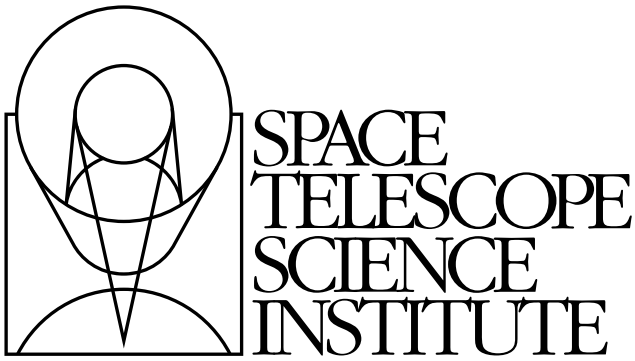

STWCS Documentation

Release 2011.xx

Nadezhda Dencheva, Warren Hack

May 03, 2013

CONTENTS

This package provides support for WCS based distortion models and coordinate transformation. It relies on PyWCS (based on WCSLIB). It consists of two subpackages: UPDATEWCS and WCSUTIL. UPDATEWCS performs corrections to the basic WCS and includes other distortion infomation in the science files as header keywords or file extensions. WCSUTIL provides an HSTWCS object which extends pywcs.WCS object and provides HST instrument specific information as well as methods for coordinate tarnsformaiton. WCSUTIL also provides functions for manipulating alternate WCS descriptions in the headers.

Contents:

CHAPTER

ONE

WCSUTIL

This package provides an HSTWCS class which performs WCS based coordinate transformations and a module for managing alternate WCS's.

1.1 HSTWCS API

class `stwcs.wcsutil.hstwcs.`**`HSTWCS`** (*fobj=None*, *ext=None*, *minerr=0.0*, *wcskey=' '*)

Create a WCS object based on the instrument.

In addition to basic WCS keywords this class provides instrument specific information needed in distortion computation.

Parameters

fobj: string or PyFITS HDUList object or None :

a file name, e.g j9irw4b1q_flt.fits a fully qualified filename[EXTNAME,EXTNUM], e.g. j9irw4b1q_flt.fits[sci,1] a pyfits file object, e.g pyfits.open('j9irw4b1q_flt.fits'), in which case the user is responsible for closing the file object.

ext: int, tuple or None :

extension number if ext is tuple, it must be ("EXTNAME", EXTNUM), e.g. ("SCI", 2) if ext is None, it is assumed the data is in the primary hdu

minerr: float :

minimum value a distortion correction must have in order to be applied. If CPERRja, CQERRja are smaller than minerr, the corersponding distortion is not applied.

wcskey: str :

A one character A-Z or " " used to retrieve and define an alternate WCS description.

1.2 HSTWCS Examples

1.2.1 Create an HSTWCS Object

- Create an HSTWCS object using a pyfits HDUList and an extension number

```
fobj = pyfits.open('some_file.fits')
w = wcsutil.HSTWCS(fobj, 3)
```

- Create an HSTWCS object using a qualified file name.

```
w = wcsutil.HSTWCS('j9irw4b1q_flt.fits[sci,1]')
```

- Create an HSTWCS object using a file name and an extension number.

```
w = wcsutil.HSTWCS('j9irw4b1q_flt.fits', ext=2)
```

- Create an HSTWCS object from WCS with key 'O'.

```
w = wcsutil.HSTWCS('j9irw4b1q_flt.fits', ext=2, wcskey='O')
```

- Create a template HSTWCS object for a DEFAULT object.

```
w = wcsutil.HSTWCS(instrument='DEFAULT')
```

1.2.2 Coordinate Transformation Examples

All coordinate transformation functions accept input coordinates as 2D numpy arrays or 2 sequences of X and Y coordinates.

```
inpix = np.array([[1., 2.], [1,3], [1,4], [1,5]])
```

or

```
X = [1.,1.,1.,1.]
Y = np.array([2.,3.,4.,5.])
```

In addition all transformation functions require an `origin` parameter which specifies if the coordinates are 0 or 1 based. For example in FITS and Fortran, coordinates start from 1, while in Python and C, the index of the first image pixel is (0,0).

- Apply the entire detector to sky transformation at once:

```
outpix=w1.all_pix2sky(inpix,1)
outpix=w1.all_pix2sky(X, Y,1)
```

- The same transformation can be done in separate steps:

1. Apply the detector to image correction

```
dpx = w.det2im(inpix,1)
```

2. Aply the SIP polynomial distortion

```
spx = w.sip_pix2foc(dpx, 1)
```

3. Apply the non-polynomial distortion from the lookup table

```
lutpx = w.p4_pix2foc(dpx,1)
```

4. The undistorted coordinates are the sum of the input coordinates with the deltas for the distortion corrections.

```
fpix = dpx + (spx-dpx) +(lutpx-dpx)
```

5. Finally the transformation from undistorted to world coordinates is done by applying the linear WCS.

```
wpix = w.wcs_pix2sky(fpix, 1)
```

1.3 User Interface: altwcs

The functions in this module manage alternate WCS's in a header.

stwcs.wcsutil.altwcs.**archiveWCS** (*fname*, *ext*, *wcskey=' '*, *wcsname=' '*, *reusekey=False*)

Copy the primary WCS to the header as an alternate WCS with wcskey and name WCSNAME. It loops over all extensions in 'ext'

Parameters

fname: string or pyfits.HDUList :

a file name or a file object

ext: an int, a tuple, string, or list of integers or tuples (e.g.('sci',1)) :

fits extensions to work with If a string is provided, it should specify the EXTNAME of extensions with WCSs to be archived

wcskey: string "A"-"Z" or " " :

if " ": get next available key if wcsname is also " " or try to get a key from WCSNAME value

wcsname: string :

Name of alternate WCS description

reusekey: boolean :

if True - overwrites a WCS with the same key

See also:

wcsutil.restoreWCS

Copy an alternate WCS to the primary WCS

Examples

Copy the primary WCS of an in memory headrlet object to an alternate WCS with key 'T'

```
>>> hlet=headerlet.createHeaderlet('junk.fits', 'hdr1.fits')
>>> altwcs.wcskeys(hlet[1].header)
['A']
>>> altwcs.archiveWCS(hlet, ext=[('SIPWCS',1),('SIPWCS',2)], wcskey='T')
>>> altwcs.wcskeys(hlet[1].header)
['A', 'T']
```

stwcs.wcsutil.altwcs.**restoreWCS** (*f*, *ext*, *wcskey=' '*, *wcsname=' '*)

Copy a WCS with key "WCSKEY" to the primary WCS

Reads in a WCS defined with wcskey and saves it as the primary WCS. Goes sequentially through the list of extensions in ext. Alternatively uses 'fromext' and 'toext'.

Parameters

f: string or pyfits.HDUList object :

a file name or a file object

ext: an int, a tuple, string, or list of integers or tuples (e.g.('sci',1)) :

fits extensions to work with If a string is provided, it should specify the EXTNAME of extensions with WCSs to be archived

wcskey: a charater :

"A"-"Z" - Used for one of 26 alternate WCS definitions. or " " - find a key from WCSNAMe value

wcsname: string (optional) :

if given and wcskey is " ", will try to restore by WCSNAME value

See also:

archiveWCS, restore_from_to

stwcs.wcsutil.altwcs.**deleteWCS**(*fname*, *ext*, *wcskey=' '*, *wcsname=' '*)

Delete an alternate WCS defined with wcskey. If wcskey is " " try to get a key from WCSNAME.

Parameters

fname: sting or a pyfits.HDUList object :

ext: an int, a tuple, string, or list of integers or tuples (e.g.('sci',1)) :

fits extensions to work with If a string is provided, it should specify the EXTNAME of extensions with WCSs to be archived

wcskey: one of 'A'-'Z' or " " :

wcsname: string :

Name of alternate WCS description

stwcs.wcsutil.altwcs.**wcsnames**(*fobj*, *ext=None*)

Returns a dictionary of wcskey: WCSNAME pairs

Parameters

fobj: string, pyfits.HDUList or pyfits.Header :

fits file name, pyfits file object or pyfits header

ext: int or None :

extension number if None, fobj must be a header

stwcs.wcsutil.altwcs.**wcskeys**(*fobj*, *ext=None*)

Returns a list of characters used in the header for alternate WCS description with WCSNAME keyword

Parameters

fobj: string, pyfits.HDUList or pyfits.Header :

fits file name, pyfits file object or pyfits header

ext: int or None :

extension number if None, fobj must be a header

stwcs.wcsutil.altwcs.**available_wcskeys**(*fobj*, *ext=None*)

Returns a list of characters which are not used in the header with WCSNAME keyword. Any of them can be used to save a new WCS.

Parameters

fobj: string, pyfits.HDUList or pyfits.Header :

fits file name, pyfits file object or pyfits header

ext: int or None :

extension number if None, fobj must be a header

stwcs.wcsutil.altwcs.**next_wcskey**(*fobj*, *ext=None*)

Returns next available character to be used for an alternate WCS

Parameters

fobj: string, pyfits.HDUList or pyfits.Header :

fits file name, pyfits file object or pyfits header

ext: int or None :

extension number if None, fobj must be a header

stwcs.wcsutil.altwcs.**getKeyFromName**(*header*, *wcsname*)

If WCSNAME is found in header, return its key, else return None. This is used to update an alternate WCS repeatedly and not generate new keys every time.

Parameters

header: pyfits.Header :

wcsname: str :

Value of WCSNAME

CHAPTER

TWO

UPDATEWCS

UPDATEWCS applies corrections to the WCS of an HST science file and adds reference information as header keywords and fits file extensions so that a science file contains all necessary information to represent astrometrically precise positions. The order in which the corrections are applied is important and is as follows:

- Detector to Image Correction
- Apply Time dependent distortion (if applciable)
- Recomputing the basic WCS
- Apply Velocity Aberration Correction
- Apply polynomial distortion through the SIP coefficients
- Apply non-polynomial distortion

Mathematically the entire transformation from detector to sky coordinates is described by:

$$(x', y') = DET2IM(x, y)$$

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} x' - CRPIX1 \\ y' - CRPIX2 \end{pmatrix}$$

$$\begin{pmatrix} \alpha \\ \delta \end{pmatrix} = \begin{pmatrix} CRVAL1 \\ CRVAL2 \end{pmatrix} + \begin{pmatrix} CD11 & CD12 \\ CD21 & CD22 \end{pmatrix} \begin{pmatrix} u' + f(u', v') + LT_x(x', y') \\ v' + g(u', v') + LT_y(x', y') \end{pmatrix}$$

where f(u',v') and g(u',v') represent the polynomial distortion correction specified as

$$f(u', v') = \sum_{p+q=2}^{AORDER} A_{pq} u'^p v'^q$$

$$g(u', v') = \sum_{p+q=2}^{BORDER} B_{pq} u'^p v'^q$$

where

- x', y' are the initial coordinates x,y with the 68th column correction applied through the DET2IM convention
- u',v' are the DET2IM-corrected coordinates relative to CRPIX1,CRPIX2

- LT_x, LT_y is the residual distortion in the lookup tables written to the header using the FITS Distortion Paper lookup table convention
- A, B are the SIP coefficients specified using the SIP convention

2.1 UPDATEWCS - User Interface

stwcs.updatewcs.**updatewcs**(*input*, *vacorr=True*, *tddcorr=True*, *npolcorr=True*, *d2imcorr=True*, *checkfiles=True*, *verbose=False*)

Updates HST science files with the best available calibration information. This allows users to retrieve from the archive self contained science files which do not require additional reference files.

Basic WCS keywords are updated in the process and new keywords (following WCS Paper IV and the SIP convention) as well as new extensions are added to the science files.

Parameters

input: a python list of file names or a string (wild card characters allowed) :

input files may be in fits, geis or waiver fits format

vacorr: boolean :

If True, vecocity aberration correction will be applied

tddcorr: boolean :

If True, time dependent distortion correction will be applied

npolcorr: boolean :

If True, a Lookup table distortion will be applied

d2imcorr: boolean :

If True, detector to image correction will be applied

checkfiles: boolean :

If True, the format of the input files will be checked, geis and waiver fits files will be converted to MEF format. Default value is True for standalone mode.

2.2 WCS Corrections

2.2.1 Time Dependent Distortion

class stwcs.updatewcs.corrections.**TDDCorr**

Apply time dependent distortion correction to distortion coefficients and basic WCS keywords. This correction **must** be done before any other WCS correction.

Parameters

ext_wcs: HSTWCS object :

An HSTWCS object to be modified

ref_wcs: HSTWCS object :

A reference HSTWCS object

Notes

Compute the ACS/WFC time dependent distortion terms as described in [R1] and apply the correction to the WCS of the observation.

The model coefficients are stored in the primary header of the IDCTAB. D_{ref} is the reference date. The computed corrections are saved in the science extension header as TDDALPHA and TDDBETA keywords.

$$TDDALPHA = A_0 + A_1 * (obsdate - D_{ref})$$

$$TDDBETA = B_0 + B_1 * (obsdate - D_{ref})$$

The time dependent distortion affects the IDCTAB coefficients, and the relative location of the two chips. Because the linear order IDCTAB coefficients ar eused in the computatuion of the NPOL extensions, the TDD correction affects all components of the distortion model.

Application of TDD to the IDCTAB polynomial coefficients: The TDD model is computed in Jay's frame, while the IDCTAB coefficients are in the HST V2/V3 frame. The coefficients are transformed to Jay's frame, TDD is applied and they are transformed back to the V2/V3 frame. This correction is performed in this class.

Application of TDD to the relative location of the two chips is done in `makewcs`.

References

[R1]

2.2.2 Velocity Aberration Correction

class `stwcs.updatewcs.corrections.`**`VACorr`**
Apply velocity aberation correction to WCS keywords.

Notes

Velocity Aberration is stored in the extension header keyword 'VAFACTOR'. The correction is applied to the CD matrix and CRVALs.

2.2.3 Simple Imaging Polynomial Coefficients

class `stwcs.updatewcs.corrections.`**`CompSIP`**
Compute Simple Imaging Polynomial (SIP) coefficients as defined in [R4] from IDC table coefficients.

This class transforms the TDD corrected IDCTAB coefficients into SIP format. It also applies a binning factor to the coefficients if the observation was binned.

References

[R4]

2.2.4 Non-Polynomial Distortion Correction

class `stwcs.updatewcs.npol.`**`NPOLCorr`**
Defines a Lookup table prior distortion correction as per WCS paper IV. It uses a reference file defined by the NPOLFILE (suffix 'NPL') keyword in the primary header.

Notes

- Using extensions in the reference file create a WCSDVARR extensions and add them to the science file.
- Add record-valued keywords to the science extension header to describe the lookup tables.
- Add a keyword 'NPOLEXT' to the science extension header to store the name of the reference file used to create the WCSDVARR extensions.

If WCSDVARR extensions exist and `NPOLFILE` is different from `NPOLEXT`, a subsequent update will overwrite the existing extensions. If WCSDVARR extensions were not found in the science file, they will be added.

It is assumed that the NPL reference files were created to work with IDC tables but will be applied with SIP coefficients. A transformation is applied to correct for the fact that the lookup tables will be applied before the first order coefficients which are in the CD matrix when the SIP convention is used.

2.2.5 Detector to Image Correction

class `stwcs.updatewcs.det2im.`**`DET2IMCorr`**

Stores a small correction to the detector coordinates as a d2imarr extension in the science file.

Notes

For the case of ACS/WFC every 68th column is wider than the rest. To compensate for this a small correction needs to be applied to the detector coordinates. We call this a detector to image transformation. The so obtained image coordinates are the input to all other distortion corrections. The correction is originally stored in an external reference file pointed to by 'D2IMFILE' keyword in the primary header. This class attaches the correction array as an extension to the science file with extname = `d2imarr`.

Other keywords used in this correction are:

`AXISCORR`: integer (1 or 2) - axis to which the detector to image
correction is applied

`D2IMEXT`: string = name of reference file which was used to create
the lookup table extension

`D2IMERR`: float, optional - maximum value of the correction

2.3 UPDATEWCS.UTILS functions

`stwcs.updatewcs.utils.`**`build_sipname`**(*fobj*, *fname=None*, *sipname=None*)

Build a SIPNAME from IDCTAB

Parameters

fobj: HDUList :

pyfits file object

fname: string :

science file name (to be used if ROOTNAMe is not present

sipname: string :

user supplied SIPNAME keyword

Returns

sipname, idctab :

stwcs.updatewcs.utils.**build_npolname**(*fobj*, *npolfile=None*)
Build a NPOLNAME from NPOLFILE

Parameters
fobj: HDUList :

pyfits file object

npolfile: string :

user supplied NPOLFILE keyword

Returns
npolname, npolfile :

stwcs.updatewcs.utils.**build_d2imname**(*fobj*, *d2imfile=None*)
Build a D2IMNAME from D2IMFILE

Parameters
fobj: HDUList :

pyfits file object

d2imfile: string :

user supplied NPOLFILE keyword

Returns
d2imname, d2imfile :

stwcs.updatewcs.utils.**build_distname**(*sipname*, *npolname*, *d2imname*)
Core function to build DISTNAME keyword value without the HSTWCS input.

CHAPTER

THREE

HEADERLET

The 'headerlet' serves as a mechanism for encapsulating WCS information for a single pointing so that it can be used to update the WCS solution of an image. The concept of a 'headerlet' seeks to provide a solution where only the WCS solution for an image that has been aligned to an astrometric catalog can be archived and retrieved for use in updating copies of that image's WCS information without getting the image data again. Multiple 'headerlets' could even be provided with each representing the alignment of an image to a different astrometric solution, giving the end user the option to get the solution that would allow them to best align their images with external data of interest to them. These benefits can only be realized with the proper definition of a 'headerlet' and the procedures used to define them and apply them to data.

The headerlet object needs to be as compact as possible while providing an unambigious and self-consistent WCS solution for an image while requiring a minimum level of software necessary to apply the headerlet to an image.

3.1 Headerlet File Structure

This new object complete with the NPOLFILE and the D2IMFILE extensions derived from the full FITS file fully describes the WCS of each chip and serves without further modification as the definition of the `headerlet`. The listing of the FITS extensions for a `headerlet` for the sample ACS/WFC exposure after writing it out to a file would then be:

```
EXT#  FITSNAME      FILENAME              EXTVE DIMENS       BITPI OBJECT

0     j8hw27c4q     j8hw27c4q_hdr.fits                       16
1       IMAGE       D2IMARR               1     4096         -32
2       IMAGE       WCSDVARR              1     64x32        -32
3       IMAGE       WCSDVARR              2     64x32        -32
4       IMAGE       WCSDVARR              3     64x32        -32
5       IMAGE       WCSDVARR              4     64x32        -32
6       IMAGE       SIPWCS                1                  8
7       IMAGE       SIPWCS                2                  8
```

This file now fully describes the WCS solution for this image, complete with all the distortion information used to originally define the solution. No further reference files or computations would be needed when this `headerlet` gets used to update an image.

The primary header must have 4 required keywords:

`HDRNAME` - a unique name for the headerlet

`DESTIM` - target image filename (the ROOTNAME keyword of the original archive filename)

`WCSNAME` - the value of WCSNAME<key> copied from the WCS which was used to create the headerlet

SIPNAME - the name of reference file which contained the original distortion model coefficients. A blank value or 'N/A' will indicate no SIP model was provided or applied. A value of 'UNKNOWN' indicates a SIP model of unknown origin.

NPOLFILE - the name of the NPOLFILE, the reference file which contained the original non-polynomial corrections. The same rules used for SIPNAME apply here as well.

D2IMFILE - the name of the D2IMFILE, the reference file which contained the detector to image correction (such as column width correction calibrations). The same rules used for SIPNAME apply here as well.

DISTNAME - a concatenation of SIPNAME, NPOLFILE, and D2IMFILE used as a quick reference for the distortion models included with this headerlet.

UPWCSVER - version of STWCS used to create the WCS of the original image

PYWCSVER - version of PyWCS used to create the WCS of the original image

3.2 User-Interface: headerlet

The headerlet module provides those functions necessary for creating, updating, and applying headerlets to FITS images.

3.2.1 Headerlet - User Interface

This module implements headerlets.

A headerlet serves as a mechanism for encapsulating WCS information which can be used to update the WCS solution of an image. The idea came up first from the desire for passing improved astrometric solutions for HST data and provide those solutions in a manner that would not require getting entirely new images from the archive when only the WCS information has been updated.

class stwcs.wcsutil.headerlet.**FuncNameLoggingFormatter**(*fmt=None*, *datefmt=None*)

format(*record*)

class stwcs.wcsutil.headerlet.**Headerlet**(*fobj*, *mode='copyonwrite'*, *logging=False*, *logmode='w'*)

A Headerlet class Ref: http://mediawiki.stsci.edu/mediawiki/index.php/Telescopedia:Headerlets

Parameters

fobj: string :

Name of headerlet file, file-like object, a list of HDU instances, or an HDUList instance

mode: string, optional :

Mode with which to open the given file object

logging: boolean :

enable file logging

logmode: 'w' or 'a' :

for internal use only, indicates whether the log file should be open in attach or write mode

apply_as_alternate(*fobj*, *attach=True*, *wcskey=None*, *wcsname=None*)

Copy this headerlet as an alternate WCS to fobj

Parameters

fobj: string, HDUList :

science file/HDUList to which the headerlet should be applied

attach: boolean :

flag indicating if the headerlet should be attached as a HeaderletHDU to fobj. If True checks that HDRNAME is unique in the fobj and stops if not.

wcskey: string :

Key value (A-Z, except O) for this alternate WCS If None, the next available key will be used

wcsname: string :

Name to be assigned to this alternate WCS WCSNAME is a required keyword in a Headerlet but this allows the user to change it as desired.

apply_as_primary (*fobj*, *attach=True*, *archive=True*, *force=False*)

Copy this headerlet as a primary WCS to fobj

Parameters

fobj: string, HDUList :

science file to which the headerlet should be applied

attach: boolean :

flag indicating if the headerlet should be attached as a HeaderletHDU to fobj. If True checks that HDRNAME is unique in the fobj and stops if not.

archive: boolean (default is True) :

When the distortion model in the headerlet is the same as the distortion model of the science file, this flag indicates if the primary WCS should be saved as an alternate and a headerlet extension. When the distortion models do not match this flag indicates if the current primary and alternate WCSs should be archived as headerlet extensions and alternate WCS.

force: boolean (default is False) :

When the distortion models of the headerlet and the primary do not match, and archive is False this flag forces an update of the primary

attach_to_file (*fobj*, *archive=False*)

Attach Headerlet as an HeaderletHDU to a science file

Parameters

fobj: string, HDUList :

science file/HDUList to which the headerlet should be applied

archive: string :

Specifies whether or not to update WCSCORR table when attaching

Notes

The algorithm used by this method: - verify headerlet can be applied to this file (based on DESTIM) - verify that HDRNAME is unique for this file - attach as HeaderletHDU to fobj

build_distname (*dest*)

Builds the DISTNAME for dest based on reference file names.

equal_distmodel(*dmodel*)

get_destination_model(*dest*)
Verifies that the headerlet can be applied to the observation

Determines whether or not the file specifies the same distortion model/reference files.

hverify()
Verify the headerlet file is a valid fits file and has the required Primary Header keywords

info(*columns=None*, *pad=2*, *maxwidth=None*, *output=None*, *clobber=True*, *quiet=False*)
Prints a summary of this headerlet The summary includes: HDRNAME WCSNAME DISTNAME SIPNAME NPOLFILE D2IMFILE

Parameters
columns: list :

List of headerlet PRIMARY header keywords to report in summary By default (set to None), it will use the default set of keywords defined as the global list DEFAULT_SUMMARY_COLS

pad: int :

Number of padding spaces to put between printed columns [Default: 2]

maxwidth: int :

Maximum column width(not counting padding) for any column in summary By default (set to None), each column's full width will be used

output: string (optional) :

Name of optional output file to record summary. This filename can contain environment variables. [Default: None]

clobber: bool :

If True, will overwrite any previous output file of same name

quiet: bool :

If True, will NOT report info to STDOUT

summary(*columns=None*)
Returns a summary of this headerlet as a dictionary

The summary includes a summary of the distortion model as :
HDRNAME WCSNAME DISTNAME SIPNAME NPOLFILE D2IMFILE

Parameters
columns: list :

List of headerlet PRIMARY header keywords to report in summary By default(set to None), it will use the default set of keywords defined as the global list DEFAULT_SUMMARY_COLS

Returns
summary: dict :

Dictionary of values for summary

tofile(*fname*, *destim=None*, *hdrname=None*, *clobber=False*)
Write this headerlet to a file

Parameters

fname: string :

file name

destim: string (optional) :

provide a value for DESTIM keyword

hdrname: string (optional) :

provide a value for HDRNAME keyword

clobber: boolean :

a flag which allows to overwrte an existing file

verify_dest(*dest*, *fname*)

verifies that the headerlet can be applied to the observation

DESTIM in the primary header of the headerlet must match ROOTNAME of the science file (or the name of the destination file)

verify_hdrname(*dest*)

Verifies that the headerlet can be applied to the observation

Reports whether or not this file already has a headerlet with this HDRNAME.

class stwcs.wcsutil.headerlet.**HeaderletHDU**(*data=None*, *header=None*, *name=None*, ***kwargs*)

A non-standard extension HDU for encapsulating Headerlets in a file. These HDUs have an extension type of HDRLET and their EXTNAME is derived from the Headerlet's HDRNAME.

The data itself is a FITS file embedded within the HDU data. The file name is derived from the HDRNAME keyword, and should be in the form <HDRNAME>_hdr.fits. If the COMPRESS keyword evaluates to True, the tar file is compressed with gzip compression.

The structure of this HDU is the same as that proposed for the 'FITS' extension type proposed here: http://listmgr.cv.nrao.edu/pipermail/fitsbits/2002-April/thread.html

The Headerlet contained in the HDU's data can be accessed by the headerlet attribute.

classmethod fromheaderlet(*headerlet*, *compress=False*)

Creates a new HeaderletHDU from a given Headerlet object.

Parameters

headerlet : Headerlet

A valid Headerlet object.

compress : bool, optional

Gzip compress the headerlet data.

Returns

hlet : HeaderletHDU

A HeaderletHDU object for the given Headerlet that can be attached as an extension to an existing HDUList.

headerlet

Return the encapsulated headerlet as a Headerlet object.

This is similar to the hdulist property inherited from the FitsHDU class, though the hdulist property returns a normal HDUList object.

stwcs.wcsutil.headerlet.**apply_headerlet_as_alternate**(**args*, ***kw*)

Apply headerlet to a science observation as an alternate WCS

Parameters

filename: string :

File name of science observation whose WCS solution will be updated

hdrlet: string :

Headerlet file

attach: boolean :

flag indicating if the headerlet should be attached as a HeaderletHDU to fobj. If True checks that HDRNAME is unique in the fobj and stops if not.

wcskey: string :

Key value (A-Z, except O) for this alternate WCS If None, the next available key will be used

wcsname: string :

Name to be assigned to this alternate WCS WCSNAME is a required keyword in a Headerlet but this allows the user to change it as desired.

logging: boolean :

enable file logging

logmode: 'a' or 'w' :

stwcs.wcsutil.headerlet.**apply_headerlet_as_primary**(**args*, ***kw*)

Apply headerlet 'hdrfile' to a science observation 'destfile' as the primary WCS

Parameters

filename: string :

File name of science observation whose WCS solution will be updated

hdrlet: string :

Headerlet file

attach: boolean :

True (default): append headerlet to FITS file as a new extension.

archive: boolean :

True (default): before updating, create a headerlet with the WCS old solution.

force: boolean :

If True, this will cause the headerlet to replace the current PRIMARY WCS even if it has a different distortion model. [Default: False]

logging: boolean :

enable file logging

logmode: 'w' or 'a' :

log file open mode

stwcs.wcsutil.headerlet.**archive_as_headerlet**(**args*, ***kw*)

Save a WCS as a headerlet extension and write it out to a file.

This function will create a headerlet, attach it as an extension to the science image (if it has not already been archived) then, optionally, write out the headerlet to a separate headerlet file.

Either wcsname or wcskey must be provided, if both are given, they must match a valid WCS Updates wcscorr if necessary.

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file
An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

hdrname: string :

Unique name for this headerlet, stored as HDRNAME keyword

sciext: string :

name (EXTNAME) of extension that contains WCS to be saved

wcsname: string :

name of WCS to be archived, if " ": stop

wcskey: one of A...Z or " " or "PRIMARY" :

if " " or "PRIMARY" - archive the primary WCS

destim: string :

DESTIM keyword if NOne, use ROOTNAME or science file name

sipname: string or None (default) :

Name of unique file where the polynomial distortion coefficients were read from. If None, the behavior is: The code looks for a keyword 'SIPNAME' in the science header If not found, for HST it defaults to 'IDCTAB' If there is no SIP model the value is 'NOMODEL' If there is a SIP model but no SIPNAME, it is set to 'UNKNOWN'

npolfile: string or None (default) :

Name of a unique file where the non-polynomial distortion was stored. If None: The code looks for 'NPOLFILE' in science header. If 'NPOLFILE' was not found and there is no npol model, it is set to 'NOMODEL' If npol model exists, it is set to 'UNKNOWN'

d2imfile: string :

Name of a unique file where the detector to image correction was stored. If None: The code looks for 'D2IMFILE' in the science header. If 'D2IMFILE' is not found and there is no d2im correction, it is set to 'NOMODEL' If d2im correction exists, but 'D2IMFILE' is missing from science header, it is set to 'UNKNOWN'

author: string :

Name of user who created the headerlet, added as 'AUTHOR' keyword to headerlet PRIMARY header

descrip: string :

Short description of the solution provided by the headerlet This description will be added as the single 'DESCRIP' keyword to the headerlet PRIMARY header

history: filename, string or list of strings :

Long (possibly multi-line) description of the solution provided by the headerlet. These comments will be added as 'HISTORY' cards to the headerlet PRIMARY header If filename is specified, it will format and attach all text from that file as the history.

logging: boolean :

enable file folling

logmode: 'w' or 'a' :

log file open mode

stwcs.wcsutil.headerlet.**attach_headerlet**(**args*, ***kw*)

Attach Headerlet as an HeaderletHDU to a science file

Parameters

filename: string, HDUList :

science file to which the headerlet should be applied

hdrlet: string or Headerlet object :

string representing a headerlet file

logging: boolean :

enable file logging

logmode: 'a' or 'w' :

stwcs.wcsutil.headerlet.**create_headerlet**(**args*, ***kw*)

Create a headerlet from a WCS in a science file If both wcskey and wcsname are given they should match, if not raise an Exception

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

sciext: string or python list (default: 'SCI') :

Extension in which the science data with the linear WCS is. The headerlet will be created from these extensions. If string - a valid EXTNAME is expected If int - specifies an extension with a valid WCS, such as 0 for a simple FITS file If list - a list of FITS extension numbers or strings representing extension tuples, e.g. ('SCI, 1') is expected.

hdrname: string :

value of HDRNAME keyword Takes the value from the HDRNAME<wcskey> keyword, if not available from WCSNAME<wcskey> It stops if neither is found in the science file and a value is not provided

destim: string or None :

name of file this headerlet can be applied to if None, use ROOTNAME keyword

wcskey: char (A...Z) or " " or "PRIMARY" or None :

a char representing an alternate WCS to be used for the headerlet if " ", use the primary (default) if None use wcsname

wcsname: string or None :

if wcskey is None use wcsname specified here to choose an alternate WCS for the headerlet

sipname: string or None (default) :

Name of unique file where the polynomial distortion coefficients were read from. If None, the behavior is: The code looks for a keyword 'SIPNAME' in the science header If not found, for HST it defaults to 'IDCTAB' If there is no SIP model the value is 'NOMODEL' If there is a SIP model but no SIPNAME, it is set to 'UNKNOWN'

npolfile: string or None (default) :

Name of a unique file where the non-polynomial distortion was stored. If None: The code looks for 'NPOLFILE' in science header. If 'NPOLFILE' was not found and there is no npol model, it is set to 'NOMODEL' If npol model exists, it is set to 'UNKNOWN'

d2imfile: string :

Name of a unique file where the detector to image correction was If None: The code looks for 'D2IMFILE' in the science header. If 'D2IMFILE' is not found and there is no d2im correction, it is set to 'NOMODEL' If d2im correction exists, but 'D2IMFILE' is missing from science header, it is set to 'UNKNOWN'

author: string :

Name of user who created the headerlet, added as 'AUTHOR' keyword to headerlet PRIMARY header

descrip: string :

Short description of the solution provided by the headerlet This description will be added as the single 'DESCRIP' keyword to the headerlet PRIMARY header

history: filename, string or list of strings :

Long (possibly multi-line) description of the solution provided by the headerlet. These comments will be added as 'HISTORY' cards to the headerlet PRIMARY header If filename is specified, it will format and attach all text from that file as the history.

nmatch: int (optional) :

Number of sources used in the new solution fit

catalog: string (optional) :

Astrometric catalog used for headerlet solution

logging: boolean :

enable file logging

logmode: 'w' or 'a' :

log file open mode

Returns

Headerlet object :

stwcs.wcsutil.headerlet.**delete_headerlet**(**args*, ***kw*)

Deletes HeaderletHDU(s) from a science file

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file
An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

hdrname: string or None :

HeaderletHDU primary header keyword HDRNAME

hdrext: int, tuple or None :

HeaderletHDU FITS extension number tuple has the form ('HDRLET', 1)

distname: string or None :

distortion model as specified in the DISTNAME keyword

logging: boolean :

enable file logging

logmode: 'a' or 'w' :

Notes

One of hdrname, hdrext or distname should be given. If hdrname is given - delete a HeaderletHDU with a name HDRNAME from fobj. If hdrext is given - delete HeaderletHDU in extension. If distname is given - deletes all HeaderletHDUs with a specific distortion model from fobj. Updates wcscorr

stwcs.wcsutil.headerlet.**extract_headerlet**(**args*, ***kw*)
Finds a headerlet extension in a science file and writes it out as a headerlet FITS file.

If both hdrname and extnum are given they should match, if not raise an Exception

Parameters
filename: string or HDUList or Python list :

This specifies the name(s) of science file(s) from which headerlets will be extracted.

String input formats supported include use of wild-cards, IRAF-style '@'-files (given as '@<filename>') and comma-separated list of names. An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename. If a list of filenames has been specified, it will extract a headerlet from the same extnum from all filenames.

output: string :

Filename or just rootname of output headerlet FITS file If string does not contain '.fits', it will create a filename with '_hlet.fits' suffix

extnum: int :

Extension number which contains the headerlet to be written out

hdrname: string :

Unique name for headerlet, stored as the HDRNAME keyword It stops if a value is not provided and no extnum has been specified

clobber: bool :

If output file already exists, this parameter specifies whether or not to overwrite that file [Default: False]

logging: boolean :

enable logging to a file

stwcs.wcsutil.headerlet.**find_headerlet_HDUs**(**args*, ***kw*)

Returns all HeaderletHDU extensions in a science file that matches the inputs specified by the user. If no hdrext, hdrname or distname are specified, this function will return a list of all HeaderletHDU objects.

Parameters

fobj: string, pyfits.HDUList :

Name of FITS file or open pyfits object (pyfits.HDUList instance)

hdrext: int, tuple or None :

index number(EXTVER) or extension tuple of HeaderletHDU to be returned

hdrname: string :

value of HDRNAME for HeaderletHDU to be returned

distname: string :

value of DISTNAME for HeaderletHDUs to be returned

strict: bool [Default: True] :

Specifies whether or not at least one parameter needs to be provided If False, all extension indices returned if hdrext, hdrname and distname are all None. If True and hdrext, hdrname, and distname are all None, raise an Exception requiring one to be specified.

logging: boolean :

enable logging to a file called headerlet.log

logmode: 'w' or 'a' :

log file open mode

Returns

hdrlets: list :

A list of all matching HeaderletHDU extension indices (could be just one)

stwcs.wcsutil.headerlet.**get_extname_extver_list**(*fobj*, *sciext*)

Create a list of (EXTNAME, EXTVER) tuples

Based on sciext keyword (see docstring for create_headerlet) walk throughh the file and convert extensions in `sciext` to valid (EXTNAME, EXTVER) tuples.

stwcs.wcsutil.headerlet.**get_header_kw_vals**(*hdr*, *kwname*, *kwval*, *default=0*)

stwcs.wcsutil.headerlet.**get_headerlet_kw_names**(*fobj*, *kw='HDRNAME'*)

Returns a list of specified keywords from all HeaderletHDU extensions in a science file.

Parameters

fobj: string, pyfits.HDUList :

kw: str :

Name of keyword to be read and reported

stwcs.wcsutil.headerlet.**headerlet_summary**(*filename*, *columns=None*, *pad=2*, *maxwidth=None*, *output=None*, *clobber=True*, *quiet=False*)

Print a summary of all HeaderletHDUs in a science file to STDOUT, and optionally to a text file The summary includes: HDRLET_ext_number HDRNAME WCSNAME DISTNAME SIPNAME NPOLFILE D2IMFILE

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

columns: list :

List of headerlet PRIMARY header keywords to report in summary By default (set to None), it will use the default set of keywords defined as the global list DEFAULT_SUMMARY_COLS

pad: int :

Number of padding spaces to put between printed columns [Default: 2]

maxwidth: int :

Maximum column width(not counting padding) for any column in summary By default (set to None), each column's full width will be used

output: string (optional) :

Name of optional output file to record summary. This filename can contain environment variables. [Default: None]

clobber: bool :

If True, will overwrite any previous output file of same name

quiet: bool :

If True, will NOT report info to STDOUT

stwcs.wcsutil.headerlet.**init_logging**(*funcname=None*, *level=100*, *mode='w'*, ***kwargs*)

Initialize logging for a function

Parameters

funcname: string :

Name of function which will be recorded in log

level: int, or bool, or string :

int or string : Logging level bool: False - switch off logging Text logging level for the message ("DEBUG", "INFO", "WARNING", "ERROR", "CRITICAL")

mode: 'w' or 'a' :

attach to logfile ('a' or start a new logfile ('w')

stwcs.wcsutil.headerlet.**is_par_blank**(*par*)

stwcs.wcsutil.headerlet.**parse_filename**(*fname*, *mode='readonly'*)

Interprets the input as either a filename of a file that needs to be opened or a PyFITS object.

Parameters

fname: string, pyfits.HDUList :

Input pointing to a file or PyFITS object. An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

mode: string :

Specifies what PyFITS mode to use when opening the file, if it needs to open the file at all [Default: 'readonly']

Returns

fobj: pyfits.HDUList :

PyFITS handle for input file

fname: string :

Name of input file

close_fobj: bool :

Flag specifying whether or not fobj needs to be closed since it was opened by this function. This allows a program to know whether they need to worry about closing the PyFITS object as opposed to letting the higher level interface close the object.

stwcs.wcsutil.headerlet.**print_summary**(*summary_cols*, *summary_dict*, *pad=2*, *maxwidth=None*, *idcol=None*, *output=None*, *clobber=True*, *quiet=False*)

Print out summary dictionary to STDOUT, and possibly an output file

stwcs.wcsutil.headerlet.**restore_all_with_distname**(**args*, ***kw*)

Restores all HeaderletHDUs with a given distortion model as alternate WCSs and a primary

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file
An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

distname: string :

distortion model as represented by a DISTNAME keyword

primary: int or string or None :

HeaderletHDU to be restored as primary if int - a fits extension if string - HDRNAME if None - use first HeaderletHDU

archive: boolean (default True) :

flag indicating if HeaderletHDUs should be created from the primary and alternate WCSs in fname before restoring all matching headerlet extensions

logging: boolean :

enable file logging

logmode: 'a' or 'w' :

stwcs.wcsutil.headerlet.**restore_from_headerlet**(**args*, ***kw*)

Restores a headerlet as a primary WCS

Parameters

filename: string or HDUList :

Either a filename or PyFITS HDUList object for the input science file
An input filename (str) will be expanded as necessary to interpret any environmental variables included in the filename.

hdrname: string :

HDRNAME keyword of HeaderletHDU

hdrext: int or tuple :

Headerlet extension number of tuple ('HDRLET',2)

archive: boolean (default: True) :

When the distortion model in the headerlet is the same as the distortion model of the science file, this flag indicates if the primary WCS should be saved as an alternate nd a headerlet extension. When the distortion models do not match this flag indicates if the current primary and alternate WCSs should be archived as headerlet extensions and alternate WCS.

force: boolean (default:False) :

When the distortion models of the headerlet and the primary do not match, and archive is False, this flag forces an update of the primary.

logging: boolean :

enable file logging

logmode: 'a' or 'w' :

stwcs.wcsutil.headerlet.**update_ref_files**(*source*, *dest*)

Update the reference files name in the primary header of 'dest' using values from 'source'

Parameters

source: pyfits.Header :

dest: pyfits.Header :

stwcs.wcsutil.headerlet.**update_versions**(*sourcehdr*, *desthdr*)

Update keywords which store version numbers

stwcs.wcsutil.headerlet.**verify_hdrname_is_unique**(*fobj*, *hdrname*)

Verifies that no other HeaderletHDU extension has the specified hdrname.

Parameters

fobj: string, pyfits.HDUList :

Name of FITS file or open pyfits object (pyfits.HDUList instance)

hdrname: string :

value of HDRNAME for HeaderletHDU to be compared as unique

Returns

unique: bool :

If True, no other HeaderletHDU has the specified HDRNAME value

stwcs.wcsutil.headerlet.**with_logging**(*func*)

stwcs.wcsutil.headerlet.**write_headerlet**(**args*, ***kw*)

Save a WCS as a headerlet FITS file.

This function will create a headerlet, write out the headerlet to a separate headerlet file, then, optionally, attach it as an extension to the science image (if it has not already been archived)

Either wcsname or wcskey must be provided; if both are given, they must match a valid WCS.

Updates wcscorr if necessary.

Parameters

filename: string or HDUList or Python list :

This specifies the name(s) of science file(s) from which headerlets will be created and written out. String input formats supported include use of wild-cards, IRAF-style '@'-files (given as '@<filename>') and comma-separated list of names. An input filename

(str) will be expanded as necessary to interpret any environmental variables included in the filename.

hdrname: string :

Unique name for this headerlet, stored as HDRNAME keyword

output: string or None :

Filename or just rootname of output headerlet FITS file If string does not contain '.fits', it will create a filename starting with the science filename and ending with '_hlet.fits'. If None, a default filename based on the input filename will be generated for the headerlet FITS filename

sciext: string :

name (EXTNAME) of extension that contains WCS to be saved

wcsname: string :

name of WCS to be archived, if " ": stop

wcskey: one of A...Z or " " or "PRIMARY" :

if " " or "PRIMARY" - archive the primary WCS

destim: string :

DESTIM keyword if NOne, use ROOTNAME or science file name

sipname: string or None (default) :

Name of unique file where the polynomial distortion coefficients were read from. If None, the behavior is: The code looks for a keyword 'SIPNAME' in the science header If not found, for HST it defaults to 'IDCTAB' If there is no SIP model the value is 'NOMODEL' If there is a SIP model but no SIPNAME, it is set to 'UNKNOWN'

npolfile: string or None (default) :

Name of a unique file where the non-polynomial distortion was stored. If None: The code looks for 'NPOLFILE' in science header. If 'NPOLFILE' was not found and there is no npol model, it is set to 'NOMODEL' If npol model exists, it is set to 'UNKNOWN'

d2imfile: string :

Name of a unique file where the detector to image correction was stored. If None: The code looks for 'D2IMFILE' in the science header. If 'D2IMFILE' is not found and there is no d2im correction, it is set to 'NOMODEL' If d2im correction exists, but 'D2IMFILE' is missing from science header, it is set to 'UNKNOWN'

author: string :

Name of user who created the headerlet, added as 'AUTHOR' keyword to headerlet PRIMARY header

descrip: string :

Short description of the solution provided by the headerlet This description will be added as the single 'DESCRIP' keyword to the headerlet PRIMARY header

history: filename, string or list of strings :

Long (possibly multi-line) description of the solution provided by the headerlet. These comments will be added as 'HISTORY' cards to the headerlet PRIMARY header If filename is specified, it will format and attach all text from that file as the history.

attach: bool :

Specify whether or not to attach this headerlet as a new extension It will verify that no other headerlet extension has been created with the same 'hdrname' value.

clobber: bool :

If output file already exists, this parameter specifies whether or not to overwrite that file [Default: False]

logging: boolean :

enable file logging

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

[R1] Jay Anderson, “Variation of the Distortion Solution of the WFC”, ACS ISR 2007-08.

[R4] David Shupe, et al, “The SIP Convention of representing Distortion in FITS Image headers”, Astronomical Data Analysis Software And Systems, ASP Conference Series, Vol. 347, 2005

S

S

R

S

T

U

V

W