AstroData API Documentation

Release X1

Kenneth Anderson

October 21, 2014

CONTENTS

CHAPTER

ONE

ASTRODATA API, METHODS AND PROPERTIES

1.1 Class AstroData

class `astrodata.AstroData.`**`AstroData`**`(`*dataset=None*, *phu=None*, *header=None*, *data=None*, *exts=None*, *extInsts=None*, *store=None*, *mode='readonly'*`)`

The AstroData class abstracts datasets stored in MEF files and provides uniform interfaces for working on datasets from different instruments and modes. Configuration packages are used to describe the specific data characteristics, layout, and to store type-specific implementations.

MEFs can be generalized as lists of header-data units (HDU), with key-value pairs populating headers, and pixel values populating the data array. AstroData interprets a MEF as a single complex entity. The individual "extensions" within the MEF are available using Python list ("[]") syntax; they are wrapped in AstroData objects. (See `AstroData.__getitem__()`). AstroData uses `pyfits` for MEF I/O and `numpy` for pixel manipulations.

While the `pyfits` and `numpy` objects are available to the programmer, `AstroData` provides analogous methods for most `pyfits` functionalities which allows it to maintain the dataset as a cohesive whole. The programmer does however use the `numpy.ndarrays` directly for pixel manipulation.

In order to identify types of dataset and provide type-specific behavior, `AstroData` relies on configuration packages either in the `PYTHONPATH` environment variable or the `Astrodata` package environment variables, `ADCONFIGPATH` and `RECIPEPATH`. A configuration package (eg. `astrodata_Gemini`) contains definitions for all instruments and modes. A configuration package contains type definitions, meta-data functions, information lookup tables, and any other code or information needed to handle specific types of dataset.

This allows `AstroData` to manage access to the dataset for convenience and consistency. For example, `AstroData` is able:

- to allow reduction scripts to have easy access to dataset classification information in a consistent way across all instruments and modes;
- to provide consistent interfaces for obtaining common meta-data across all instruments and modes;
- to relate internal extensions, e.g. discriminate between science and variance arrays and associate them properly;
- to help propagate header-data units important to the given instrument mode, but unknown to general purpose transformations.

In general, the purpose of `AstroData` is to provide smart dataset-oriented interfaces that adapt to dataset type. The primary interfaces are for file handling, dataset-type checking, and managing meta-data, but `AstroData` also integrates other functionalities.

The AstroData constructor constructs an in-memory representation of a dataset. If given a filename it uses pyfits to open the dataset, reads the header and detects applicable types. Binary data, such as pixel data, is left on disk until referenced.

Parameters

- **dataset** (*string, AstroData, HDUList*) – the dataset to load, either a filename (string) path or URL, an 'AstroData' instance, or a 'pyfits.HDUList'
- **phu** (*pyfits.core.Header*) – Primary Header Unit. This object is propagated to all astrodata sub-data ImageHDUs. Special handling is made for header instances that are passed in as this arg., where a phu will be created and the '.header' will be assigned (ex. hdulist[0], ad.phu, ad[0].hdulist[0], ad['SCI',1].hdulist[0], ad[0].phu, ad['SCI',1].phu, and all the previous with .header appended)
- **header** – extension header for images (eg. 'hdulist[1].header', 'ad[0].hdulist[1].header', 'ad['SCI',1].hdulist[1].header')
- **data** (*numpy.ndarray*) – the image pixel array (eg. 'hdulist[1].data', 'ad[0].hdulist[1].data', 'ad['SCI',1].hdulist[1].data')
- **exts** (*list*) – (advanced) a list of extension indexes in the parent 'HDUList' that this instance should refer to, given integer or (EXTNAME, EXTVER) tuples specifying each extension in the pyfits index space where the PHU is at index 0, the first data extension is at index 1, and so on. I.e. This is primarily intended for internal use when creating "sub-data", which are AstroData instances that represent a slice, or subset, of some other AstroData instance.

  NOTE: if present, this option will override and obscure the <extInsts> argument, in other word <extInsts> will be ignored.

  Example of sub-data:

  sci_subdata = ad["SCI"]

  The sub-data is created by passing "SCI" as an argument to the constructor. The 'sci_subdata' object would consist of its own 'AstroData' instance referring to it's own `HDUList`, but the HDUs in this list would still be shared (in memory) with the 'ad' object, and appear in its `HDUList` as well.
- **extInsts** (*list of pyfits.HDU objects*) – (advanced) A list of extensions this instance should contain, specified as actual pyfits.HDU instances. NOTE: if the 'exts' argument is also set, `extInsts` is ignored.
- **store** (*string*) – directory where a copy of the original file will be stored. This is used in the special case where the filename is an URL to a remote fits file. Otherwise it has no effect.
- **mode** (*string*) – IO access mode, same as pyfits mode ("readonly", "update", or "append") with one additional AstroData-specific mode, "new". If the mode is "new", and a filename is provided, the constructor checks that the named file does not exist on disk, and if it does not it creates an empty `AstroData` of that name but does not write it to disk. Such an `AstroData` instance is ready to have HDUs appended, and to be written to disk at the user's command with `ad.write()`.

`__getitem__`(*ext*)

AstroData instances behave as list-like objects and therefore pythonic slicing operations may be performed on instances of this class. This method provides support for list slicing with the "[]" syntax. Slicing is used to create AstroData objects associated with "subdata" of the parent AstroData object, that is, consisting of an HDUList made up of some subset of the parent MEF.

E.g.,

datasetA = AstroData(dataset="datasetMEF.fits") *datasetB = datasetA['SCI']*

datasetC = datasetA[2]

datasetD = datasetA[("SCI",1)]

etc.

In this case, after the operations, datasetB is an `AstroData` object associated with the same MEF, sharing some of the the same actual HDUs in memory as `datasetA`. The object in `datasetB` will behave as if the SCI extensions are its only members, and it does in fact have its own pyfits.HDUList. Note that 'datasetA' and 'datasetB' share the PHU and also the data structures of the HDUs they have in common, so that a change to 'datasetA[('SCI',1)].data' will change the 'datasetB[('SCI',1)].data' member and vice versa. They are in fact both references to the same numpy array in memory. The 'HDUList' is a different list, however, that references common HDUs. If a subdata related 'AstroData' object is written to disk, the resulting MEF will contain only the extensions in the subdata's 'HDUList'.

Note: Integer extensions start at 0 for the data-containing extensions, not at the PHU as with pyfits. This is important: 'ad[0]' is the first content extension, in a traditional MEF perspective, the extension AFTER the PHU; it is not the PHU! In `AstroData` instances, the PHU is purely a header, and not counted as an extension in the way that headers generally are not counted as their own elements in the array they contain meta-data for. The PHU can be accessed via the 'phu' member.

Parameters ext (*<str>, <int>, or <tuple>*) – Integer index, an index tuple (EXTNAME, EXTVER), or EXTNAME name. If an int or tuple, the single extension identified is wrapped with an AstroData instance, and single-extension members of the AstroData object can be used. A string 'EXTNAME' results in all extensions with the given EXTNAME wrapped by the new instance.

Returns AstroData instance associated with the subset of data.

Return type <AstroData>

Raises KeyError, IndexError

`data`

The data property can only be used for single-HDU AstroData instances, such as those returned during iteration. To set the data member, use *ad.data = newdata*, where *newdata* must be a numpy array. To get the data member, use *npdata = ad.data*.

The "data" member returns appropriate HDU's data member(s) specifically for the case in which the AstroData instance has ONE HDU (in addition to the PHU). This allows a single-extension AstroData, such as AstroData generates through iteration, to be used as though it simply is just the one extension. One is dealing with single extension AstroData instances when iterating over the AstroData extensions and when picking out an extension by integer or tuple indexing. Eg.,

for ad in dataset[SCI]:

# ad is a single-HDU index ad.data = newdata

Returns data array associated with the single extension

Return type <ndarray>

Raises Errors.SingleHDUMemberExcept

`filename`

'filename' is monitored so that the mode can be changed from 'readonly' when 'filename' is changed.

`header`

Returns the header member for Single-HDU AstroData instances.

The header property can only be used for single-HDU AstroData instances, such as those returned during iteration. It is a property attribute which uses *get_header(..)* and *set_header(..)* to access the header member with the "=" syntax. To set the header member, use *ad.header = newheader*, where *newheader* must be a pyfits.Header object. To get the header member, use *hduheader = ad.header*.

Returns header

Return type pyfits.Header

Raises Errors.SingleHDUMemberExcept

`headers`

Returns header member(s) for all extension (except PHU).

Returns list of pyfits.Header instances

Return type <list>

`hdulist`

List of header-data units on the instance.

Returns The AstroData's HDUList as returned by pyfits.open()

Return type <pyfits.HDUList>

`append`(*moredata=None*, *data=None*, *header=None*, *extname=None*, *extver=None*, *auto_number=False*, *do_deepcopy=False*)

Appends header-data units (HDUs) to the AstroData instance.

Parameters

- **moredata** (*pyfits.HDU, pyfits.HDUList, or AstroData*) – either an AstroData instance, an HDUList instance, or an HDU instance to add to this AstroData object. When present, data and header arguments will be ignored.
- **data** (*numpy.ndarray*) – 'data' and 'header' are used to construct a new HDU which is then added to the `HDUList` associated to the AstroData instance. The 'data' argument should be set to a valid numpy array. If 'modedata' is not specified, 'data' and 'header' must both be set.
- **header** (*pyfits.Header*) – 'data' and 'header' are used to construct a new HDU which is then added to the 'HDUList' associated to AstroData instance. The 'header' argument should be set to a valid pyfits.Header object.
- **auto_number** (<*bool*>) – auto-increment the extension version, 'EXTVER', to fit file convention
- **extname** (<*str*>) – extension name as set in keyword 'EXTNAME' (eg. 'SCI', 'VAR', 'DQ'). This is used only when 'header' and 'data' are used.
- **extver** (<*int*>) – extension version as set in keyword 'EXTVER'. This is used only when 'header' and 'data' are used.
- **do_deepcopy** (<*bool*>) – deepcopy the input before appending. May be useful when auto_number is True and the input comes from another AD object.

`close`()

Method will close the 'HDUList' on this instance.

`insert`(*index*, *moredata=None*, *data=None*, *header=None*, *extname=None*, *extver=None*, *auto_number=False*, *do_deepcopy=False*)

Insert a header-data unit (HDUs) into the AstroData instance.

Parameters

- **index** (*<int> or <tuple> (EXTNAME,EXTVER)*) – the extension index, either an int or (EXTNAME, EXTVER) pair before which the extension is to be inserted. Note: the first data extension is [0]; cannot insert before the PHU. 'index' is the Astrodata index, where 0 is the 1st extension.
- **moredata** (*pyfits.HDU, pyfits.HDUList, or AstroData*) – An AstroData instance, an HDUList instance, or an HDU instance. When present, data and header will be ignored.
- **data** (*numpy.ndarray*) – 'data' and 'header' are used in conjunction to construct a new HDU which is then added to the HDUList of the AstroData instance. 'data' should be set to a valid numpy array. If 'modedata' is not specified, 'data' and 'header' both must be set.
- **header** (*pyfits.Header*) – 'data' and 'header' are used in conjunction to construct a new HDU which is then added to the HDUList of the instance. The 'header' argument should be set to a valid pyfits.Header object. If 'moredata' is not specified, 'data' and 'header' both must be set.
- **extname** (*<str>*) – extension name (eg. 'SCI', 'VAR', 'DQ')
- **extver** (*<int>*) – extension version (eg. 1, 2, 3)
- **auto_number** (*<bool>*) – auto-increment the extension version, 'EXTVER', to fit file convention. If set to True, this will override the 'extver' and 'extname' arguments settings.
- **do_deepcopy** (*<bool>*) – deepcopy the input before appending. May be useful when auto_number is True and the input comes from another AD object.

`open`(*source*, *mode='readonly'*)

Method wraps a source dataset, which can be in memory as another AstroData or pyfits HDUList, or on disk, given as the string filename.

NOTE: In general, users should not use 'open' directly, but pass the filename to the AstroData constructor. The constructor uses open(..) however. Users should use the constructor, which may perform extra operations.

Parameters

- **source** (*<str> | <AstroData> | <pyfits.HDUList>*) – source contains some reference for the dataset to be opened and associated with this instance. Generally it would be a filename, but can also be an AstroData instance or a pyfits.HDUList instance.
- **mode** (*<str>*) – IO access mode, same as the pyfits open mode, 'readonly, 'update', or 'append'. The mode is passed to pyfits so if it is an illegal mode name, pyfits will be the subsystem reporting the error.

`remove`(*index*, *hdui=False*)

Parameters **index** (*<int>, or <tuple> (EXTNAME,EXTVER)*) – the extension index, either an int or (EXTNAME, EXTVER) pair before which the extension is to be inserted. Note: the first data extension is [0], you cannot insert before the PHU. Index always refers to Astrodata Numbering system, 0 = HDU

`store_original_name`()

Method adds the key 'ORIGNAME' to PHU of an astrodata object containing the filename when object was instantiated (without any directory info, ie. the basename).

If key has all ready been added (ie. has undergone processing where store_original_name was performed before), then the value original filename is just returned. If the key is there, but does not match the original filename of the object, then the original name is returned, NOT the value in the PHU. The value in the PHU can always be found using ad.phu_get_key_value('ORIGNAME').

write (*filename=None*, *clobber=False*, *rename=None*, *prefix=None*, *suffix=None*)

The write method acts similarly to the 'pyfits HDUList.writeto(..)' function if a filename is given, or like 'pyfits.HDUList.update(..)' if no name is given, using whatever the current name is set to. When a name is given, this becomes the new name of the `AstroData` object and will be used on subsequent calls to write for which a filename is not provided. If the `clobber` flag is `False` (the default) then 'write(..)' throws an exception if the file already exists.

Parameters

- **filename** (<*str*>) – name of the file to write to. Optional if the instance already has a filename defined, which might not be the case for new AstroData instances created in memory.
- **clobber** (<*bool*>) – This flag drives if AstroData will overwrite an existing file.
- **rename** (<*bool*>) – This flag allows you to write the AstroData instance to a new filename, but leave the 'current' name in memory.
- **prefix** (<*str*>) – Add a prefix to `filename`.
- **suffix** (<*str*>) – Add a suffix to `filename`.

type (*prune=False*)

Returns a list of type classifications. It is possible to 'prune' the list so that only leaf nodes are returned, which is useful when leaf nodes take precedence such as for descriptors.

Note: types consist of a hierarchical tree of dataset types. This latter tree maps roughly to instrument-modes, with instrument types branching from the general observatory type, (e.g. 'GEMINI').

Currently the distinction betwen status and type is not used by the system (e.g. in type-specific default recipe assignments) and is provided as a service for higher level code, e.g. primitives and scripts which make use of the distinction.

Parameters **prune** (<*bool*>) – flag which controls 'pruning' the returned type list so that only the leaf node type for a given set of related types is returned.

Returns list of classification names

Return type <list> of strings

status (*prune=False*)

Returns the set of 'status' classifications, which are those that tend to change during the reduction of a dataset based on the amount of processing, e.g. RAW vs PREPARED. Strictly, a 'status' type is any type defined in or below the status part of the 'classification' directory within the configuration package. For example, in the Gemini type configuration this means any type definition files in or below the 'astrodata_Gemini/ADCONFIG/classification/status' directory.

Parameters **prune** (<*bool*>) – flag which controls 'pruning' the returned type list so that only the leaf node type for a given set of related status types is returned.

Returns list of classification names

Return type <list> of strings

count_exts (*extname=None*)

The count_exts() function returns the number of extensions matching the passed <extname> (as stored in the HDUs "EXTNAME" header).

Parameters **extname** (<*str*>) – the name of the extension, equivalent to the value associated with the "EXTNAME" key in the extension header.

Returns number of <extname> extensions

Return type <int>

`ext_index` (*extension*, *hduref=False*)
Takes an extension index, either an integer or (EXTNAME, EXTVER) tuple, and returns the index location of the extension. If hduref is set to True, then the index returned is relative to the HDUList (0=PHU, 1=First non-PHU extension). If hduref is False (the default) then the index returned is relative to the AstroData numbering convention, where index=0 is the first extension in the MEF file.

`rename_ext` (*name*, *ver=None*, *force=True*)
The rename_ext(..) function is used in order to rename an HDU with a new EXTNAME and EXTVER identifier. Merely changing the EXTNAME and EXTVER values in the extensions pyfits.Header is not sufficient. Though the values change in the pyfits.Header object, there are special HDU class members which are not updated.

Warning This function manipulates private (or somewhat private) HDU members, specifically 'name' and '_extver'. STSCI has been informed of the issue and has made a special HDU function for performing the renaming. When generally available, this new function will be used instead of manipulating the HDU's properties directly, and this function will call the new pyfits.HDUList(..) function.

Note Works only on single extension instances.

Parameters

- **name** (<*str*>) – New 'EXTNAME' for the given extension.
- **ver** (<*int*>) – New 'EXTVER' for the given extension
- **force** (<*bool*>) – Will update even on subdata, or shared hdulist. Default=True

`info` (*oid=False*, *table=False*, *help=False*)
Prints to stdout information about the phu and extensions found in the current instance.

`get_key_value` (*key*)
The get_key_value() function is used to get the value associated with a given key in the data-header unit of a single-HDU AstroData instance (such as returned by iteration).

Note Single extension AstroData objects are those with only a single header-data unit besides the PHU. They may exist if a single extension file is loaded, but in general are produced by indexing or iteration instructions, Eg.:

sead = ad[("SCI",1)]

for sead in ad["SCI"]: ...

The variable "sead" above is ensured to hold a single extension AstroData object, and can be used more convieniently.

Parameters **key** (<*str*> *header keyword*) – name of header keyword to set

Returns header keyword value

Return type <int>, or <float>, or <str>

Raises SingleHDUMemberExcept

`set_key_value` (*key*, *value*, *comment=None*)
The set_key_value() function is used to set the value (and optionally the comment) associated with a given key in the data-header of a single-HDU AstroData instance. The value argument will be converted to string, so it must have a string operator member function or be passed in as string.

Note Single extension AstroData objects are those with only a single header-data unit besides the PHU. They may exist if a single extension file is loaded, but in general are produced by indexing or iteration instructions.Eg.:

sead = ad[("SCI",1)]

for sead in ad["SCI"]: ...

The variable "sead" above is ensured to hold a single extension AstroData object, and can be used more convieniently.

Parameters

- **key** (*<str>*) – header keyword
- **value** (*<int>, or <float>, or <str>*) – header keyword value
- **comment** (*<str>*) – header keyword comment

`phu_get_key_value`(*key*)

The phu_get_key_value(..) function returns the value associated with the given key within the primary header unit of the dataset. The value is returned as a string (storage format) and must be converted as necessary by the caller.

Parameters **key** (*<str>*) – name of header value to retrieve

Returns keyword value as string or None if not present.

Return type <str>

`phu_set_key_value`(*keyword=None*, *value=None*, *comment=None*)

Add or update a keyword in the PHU of the AstroData object with a specific value and, optionally, a comment

Parameters

- **keyword** (*<str>*) – Name of the keyword to add or update in the PHU
- **value** (*<int>, <float>, or <str>*) – Value of the keyword to add or update in the PHU
- **comment** (*string*) – Comment of the keyword to add or update in the PHU

CHAPTER

TWO

INDICES AND TABLES

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

PYTHON MODULE INDEX

INDEX