Memo: UVCal FITS Format (*.calfits*)

Zaki Ali, Bryna Hazelton, Adam Beardsley,
Paul La Plante, Theodora Kunicki, and the pyuvdata team

July 3, 2017
Revised May 22, 2020

1 Introduction

This memo introduces a new FITS-based file format for storing calibration solutions to use with pyuvdata[1], a python package which provides a software interface for interferometry. We describe the required and optional parameters of a UVCal FITS—hereafter *calfits*—file and pyuvdata's interface for reading and writing these files. For usage examples please refer to the pyuvdata tutorial: http://pyuvdata.readthedocs.io/en/latest/tutorial.html.

2 Overview

Calfits is an adaptation of the FITS file format to enable the storage of calibration information for radio interferometric arrays. Because it builds on the existing FITS file format, all *calfits* files should be properly formatted according to the FITS standard, which is widely available[2].

Any valid *calfits* file corresponds directly to a UVCal object within pyuvdata. As such, every new HDU keyword introduced here has a one-to-one correspondence with UVCal object parameters. In this memo, each new keyword is followed by its corresponding UVCal parameter, in parentheses. For more information about the UVCal class and its parameters, please refer to pyuvdata's documentation: http://pyuvdata.readthedocs.io/en/latest/uvcal.html.

A UVCal object stores calibration solutions as either a "gain"-type solution, or as a "delay"-type solution. These types represent two distinct calibration conventions, both widely used in radio astronomy. Depending on the calibration type (gain or delay), the *calfits* format may consist of up to 4 HDUs. In either case, the primary header is of the

[1]https://github.com/RadioAstronomySoftwareGroup/pyuvdata
[2]https://fits.gsfc.nasa.gov/fits_documentation.html

same basic format and consists of relevant meta-information for a UVCal object to be instantiated. It should be noted that while "gain"-type and "delay"-type *.calfits* files share the same primary header axes, the shape of these are different in each case; for further information, refer to section 3. The second HDU is also the same in either case and is the ANTENNAS HDU. The third HDU, present only in "delay"-type calibrations, stores flags which indicate which frequencies should have delays applied. The fourth HDU is completely optional for both types of calibrations, and stores data pertaining to the overall error of the calibration solution.

3 Primary Header

The primary HDU of a *calfits* file is an image-type HDU with six axes. These axes, in order, represent

1. Data — varies between "delay"- and "gain"-type calibrations as follows:

   - Delay-type: this axis stores the concatenation of one or two UVCal arrays [delay_array, quality_array], in that order. Correspondingly, this axis will be of length 1 or 2, depending on whether the quality_array is present. These arrays both have the shape (Nants_data, 1, Nfreqs, Ntimes, Njones), where these variables take the values stored in the corresponding UVCal object. Note that in a "delay"-type calibration, Nfreqs always = 1.
   - Gain-type: this axis stores the concatenation of the UVCal arrays [real(gain_array), imaginary(gain_array), flag_array, input_flag_array, quality_array], in that order. If there is no input_flag_array or no quality_array for this calibration, they are *not* appended to the axis. Correspondingly, in a "gain"-type calibration, this axis may be of length 3 to 5, depending on the inclusion of the input_flag_array and quality_array. In older files, the quality_array is always present while the input_flag_array was optional. In newer files, the input_flag_array is not present (support for it in UVCal was deprecated as of `pyuvdata` version 3.0) while the quality_array can be present or absent and a boolean header item **HASQLTY** will be present to indicate its presence (to make it clear which array is present if the axis is length 4). So if the axis is length 4 and there is no **HASQLTY** header keyword, the fourth array is the quality_array and there is no input_flag_array. If the axis is length 4 and the **HASQLTY** header keyword is present, then if **HASQLTY** is True, the fourth array is the quality_array, otherwise it is the input_flag_array. These arrays all have the shape (Nants_data, 1, Nfreqs, Ntimes, Njones).

2. An integer representing polarization values.

3. Time.

4. Frequency — in a "delay"-type calibration, this is a placeholder axis of length 1.
5. The spectral window number (currently, uvcal only supports a single spectral window).
6. Antenna number.

The following are required keywords in the primary header of a *calfits* file. For a more detailed explanation of what these keywords mean, see the descriptions on pyuvdata's ReadTheDocs uvcal_parameters page. The uvcal parameter corresponding to each keyword is noted in parentheses.

3.1 Standard FITS Keywords

Some text descriptions in this subsection are adapted from the official FITS 4.0 Standard, which is available at https://fits.gsfc.nasa.gov/fits_standard.html. Only FITS standard keywords which are required by the *calfits* format, and those with corresponding uvcal object parameters will be listed here.

3.1.1 Mandatory standard FITS keywords

- **BITPIX**: *integer* Bits per data value, with sign indicating data type. Possible values and their corresponding data types are:
  - -64: double-precision floating point number
  - -32: single-precision floating point number
  - 8: character or unsigned 8-bit binary integer
  - 16: 16-bit two's complement binary integer
  - 32: 32-bit two's complement binary integer
  - 64: 64-bit two's complement binary integer
- **CTYPEm, CUNITm, CDELTm, CRPIXm, CRVALm**: *string, string, float, float, float* Information about the mth axis. In order, describing coordinate type, coordinate unit, step size (delta) between coordinate values on that axis, coordinate reference pixel, and the coordinate's physical value at that reference pixel. Axes are assumed to be linear.
- **EXTEND**: *boolean* May this FITS file contain extensions? Always *True* in valid *calfits* files.
- **SIMPLE**: *boolean* Does file conform to the Standard? The SIMPLE keyword is required to be the first keyword in the primary header of all FITS files. The value field shall contain a logical constant with the value *True* if the file conforms to the

standard. This keyword is mandatory for the primary header and is not permitted in extension headers. A value of *False* signifies that the file does not conform to this standard.

- **NAXIS:** *integer* Number of axes in the current HDU. A valid *calfits* file always has NAXIS = 6 in its primary HDU.
- **NAXISn**: *integer* The length of the nth axis.
- **TELESCOP**: *string* Observing telescope. Although this keyword is optional in standard FITS files, it is required for *calfits*. (telescope_name)

3.1.2 Optional, but commonly included standard FITS keywords

- **COMMENT**: *string* Descriptive comment. Any number of COMMENT card images may appear in a header.
- **HISTORY**: *string* Processing history of the data. Any number of HISTORY card images may appear in a header.
- **OBSERVER**: *string* The name of the observer. (observer)

3.2 Mandatory *calfits* Keywords

- **CALSTYLE**: *string* Style of calibration. Possible values are “sky” or “redundant”. (cal_style)
- **CALTYPE**: *string* Calibration type parameter. Possible values are “delay” or “gain”. (cal_type)
- **CHWIDTH:** *float* Channel width of of a frequency bin, in units of Hz. (channel_width)
- **GNCONVEN**: *string* Gain convention. The convention for applying the calibration solutions to data. Values are “divide” or “multiply”, indicating whether one should divide or multiply uncalibrated data by gains. Mathematically this indicates the alpha exponent in the equation: (calibrated data) = (gain$^\alpha$) $\times$ (uncalibrated data). A value of “divide” represents $\alpha = -1$ and “multiply” represents $\alpha = 1$. (gain_convention)
- **INTTIME:** *float* Integration time of a time bin, in units of seconds. (integration_time)
- **TMERANGE:** (minimum: *float*, maximum: *float*) Time range (in JD) that cal solutions are valid for. (time_range)

- **XORIENT:** *string* Orientation of the physical dipole corresponding to what is labeled as the x polarization. Possible values are are "east" (indicating east/west orientation) or "north" (indicating north/south orientation). (x_orientation)
- **HASQLTY** *boolean* Indication of whether or not the quality_array is present. Only in newer files, in older files the quality_array was always present, so if this keyword is missing it can be treated as True.

3.2.1 Required if CALSTYLE = "sky"

- **CATALOG**: *string* (Required if CALSTYLE = "sky".) Name of the calibration catalog. (sky_catalog)
- **REFANT**: *string* (Required if CALSTYLE = "sky".) Phase reference antenna. (ref_antenna_name)

3.2.2 Required if CALTYPE = "delay"

- **FRQRANGE:** *float* Required if CALTYPE = "delay". Frequency range that solutions are valid for, in Hz. (freq_range)

3.3 Optional Keywords

- **BL_RANGE:** *float* Range of baselines used for calibration. (baseline_range)
- **FIELD**: *string* A short string describing the field center or dominant source. This used to be required if CALSTYLE = "sky", but it is no longer used (deprecated as of pyuvdata v2.3.3, removed in version 2.5). (sky_field)
- **DIFFUSE:** *string* Name of diffuse model used for sky model. (diffuse_model)
- **GNSCALE:** *string* The gain scale of the calibration, which indicates the units of the calibrated visibilities. For example, Jy or K. (gain_scale)
- **HASHCAL:** *string* Commit hash of calibration software (from ORIGCAL) used to generate solutions. (git_hash_cal)
- **NSOURCES:** *integer* Number of sources used in sky model. (Nsources)
- **ORIGCAL:** *string* Origin (on github for example) of calibration software. URL and branch. (git_origin_cal)

4 Antenna HDU

This HDU is a binary table extension with the extension name "ANTENNA". The Antennas HDU is mandatory in all *calfits* files, and stores detailed information about the calibration solution, per antenna. This binary table has a number of rows equaling the number of antennas in the dataset, and three fields, containing the individual antennas' names, indices, and integer antenna numbers matching the 0th axis of the uvcal object "gain_array."

The three fields of this binary table are:

- **ANTNAME**: List of antenna names, length equal to the number of antennas in the telescope (i.e., the value of NAXIS6). (antenna_names)
- **ANTINDEX**: Array of all integer-valued antenna numbers in the telescope with length equal to the number of antennas in the telescope (i.e., the value of NAXIS6). Ordering of elements matches that of ANTNAME. This array is not necessarily identical to ANTARR, in that this array holds all antenna numbers associated with the telescope, not just those antennas with data, and has an in principle non-specific ordering.(antenna_numbers)
- **ANTARR**: Array of integer antenna numbers that appear in this calibration solution, with a length equal to the calfits parameter Nants_data, which describes the number of antennas with associated gain solutions. (ant_array)

5 Flags HDU (CALTYPE = "delay" only)

This extension is an image HDU with extension name "FLAGS". For "delay"-type calibration solutions, the length of the frequency axis in the primary HDU is set equal to 1 as a placeholder value, and the Flags HDU is mandatory.

This image HDU has the same axes as the primary header, however, the length of the frequency axis is increased to cover all frequencies where delays may be applied. The first axis in the Flags HDU stores a binary flag, which indicates whether or not to apply the delay, as stored in the data axis of the primary HDU.

6 Total Quality HDU

This is an optional extension with extension name "TOTQLTY". For both delay-types, this optional HDU may contain information about the overall χ^2 value of the whole array. The axes of this HDU are the same as those of the primary header, except that it lacks the "antennas" axis. For "delay"-type calibrations, the frequency axis has a length of 1 as above. If this HDU is present, there will be 3 total HDUs for "gain"-type files, and 4 total HDUs for “delay”-type. Note that self-consistency checks are run when reading and writing calfits files to ensure that arrays have the proper size across the various HDUs.