery of the files to the databases is handled by the INS/CDBS Team. If atlases and libraries or HST calibration spectra are being delivered, the INS/CDBS group will

work with the instrument teams to assess their accuracy. Checking and quality assessment of reference files is required for the following reasons:

- to ensure correct keyword content and syntax
- to ensure correct file structure and format
- to ensure that the files will work properly with the pipeline calibration software
- to ensure that the files are correctly selected by SYNPHOT
- to verify scientific validity
- to ensure proper documentation of the reference files

Accordingly, we define in this document the standard procedures for checking and assessing the quality of the reference files. The procedures outlined here should follow the creation of the reference files, which are assumed to be in ICD-47 format. We first provide an outline of the steps to be performed, followed by a more detailed description. If errors are encountered while testing the reference files or SYNPHOT data files, they must be resolved before proceeding to the next step. Unresolved problems with the CDBS tasks should be referred to the INS/CDBS Team (cdbs@stsci.edu). The detailed steps and procedures for ingesting reference files into the databases are described in TIR CDBS 2008-02.

Summary: Preparation and Assessment of Reference Files

1. Update all the reference files' headers properly (including header keywords and history lines).
2. In the case of pipeline reference tables, update the Comment and Pedigree columns
3. Rename files so they contain no capital letters and in the case of SYNPHOT, have no version number.
4. Run fitsverify on the FITS files.
5. Run the CDBS `certify` tool on the standard FITS files or the GEIS header files.
6. Perform a detailed quality assessment of the data in the reference files.
7. In the case of table pipeline reference files, make sure that all the necessary rows are present.
8. Test the files using CALXXX or SYNPHOT, using the compare_table IDL procedure, and if possible with the ETCs.
9. Fill out the delivery template and notify the INS/CDBS Team that the files are ready for delivery.

Detailed description of the preparation steps

1. Update all the reference files' headers properly (including header keywords and history lines)

Once the reference files have been created, it is necessary to update the primary headers with the proper values. The relevant header keywords are:
`USEAFTER` date,
`PEDIGREE` of the file,
`DESCRIP`tion,
`COMMENT`, and
`HISTORY` lines.

Header keywords that are instrument specific, like `DETECTOR` or `FILENAME`, should be filled in compliance with the formating outlined in the ICD-47 document.) These header keywords are crucial for the correct selection and documentation of the reference files and can be added using the IRAF task `hedit`. For SYNPHOT data files (also known as throughput tables) three extra keywords have to be present:
`INSTRUME`,
`COMPNAME`, and
`DBTABLE`.

The `USEAFTER` and `PEDIGREE` (and `INSTRUME` and `DBTABLE` for SYNPHOT data files) keywords will most likely already exist in the header; `DESCRIP` and `COMMENT` (and `COMPNAME` for SYNPHOT data files) may need to be "added". This can be done by setting the add parameter equal to yes in `hedit`. Note that the information contained in the `USEAFTER` date keyword is used differently by the pipeline and SYNPHOT software. In order to avoid confusion, each of these cases is discussed independently. Also, if the reference file's primary header has information that is not relevant to the reference file itself (e.g., calibration keywords, observation parameters, calibration switches, etc., of one of the datasets used to create the file), erase them before delivering the file. Make the primary header short, concise, and clear.

1.1 USEAFTER (in Pipeline Reference Files)

This is the most crucial of all the header keywords in a reference file that will be used by the calibration pipeline. It indicates the date and time after which this file should be used in the calibration of science data. The format for this keyword is "`Month dd yyyy hh:mm:ss`", where the time entry is optional. For the month, any valid abbreviation is acceptable (or write out the whole word). The time should be set equal to 00:00:00 for most of the reference files, which is the same as not using a time entry. Exceptions to this case are for those files, like STIS or ACS bias and dark reference files, that should not be used across anneal occurrences and therefore the time is set equal to the anneal time. For example, a reference file that has a `USEAFTER` date of

"Jan 26 2005 05:23:53", should not be used to calibrate data taken BEFORE that date and hour. Furthermore, although most reference files are "back dated" to launch in order to be used for all data, there are cases where more than one file of the same type and for the same mode exist, but with different USEAFTER dates. Special care should be taken in deciding the USEAFTER date for those cases; to make sure that they will correctly supersede outdated reference files or cover the specific date ranges. As a rule of thumb, for those new reference files that will replace an old reference file, the exact USEAFTER date and time of the old reference file must be used for the new reference file.

1.2 USEAFTER (for SYNPHOT Data Files)

The USEAFTER keyword in SYNPHOT data files (also known as throughput tables) is used in a different way than the USEAFTER date keyword in the pipeline reference files. The SYNPHOT software itself does not care about this keyword, but it is used by the CDBS database to determine when the files were created. In this case, the USEAFTER date information is used by one of the CDBS tools, the MKCOMPTAB task, that creates the Master Component Table (MC table), or "lookup table". This task is used by the INS/CDBS Team in the delivery process to recreate the MC table. (Note that for the purpose of testing, the MC table should be edited manually; for example, with the task TEDIT.) In this case, for a particular SYNPHOT data file type, the MKCOMPTAB task looks into the CDBS database and identifies the files with the most recent USEAFTER dates. The most recent files are then used to populate the MC table for each of the SYNPHOT components. The format for this keyword is "Month dd yyyy [hh:mm:ss]", where the time entry is optional.

Note that if the CDBS tool MKCOMPTAB finds several versions of a data file of the same type with the same USEAFTER date, it will list all of them in the MC table and therefore there will be repeated entries for the same SYNPHOT component. In this case, the order in which those files appear is important and SYNPHOT will use only the last of those data files listed in the MC table. Pysynphot, on the other hand, will use the first file listed, so duplicated entries should be fixed before we migrate to the new software.

1.3 PEDIGREE

The header keyword PEDIGREE describes the type of data used to create the reference file:
INFLIGHT (needs dates),
GROUND or
DUMMY.
If PEDIGREE = INFLIGHT, the whole range of dates of the data used to create the file should be listed. For example, "INFLIGHT 21/01/2000 26/01/2000"; where the dates have format *dd/mm/yyyy* (note the four digit year). If the data in a reference file have more than one kind of pedigree (e.g., there are rows with pedigree INFLIGHT and GROUND), set the header keyword PEDIGREE to the one occurring more in the reference file, as only one kind of pedigree can be used in the header. Also, when working with tables, make sure to update the pedigree within the table;

i.e. the column pedigree of table reference files for those rows that were changed/updated. More on this in next section.

1.4 DESCRIP

The DESCRIP (description) keyword should be a concise sentence; one that makes it easy for users to decipher the type of reference file or the changes being made. This keyword should be a one-line explanation of why the file is being delivered. For example: `DESCRIP = "Reference superdark created from proposal 7276"`, provides a simple description of the file. Other examples are:

```
DESCRIP = "In-flight low-order flat based on observations of GD 71"
DESCRIP = "Revised sensitivities for first order M modes"
or
DESCRIP = "New blaze function corrections and sensitivities for eschelle modes".
```

Note that the last two examples summarize changes made to the files. More complete information will be provided in the HISTORY lines.

1.5 COMMENT

In the COMMENT keyword, the names of the creators of the reference file should be entered; e.g., COMMENT = "Reference file was created by R. Diaz-Miller and P. Goudfrooij."

Note, however, that some FITS reference files have a default COMMENT section that refers to the FITS file format and which cannot be modified or erased. The FITS default COMMENT section is different than the COMMENT section (header keyword) referred to here. The way to distinguish between these two is by their format. In the case of the CDBS required COMMENT line, the word COMMENT is followed by an "=", as in the example above and should list the people who created the file. If this does not exist in the current FITS file, it can be added in one of two ways.

Using IRAF, you can first delete the default COMMENT lines that appear in the file and then add the new one. The commands needed to do this are:

```
cl> thedit file.fits[0] comment delete+
cl> hedit file.fits[0] comment "= 'comment string'" add+
```

Note that the "=" should be added at the beginning, or a comment section line would be added rather than the header keyword you were trying to create. The other way to add the header keyword is by using Pyraf as follows.

In Pyraf:

```
import pyfits
hdulist=pyfits.open(myfile.fits,mode=update)
hdulist[0].header.add_comment(= comment string,before=origin)
```

```
hdulist.flush()
```

This will add a header keyword `COMMENT` leaving intact the FITS comment section already in the file.

1.6 DBTABLE (for SYNPHOT data files only)

The `DBTABLE` keyword should be set to "CRTHROUGHPUT" for all SYNPHOT throughput files.

1.7 COMPNAME (for SYNPHOT data files only)

The `COMPNAME` keyword should have the SYNPHOT name of the component to which the delivered reference file applies. That is, the `COMPNAME` keyword in the HEADER of the file has to match the TMG and TMC `COMPONENT` name. For example, if the TMG and TMC files have COMPNAME = ir_f126n, the COMPNAME header keyword of the file to be used should be "ir_f126n". Check previously delivered files to make sure that you are using the right value. For some files, this is the first part of the name of the file you are delivering. For example, the data file "stis_a2d8_new_syn.fits", has `COMPNAME stis_a2d8`. The important thing is that there is agreement between the `COMPNAME` header keyword and the `COMPNAME` columns.

1.8 INSTRUME (for SYNPHOT data files only)

The `INSTRUME` (instrument) keyword should have the instrument to which the SYNPHOT data files apply. The instrument name should be in lowercase.

1.9 HISTORY

The `HISTORY` lines can be added to the FITS file header in many different ways, and only some of them will be mentioned here. In the case of WFPC2 GEIS files, the header is an ASCII file and it can be edited using any editor.

In the case of FITS files, one way to add the history section is via the IRAF task called `stfhistory` in the `stsdas.toolbox.headers package`. This task loads an ASCII file, given in the parameter "text," into your reference file header as history lines, pre-appending the word `HISTORY` to each line. For example:

```
PACKAGE = headers
TASK = stfhistory

input = reference_file.fits[0] Image template
```

```
text = @historyFile.ascii     History text
(verbose= yes)                Verbose message switch
(mode = al)
```

will input the text contained in the file ‘‘historyFile.ascii’’ as HISTORY lines in the header [0] of the reference file reference_file.fits. Make sure that in this ASCII file each line has 62 characters or less, otherwise the stfhistory task will break the text as needed and you will end up with a HISTORY section that has rows of different length.

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 to the GOs 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 to create the image, the total exposure time, the kind of software used to create the reference file, or the value above which hot pixels were updated.
3. When 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 be mentioned in the COMMENT keyword. This is because the COMMENT keyword does not show up in StarView.

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 the file as well. In some cases it can be useful to look at the headers of previously delivered reference files to see what was placed in their history lines. However, in those cases where a file has changed appreciably since the last delivery (i.e., if most of the rows have changed), rather than repeat history lines and produce a lengthly output that is hard to follow, simply reference the last appropriate reference file and TIR/ISR number in the history section of the new file. 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 made 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 have added it to your primary header with `stfhistory`, you can "undo" what you did with a task called `eheader`. The IRAF `stsdas.toolbox.eheader` task will allow you to edit the primary image header using an interactive text editor. (It might work only with "vi".) After editing is complete, the changes are saved back to the original file. For example, to erase lines in "vi" go to the line you want to erase using the arrow keys and then type "*dd*". To erase a word, type "*dw*"; to erase one character, type "*x*". To insert text, type "*i*" and then start inserting your text; to escape insertion mode, use the $< ESC >$ key. To save your changes, type "$< ESC >: wq < return >$". This last command will take you out of the "vi" editor. Notice that the `eheader` task also can be used to add history lines. In this case you should remember to add the word "`HISTORY  `" (leave four spaces after this word) at the beginning of the line.

Special considerations for WFPC2 GEIS files

In the case of GEIS files (e.g., WFPC2 reference files), make sure that the headers are FITS compliant, even if those are ASCII files only. For this, replace tabs with spaces and double quotations with single quotations; this is because double quotations and tabs are not FITS compliant. In the case of single quotations, make sure that enough empty spaces are left between the single quotes by matching the format of the rest of the rows in the header file. Following these guidelines will make sure that all the header keywords are FITS compliant when the files are

converted to "waiver" FITS files.

2. In the case of tables, update the Comment and Pedigree columns

For most of the pipelines, information regarding the reference files used for the calibration of the data is provided in the trailer files. The usual information provided in the trailer files are the name of the reference file and information on their pedigree and comment or description. In the case of image reference files, and some of the table reference files, the comment and pedigree that appear in the trailer file are extracted from the header keywords of the file. In the case of table reference files that have a "pedigree" and/or "comment" column, the pedigree and comment section of the trailer file are extracted from the particular row used. Therefore, it is recommended to pay particular attention to the content of these two columns when updating these table reference files. It is important to make sure that the information in these columns is accurate (in the case of "pedigree") and provides a clear description of the data contained in the row and that is relevant for the user to know when reviewing the trailer files.

3. Rename files so they contain no capital letters or version number.

CDBS cannot process files with names containing capital letters. Such files should be renamed to contain only lower case letters. Also, in the case of SYNPHOT data files, rename the files to contain no version numbers; we suggest replacing the version number by the word "new". For example, replace the file name "stis_g750l_007_syn.fits" with "stis_g750l_new_syn.fits". In the end, this will facilitate the handling of the SYNPHOT data files by the INS/CDBS Team.

Mac users should be working in the centralized storage area in order to use the CDBS test scripts. In order to correctly define the paths and use the scripts, each user should follow the directions on the 'Scripts and Environment Setup for Mac Users' page created by Paul Goodfrooij. (`http://www.stsci.edu/wiki/CSSC/ScriptsAndEnvironmentSetupForMacUsers`)

4. Verify that the files are in standard FITS format (not applicable to GEIS files)

In order to use the CDBS tools described in the following sections in the UNIX shell line, the path to the CDBS tools (`/grp/hst/cdbs/tools/bin/`) has to be defined in your private directory `PATH` (in the .setenv file), otherwise the whole path to the script has to be used. FITS reference files that are not in standard FITS format cannot be ingested into the databases. To verify them in the `smalls.stsci.edu` domain, run the `fitsverify` tool on the files:

```
fitsverify filename.fits
```

Wild-cards may be used for filenames (filename.fits in this example can be replaced by *.fits).

`Farris_fitsverify` is used instead in the science cluster or while working in the centralized

storage with a Mac or Linux OS. This is because the `fitsverify` script was updated recently to fitsverify version 4.13 (which is compliant with the GSFC Heasarc version, distributed by Bill Pence). Because of changes in formatting in the new script, it might not work for some of the old format FITS files. If possible, correct the FITS files to be compliant with the new `fitsverify` version; however, the CDBS software requires only that these are compliant with the old `fitsverify` version (which corresponds to the Allen Farris version). This old version can be accessed in the STScI Science cluster with the command line :

```
farris_fitsverify filename.fits
```

(Also, note that in the `smalls.stsci.edu` domain the `fitsverify` and `farris_fitsverify` versions are the same.) The output of these two versions will differ considerably. A sample output for the `fitsverify` (v4.13) for the FITS binary table "p9r19206o_tds.fits" looks like this:

```
                    fitsverify 4.13 (CFITSIO V2.510)
                    --------------------------------

File: p9r19206o_tds.fits

2 Header-Data Units in this file.

=================== HDU 1: Primary Array ====================

49 header keywords

Null data array; NAXIS = 0

=================== HDU 2: BINARY Table =====================

  48 header keywords

 TDS(1) (10 columns x 6 rows)


Col# Name (Units)               Format
  1  OPT_ELEM                   12A
  2  WAVELENGTH (angstrom       60D
  3   TIME (MJD)                12D
  4  SLOPE (percent per y       720D
  5  NWL                        1I
  6  NT                         1I
  7  REFTEMP (K)                1D
  8  TEMPSENS (percent /K       60D
  9  PEDIGREE                   67A
```

```
 10  DESCRIP                 67A

++++++++++++++++++++++ Error Summary ++++++++++++++++++++++

 HDU#  Name (version)  Type            Warnings Errors
 1                     Primary Array   0        0
 2         TDS (1)     Binary Table    0        0

*** Verification found 0 warning(s) and 0 error(s). ****
```

On the other hand, output from the farris_fitsverify 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 [1024,10w24]
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 (18/12/00 instead of the correct format Dec. 18, 2000)
- missing `PCOUNT` and `GCOUNT` keywords in extension headers

Note that this command checks for standard FITS header keywords and does not report any missing instrument-dependent header keywords; i.e., those that give the information necessary to define the observing mode to which these files apply and that should be present in all reference files. A complete list of the standard FITS header keywords can be found in ICD-47.

5. Run the CDBS `certify` tool on the standard FITS or GEIS header files

The CDBS `certify` tool performs additional checking on the syntax and keyword values in the FITS files or the GEIS header files (WFPC2 uses GEIS files), ensuring adherence to ICD-47 specifications for each type of reference file. Instrument specific header keywords and columns (in a table file) that are necessary for the correct selection of a reference file will be checked. This tool is run by typing the following command in a UNIX shell.

```
certify filename
```

Wildcards may be used for filenames, e.g., **.fits* or *.*h. The `certify` tool does not check all the keyword syntax and values in the reference file, but only those that are used specifically in CDBS, OPUS, and DADS systems for selecting and tracking the reference files. More detailed documentation on the `certify` task is available in postscript format on the CDBS web page (http://www.stsci.edu/hst/observatory/cdbs/documents/). A complete list of the standard instrument-dependent header keywords can be found in ICD-47. The required keywords are specified to `certify` via CDBS template files. There is one template file for each reference file type and these are located in the CDBS working areas of the STScI science cluster and the `smalls.stsci.edu` domain. A sample of the template file for the STIS PHT 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
```

A sample of the output of `certify` for a file with no errors:

```
== Checking mama2_PFL.fits ==
```

while the output for a file that has a problem would be:

```
== 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 or keyword values. The list of required and valid values for the header keywords can be found as well in ICD-47. If this check does not reveal the source of the error, contact the INS/CDBS Team.

6. Perform a detailed quality assessment of the data in the reference files.

The integrity of the calibration reference files is critical for the production of high-quality science results. It is therefore important that before delivering them to the pipeline or SYNPHOT database, each Instrument group thoroughly test and validate them to the best of their ability. In addition to checking for proper structure and syntax, checking the data is essential.

For those cases where automatic checking has not been implemented, there are many IRAF tasks that can be used to examine files. For headers: `eheader`, `imhead`, `hedit`, `tlcol`; for data: `listpix`, `mstat`, `tread`, `tstat`, `display`. Consistency is strongly recommended for the testing and integrity checking of the reference files. Some of the questions that might be answered before delivering the files could be: Do the new files look like the old ones? If not, why? Have the column `PEDIGREE` values been updated according to the changes made in the rows? Do the comments in the updated row have the correct information? Are the error columns and error extension consistent with the data? Keep careful notes on how you determined that the new reference file was good. For image reference files, it is good to run the IRAF task `imstat` on the file and determine if the values are reasonable in all extensions: science, error and data quality. Often, plotting or displaying the file will reveal obvious problems that otherwise might be missed.

7. In the case of table PIPELINE reference files, check all rows are present and which changed.

This step is necessary to ensure that when the table reference file was changed, all of the new rows were added and that needed rows were not accidentally deleted. This will also help in gathering information about the rows that have changed and that should be listed in point 11a of the delivery form. Knowledge of the rows that changed and their level of change is now required for identification of the data that should be re-processed by the static archives at STScI, CADS, and ECI. A script called `compare_table.pro` was developed for this testing. This script compares two FITS reference file tables, looking for missing elements by checking the differences in each row between the new and old reference files. The comparison elements are limited to those specified in the COLUMNS parameter, which must be specified for the program to run. The parameter COLUMNS is an n-element array of strings with the names of the columns to compare within the table. When a combination is missing in either the old or new file, the procedure will flag it in the standard output. If the parameter SAVEFILE is set to a value other than 0, the output of this procedure will be saved in a file called "compare_table.out". The calling sequence for the procedure is shown below.

```
idl>compare_table,'my_new_file.fits','old_reference_file.fits',$
COLUMNS=['ccdchip','filter1','filter2'],SAVEFILE=1
```

where 'my_new_file.fits' is the name of the new reference file, 'old_reference_file.fits' is the old reference file, 'ccdchip, filter1, and filter2' are the columns to be compared, and the optional SAVEFILE parameter is set to a value other than 0 to have the output written to a file called "compare_table.out".

Any changes should be examined to be sure they were the intended changes. These differences should be listed in point 11a of the delivery form when the files are delivered to CDBS. This script can be found in the Science Cluster in the STScI IDL area and in `/data/garnet2/idl/stsci/` or in the *srefpipe* account */calib/cdbs_delivery/usefulscripts/* directory. More information on this procedure is located at `http://www.stsci.edu/hst/observatory/cdbs/deliveries/compare_table_pro`.

8. Test the reference files using CALXXX or SYNPHOT and ETC.

CALXXX must be used to calibrate actual data with the reference files being prepared for delivery. The highest level check should verify that the appropriate stages of CALXXX do not crash and that the calibrated data makes sense. Make sure that the proper CALXXX switches are set to `PERFORM` so the files being tested are actually used. Real datasets should be calibrated with both the new reference file and the old reference file, and the differences understood and thoroughly documented. Even in the case of simple changes to the file, the deliverer should make sure that CALXXX runs using these files.

SYNPHOT data files, on the other hand, have to be tested using the SYNPHOT and ETC software (two independent tests using different tools). The ETC software relies heavily on the

SYNPHOT package to predict count rates and S/N values via calls to SYNPHOT tasks. However, the results that the ETC produces also depend on calculations performed by the ETC software itself. In order to maintain the integrity of both systems, changes made in SYNPHOT data files should also be tested against the ETC software. Unfortunately, the testing environment that was used for these tests is not currently working and therefore these files cannot be tested against the ETCs now. In any case and for documentation, we will keep the information related to the old ETC test infrastructure. Once the test area is working again this document will be updated and the instrument teams will be notified.

8.1 Testing the file in SYNPHOT

The testing under SYNPHOT is accomplished using the `stsdas.hst_calib.synphot` package in IRAF. Here, it is assumed that the files are being tested in the STScI Science cluster or in a system that mounts the centralized storage area /grp. If working on a different platform, the paths to the directories mentioned below will change.

1. The first step of this process is to create a dummy "Master Component Table" (TMC table) that points to the data files you are delivering. For this, make a copy of the most current TMC table in a working directory. All the TMC tables are located in the directory `/grp/hst/cdbs/mtab/` in the centralized storage area. Although there are several TMC tables in this directory, the most up to date is listed last in alphanumeric order. After copying this file to the working directory, rename it using a naming convention that puts it last in an alphabetically sorted list of the `mtab/` directory. This is because this file will also be used for the test in the ETC environment, for which this naming convention is important (see next section).

2. Now edit the row(s) for the COMPONENT(s) associated with the data file(s) you are delivering. If there is more than one row in this table for the same COMPONENT name, make sure you edit only the last one. (This will change when we migrate to Pysynphot.) Change the `FILENAME` column of this row to point to the SYNPHOT data file being delivered. You can edit this file via the IRAF task `tedit`. Note that there is a required 68 characters limit for this column. In order to prevent a problem with long path names, as well as for the use of this file for the testing with the ETC software, we strongly recommend defining a logical IRAF variable for this directory. In the ETC test environment this file has to be called "comptest", so we recommend using this name here as well. In this case the `FILENAME` column value will have the following format:

   sy> comptest$rootname_new_syn.fits

   In order to test the files with SYNPHOT, the IRAF logical variable "comptest" also has to be defined. This can be accomplished by typing in the IRAF command line:

   sy> set comptest = "/data/mypath/myworkingdir"

3. Now modify the parameter "cmptbl" in the `stsdas.hst_calib.synphot.refdata` task with the name of the dummy TMC table using the same IRAF logical variable name to point to the working directory.

4. If you are delivering a "Master Graph Table" (TMG table) together with your new SYNPHOT data file(s), you also have to modify the parameter "grtbl" of the `refdata` task. This is done in the same way as for the TMC table.

5. Now run the SYNPHOT task that will use this data file. Note that if you're also delivering the "HST Thermal Component Table (TMT table), this cannot be tested at this point. This will be possible when testing the ETCs (see section 8.2)

6. Make sure that you used the correct SYNPHOT data files by running the task `stsdas.hst_calib.synphot.showfiles`.

7. If you are delivering a new TMG file, you will also have to run the task `synphot.grafcheck` to validate the structure of the graph table. (e.g. `grafcheck qc80137jm_tmg.fits`)

For more information on the SYNPHOT package and tasks, refer to the *Synphot User's Guide* (http://www.stsci.edu/resources/software_hardware/stsdas/synphot/SynphotManual.pdf).

8.2 Documantation about the old infrastructure to test the file with ETC

In the old test environment, the ETC Team created a testing environment where the Instruments and OTA teams were able to test new SYNPHOT files. In here we will provide the information on how to set and run the ETC tests in the old ETC Test infrastructure; however and as we mentioned before, this test cannot be done now so the instrument teams can skip this step until further notice.

"In the ETC environment, the deliverer is able to identify those ETC modes that were affected by the changes made to the SYNPHOT data file(s) and make sure that the differences between the old files and the ones being tested are the expected ones. In order to consistently verify the integrity of the system, the instrument teams have been maintaining a complete ETC test suite that cover most of the cases where the new SYNPHOT data file will affect the ETCs output. If the current test suite is not complete, please contact the INS/CDBS Team to help you update it. The test of SYNPHOT data files against the ETCs has to be done in `"panthro.stsci.edu"`, which is part of the Linux cluster. In order to be able to run the ETC regression test in this computer, some environment variables and libraries need to be set in the user's account.

- First, it is necessary to set the environmental variables that give the ANT and JAVA directory paths. This can be accomplished by adding the following lines to your .setenv file:

  setenv ANT_HOME /usr/local/apache-ant-1.6.5
  setenv JAVA_HOME /usr/java/jdk1.5.0_02
  setenv PATH .:/usr/local/apache-ant-1.6.5/bin:/usr/java/jdk1.5.0_02/bin:'binpath -sh':${PATH}

- In the `HOME` directory, there must exist a directory called `.ant/lib/` (note that the directory name starts with a period), and it must contain the file `junit.jar`. This file can be found in `/etc_test/data1/junit3.8.1/` directory.

Once the account is set up properly the test can be run, but before we go into this, lets first talk a little about the ETC test area. The area to run the test for the INS groups is set in a special directory: `/etc_test/data1/insTest/`. In this directory resides all the executables, test suite, and reference runs needed for the test. Nothing in this directory has to be changed; however, it is necessary to be in this directory in order to run the test script. There are several scripts that run the regression test for each of the instruments (ACS, NICMOS, STIS) as well as generate the baseline run against which the new files will be compared. Both of these tests look to the same test suite, the main difference is that the baseline run just generates the ETC output files, while the regression test additionally makes a comparison between the new ETC run and that of the baseline run. Since we want to make sure that the only differences we see are due to the files we are testing, we recommend that before we transfer the new SYNPHOT tables to the etc_test area, you first run the script that generates the baseline test run. With this in mind the chronological steps for the ETC test are as follows.

1. Run the baseline generation. There is one script per instrument:

Table 1: ETC test scripts

Instrument	Generate Baseline script	Make Comparison script
ACS	acsGen	acsCheck
NICMOS	nicmosGen	nicmosCheck
STIS	stisGen	stisCheck
All	insGen	insCheck

For example, to run the NICMOS script type

```
etc_test> ./nicmosGen
```

This will generate the file `etc.nicmosGen.out` in your home directory. This file contains the output of the ETC's run. In the above table, the script `insGen` will run the script for all the instruments at the same time.

2. Now, you have to set up the system to see the new SYNPHOT files(s) being tested. First you should copy them to the directory `/etc_test/data1/insTest/comptest/` in `etc_test.stsci.edu`. Note that all the current SYNPHOT data files and master tables that are used for all INS/ETC test runs are located in the directory `/etc_test/data1/cdbsTest/comp/`; however, the INS/ETC testing environment will also look for files in the `/etc_test/data1/insTest/comptest/` directory, which is called through the logical variable "comptest" mentioned in point two of section 8.1.

3. In order to use the new SYNPHOT data file(s) in the INS/ETC environment test, a dummy TMC table is also needed. If the TMC table was already modified using the guidelines outlined in points one and two of section 8.1 above, then continue with the next step. If not,

   - create a dummy TMC table using the guidelines outlined in step one of section 8.1, making sure that the modifications to this dummy TMC table are done in a working directory and not in the `mtab/` directory. In the case of `etc_test.stsci.edu` the TMC tables are located in the `/etc_test/data1/cdbsTest/mtab/`.
   - the dummy TMC table should be renamed using a naming convention that makes this file the last in an alphabetically sorted list of the `mtab/` directory. This is because, given the list of TMC tables in `/etc_test/data1/cdbsTest/mtab/`, SYNPHOT sorts them alphabetically by name and uses only the last file in the list.
   - Modify the TMC table using the guidelines outlined in step two of section 8.1, making sure that the `FILENAME` path has the format `comptab$rootname_new_syn.fits`; this is, the filenames are preceded by the logical `comptest$`. In this case, however, it is not necessary to set this logical variable within IRAF.
   - If you are also delivering a TMT table with the Synphot Throughput tables, you have to modify this file using the same guidelines described in step two of section 8.1 for the TMC table. This is because the TMT file has, in the column `FILENAME`, the path to the thermal libraries. The format for this column is `crnicmoscomp$nic2_edge_001_th.fits`; this is, the filenames are preceded by the logical `crnicmoscompt$`. In this case, however, it is not necessary to set this logical variable within IRAF. Note that once the test is done, the `FILENAME` entries modified according to step two of section 8.1 have to be renamed to their original logical `crnicmoscomp$`. This is because this file is not touched by the INS/CDBS Team, and delivered as is. Once this is done, rename this file to a name that makes it to appear last when sorting alphabetically all of the TMT files.
   - If you are also delivering a TMG table, rename it to a name that makes it last when all the TMG tables in the `etc_test.stsci.edu mtab` directory are alphabetically sorted.

4. For the INS/ETC test, the dummy TMC table (and the TMG and TMT tables, if applicable) should be temporarily placed in the `/etc_test/data1/cdbsTest/mtab/` directory. (Once the test is done, this dummy table(s) has to be deleted from the `mtab/` directory).

5. If you are delivering a new TMT table, we recommend testing it in SYNPHOT now. It is in this computer where the dummy/new TMT table can be copied to the `mtab` directory and be picked up by SYNPHOT.

6. Now we are ready to run the comparison test. Column 3 in Table 1 lists the scripts you should use for this test. For example, for NICMOS the command line is:

   ```
   etc_test> ./nicmosCheck &
   ```

   If using the script that does the comparison for all the instruments at the same time, it will take about 1 hour to run; for individual instrument it will take about 20 minutes. These scripts will produce a file called `etc.xxxxCheck.out` in the home directory; where `xxxx` is

the instrument name. This file contains a list of the test suite input parameters and the ETC output and the differences found between the current run and the base run created in step 1. The information in the output file can be divided in two parts. The first half of the output file provides information on the input parameters for the test cases used; the second half contains the differences found for each of the test cases. To see the output of the test open the file in your favorite editor and go to the end of the file. From the end of the file, scroll up, if no differences are found, the output looks something like this:

```
[junit] NICIMAG844 ----------------
[junit] NICIMAG845 ----------------
[junit] NICIMAG846 ----------------
[junit] NICIMAG847 ----------------
[junit] NICIMAG848 ----------------
[junit] NICIMAG849 ----------------
```

Here, the dashes for the test cases #844 to #849 indicate that the NICMOS IMAGING mode have no differences from the reference run cases. If there are differences these will appear as follows:

```
[junit] NICIMAG1483 ----------------
[junit] Time = 9.478665205037517; reference = 9.733099656333973
***** FAIL ******
[junit] TotalDarkCurrent = 2.8435996745057497; reference=2.9199300129277814
***** FAIL ******
[junit] TotalRate = 27.236754972300826; reference = 26.568083026796238
***** FAIL ******
[junit] TotalESZLCounts = 10.81022631694853; reference = 11.100403672287712
***** FAIL ******
[junit] TotalThermalCounts = 9.488143870242554E-4; reference =
9.742832755990306E-4
***** FAIL ******
[junit] SourceRate = 44.68496704101562; reference = 43.52667999267579
***** FAIL ******
```

where `NICIMAG1483` indicates that the test case `#1483` corresponds to a NICMOS imaging ETC mode. The differences listed here are for the `Time, TotalDarkCurrent, TotalRate, TotalESZLCount, TotalThermalCounts, SourceRate` parameters, where the values derived from the reference run are indicated as `reference`. Make sure that all the reported differences agree with the changes made to the new SYNPHOT data file(s). Further information on the ETC's input and output for this run can be found in the first part of this output file.

7. Once the real differences are understood, the test is completed and the new SYNPHOT data file(s) can be delivered. Send, along with the delivery form, the location of the `./xxxxCheck` run, documentation on the differences found, and appropriate explanation on how these

differences relate to the new delivered SYNPHOT data file(s). This information is needed by the INS/CDBS Team to assess compliance with CDBS guidelines and keep the INS/ETC testing environment up to date.

8. If you are also delivering a TMT table, rename the changed rows to their original logical `crnicmoscomp$`. This is because this file is not touched by the INS/CDBS Team, and will be delivered as is.

9. Remember that after the test is completed, the dummy TMC table (also the TMG and TMT tables if applicable) has to be erased from the `mtab/` directory; this will prevent any confusion for future testing of SYNPHOT data files.

More details on the reference runs or on how the SYNPHOT/ETC test works can be found in Appendix A of this document. If considered necessary, and for the purpose of testing SYNPHOT data files in the ETC environment, changes to the current test cases or parameter list can be made. Also, new cases can be implemented. For any changes, please contact the INS/CDBS Team. Later on, we will provide the necessary documentation on the procedures to follow in these cases. ”

9. 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 out the delivery template (shown below) and send it to the INS/CDBS Team (cdbs@stsci.edu). This template, besides helping the INS/CDBS Team to correctly deliver the files to the Data Management Systems, can be used as a check list for all the steps indicated above. Make sure that the files and directory where the files are located are accessible by anybody. The delivery template can also be found in the INS/CDBS web page under Delivery Notification Template (http://www.stsci.edu/hst/observatory/cdbs/deliveries/delivery_form.html) and contains the following information:

```
1- Name of deliverer:
(other e-mail addresses)
2- Date of delivery:
3- Instrument:
4- Type of file (bias,pht,etc.):
5- History section in header [0] complete?  (yes/no):
6- USEAFTER, PEDIGREE, DESCRIP, and COMMENT have been checked?  (yes/no)
7- CDBS Verification complete?  (fitsverify, certify,etc.):
8- Should these files be ingested in the OPUS, DADS and CDBS databases?  (if not
indicate clearly which ones)
9- Files run through CALXXX or SYNPHOT? (yes/no):
10- Does it replace an old reference file?  (yes/no):
10a- If yes, which one?
11- What is the level of change of the file?  (e.g.  compared to old file it
could be:  SEVERE, MODERATE, TRIVIAL, 1%, 5% etc.):
```

```
11a- If files are tables with information about different observing modes, please
indicate exactly which rows have changed.
12- Description of how the files were ‘‘tested’’ for correctness:
13- Additional considerations:
14- Reason for Delivery:
15- Disk location and name of files:
```

In this template, in the field "(other e-mail addresses)" you should provide all the e-mail addresses of people or mailing lists that should be notified of any problem or success of the delivery. The field "Date of delivery" should have the date when you deliver the file to the INS/CDBS Team. In question number five, regarding the completion of the history section, you should put "yes" only when you have made sure that the history section of the file describes the changes made to the file or explains how the values in the file were derived. If you deliver the file with answer "no", we might have to contact you again to make sure this section gets filled properly. Although most of the reference files are usually delivered to the OPUS, DADS, and CDBS databases, there could be cases when teams choose not to do so; indicate this in question number eight. In question ten, we want to confirm that any delivered reference files correctly replace, when applicable, other files currently in the system. We need to know the name of the file that will be replaced.

In order to properly populate the databases, we also need to know the level of change of the delivered files compared to old files or if it is a new file. This information should be given in question 11. If the file is a new reference file or it covers a range of dates not covered previously, the level of change should be set to SEVERE or 100%. Another case when it should be considered as SEVERE is when the change warrants recalibration of the data. If the changes will have a moderate impact in the results of the calibration, or just make them better but do not warrant recalibration, then the change is MODERATE. Also, you could specify the percent change; for example, for the throughputs or images. If the changes made to the file were to correct minor mistakes only and they will not impact or will have negligible impact on the calibrated products, the change level is considered TRIVIAL.

Note that in the case of tables we will need more detailed information on which rows changed. This is in particular important for tables that contain information about several observing modes. For these, we need to know which datasets will be affected by the delivery; so the mirror sites, and the HLA, can identify with confidence which particular datasets will need recalibration. For these cases, please provide the list of rows that were changed. You can determine which rows were changed by running the script `'compare_table.pro'` discussed in section 7 above. The script is located in the Science Cluster in the STScI IDL area. If you do not have that set in your IDL PATH, you can also find a copy of the procedure in `/data/garnet2/idl/stsci/`. This list of changed rows should be provided in line '11a' in the delivery form. In the case of SYNPHOT tables we do not need to know the particular rows changed.

We also need information on how the files were "tested" for correctness. In question 12, briefly describe which kind of test you performed to assess the scientific correctness of the files. For example: "Verified no change for first order modes when compared to previously ingested reference file. Calstis ignores this file for echelle and imaging modes." or "All extensions examined with imstat to count number and size of hot pixels. Run through calstis with data sets from test

suite to ensure that changes in calibrated output were minimal as expected." In question 13, describe any special needs associated with the delivery. Is the installation meant to be coincident with a specific OPUS build or are there other timing requirements or urgency? Note that the INS/CDBS Team makes every effort to deliver your files to OPUS within 24 hours of receiving the delivery form, and will contact you if this schedule cannot be met. In question 14, please list details of the reason the files are being delivered, i.e. "Update gains for instrument combination X and Y" or "WFPC2 darks to be used for data taken after Dec. 1, 2008". Finally, in question 15, give the directory path where the files are located. Also, provide a complete list of the files delivered. Make sure that the files are readable by anybody so we can get a copy of them . List all the reference files that are being delivered. For example,

```
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp03_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp04_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp05_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp06_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp07_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp08_hrc_rdrk.fits
-rw-r--r-- 2 acsref 10532160 Mar 24 23:13 grp09_hrc_rdrk.fits
```

Mailing the form to the INS/CDBS Team concludes the reference file creator/tester work. Once the files are in the system, the person delivering the file will be notified.

Acknowledgements

We would like to thank M. Lallo, E. Barker and H. McLaughlin for their comments and suggestions. Their input was important to make this document more clear and complete.

References

C. Cox & C. Tullos CDBS 1998, "*Delivering Calibration Reference Files*"

R. Diaz-Miller, STIS TIR 2003-01, "*Assessment and Delivery of Spectrographs Calibration Reference Files*"

Laidler et al, 2005, "*Synphot User's Guide*", Version 5.0 (Baltimore: STScI). B. Simon, CDBS 1996, "*certify*"

ICD-47 (http://www.ess.stsci.edu/projects/distribution/ICD47/ICD-47_RevF.pdf)

Appendix A

The check for changes in the ETC output due to changes in SYNPHOT data files is done in a testing environment that is a replica of the test environment used by the APT team to test the ETC software. The test is done using test cases stored in the `insTest` directory.

Each instrument has its own test suite, which is the same used for the test of the ETCs. The test suite, or the files with the parameters used for testing, are located in the directory `/etc_test/data1/insTest/ETC/JUNIT/testFiles/regressionRepository/`. In this directory there is a subdirectory for each of the instruments; which in turn contain subdirectories for each observation mode (e.g. IMAGING or SPECTROSCOPIC), source type (e.g. EXTENDED, POINT), and in some cases by detector (e.g. HRC, WFC). An example of these files is the NICMOS test case `log979.xml` for point imaging mode. The first part of the file looks like this:

```
<WebETC_Request>
<RequestID>NICIMAG979</RequestID>
<fIndex>-1</fIndex>
<SNR>10</SNR>
<nic3Mode>ACCUM</nic3Mode>
<detector>nic1</detector>
<fsorigin>SpectrumBlackBody</fsorigin>
<EarthshineMult>1.0</EarthshineMult>
<fcalfile>GD71</fcalfile>
<felambda>5500</felambda>
<femag>15</femag>
<fbbtemp>5500</fbbtemp>
<fplambda>5500</fplambda>
<science_mode>Imaging</science_mode>
<fRedshift>0.0</fRedshift>
<ZodiSpec>ZodiStandard</ZodiSpec>
<background0>edu.stsci.hst.HstBackgroundModelSky</background0>
<nic2Mode>ACCUM</nic2Mode>
<fL1_flux>0.</fL1_flux>
<fOtherFile />
<fStellar>O5_V</fStellar>
<fL3_flux>0.</fL3_flux>
<fL1_fwhm>0.</fL1_fwhm>
<ModelName>HST/NICMOS</ModelName>
<nic2filt0>F110W2</nic2filt0>
<fL3_fwhm>0.</fL3_fwhm>
<fbpgsfile>O5V</fbpgsfile>
<extractionRegionSquare>1</extractionRegionSquare>
<febmvtype>gal1</febmvtype>
<ZodiMag>30.0</ZodiMag>
<mode_access>supported</mode_access>
<crsplit>1</crsplit>
<gain>1</gain>
<EarthshineSpec>EarthshineStandard</EarthshineSpec>
<nic1Mode>ACCUM</nic1Mode>
<fdiameter>10</fdiameter>
<fL1_center>0.</fL1_center>
```

<**Time**>**1000**</**Time**>
<fpmag>22</fpmag>
<instrument>NICMOS</instrument>
<fL2_center>0.</fL2_center>
<EarthshineStandard>Average</EarthshineStandard>
<fftype>fpmag</fftype>
<ZodiStandard>Average</ZodiStandard>
<fnonstellar>Elliptical</fnonstellar>
<ExposureName>server exposure</ExposureName>
<ZodiMult>1.0</ZodiMult>
<fpunit>Bessell/H</fpunit>
<fL3_center>0.</fL3_center>
<fL3_center>0.</fL3_center>
<feunit>Johnson/V</feunit>
<xRegionType>Square</xRegionType>
<**simmode**>**SNR**</**simmode**>
<fIsLambda>true</fIsLambda>
<fL2_flux>0.</fL2_flux>
...

As you might notice, the test cases are in XML format; so the nomenclature is difficult to decipher. Furthermore, the files provide values for all the exposure parameters, spectral distribution, etc., even if those are not necessary for the particular test case. The way in which the software determines which mode and parameters should be used is via special parameters also listed in this file. For example, both parameters: `Time` and `S/N` (shown in bold face), have assigned values but only the mode determined by the parameter `<simmode>` (SNR or Time) is used. The logs for the reference runs can be found in these directories too; and these will be used to compare against the output of the new run. Another file in these directories is the parameter list file (`parameterList.txt`). This file contains a list of the parameters that are used to limit the amount of information compared in each regression test. For example, the following list indicates that, from all the parameters that result from an ETC run, only the Time, Slitless Source Count Rate, Total Sky Counts, Sky Count Rate, Total Counts, and Total Count Rate in a Box are compared:

```
<requestParameters>
<Time> <TotalSourceCounts Class="edu.stsci.hst.SpectrumResultHandler" />
<SlitlessSourceTotalCountRate/>
<TotalSkyCounts/>
<SimpleSlittedESZLSkyRate Class="edu.stsci.hst.SpectrumResultHandler"
switchName="SlittedSpectroscopy" switchValue="true" />
<SkyRate/>
<TotalCounts/>
<TotalRateForBox/>
</requestParameters>
```

Note also that the reference runs, used for the comparison against the test run, might have been generated a while ago, so these will need to be modified every time a new SYNPHOT data file is delivered or when there are changes to the ETCs. Also, the test suite is fixed. If a new test needs to be added, or current test cases have to be modified, the reference cases have to be regenerated. Currently the update of test cases is done in a controlled way via builds against the ETC software. However, in order to simplify the determination and review of these possible new test cases, a development area was created where the instrument teams can experiment with them. Once it is determined that these new test cases should become a permanent part of the test suite, these should be delivered to the ETC group via a PR against ETC and CDBS (the latter as secondary subsystem). The ETC group will make sure that those test cases are incorporated to the current ETC standard test suite and that the reference run is re-generated. The development area where new test cases can be tested is located in: `/etc_test/data1/insTest/ETC/JUNIT/testFiles/instrumentDev/`; where the script to generate the reference run is called `insGenerate`. The `insCheck` script in this directory will work just as the `insTest` script. It is recommended to anyone modifying the test cases, the parameter list, or developing new test cases, to consult the ETC development team. Their knowledge of the software and the XML file formats can help to make the changes almost painless.