

SOLARNET Metadata Recommendations for Solar Observations

Version 1.4a-pub – 1. December 2020

Stein Vidar Hagfors Haugan and Terje Fredvik

The latest published version of this document can be found at <https://arxiv.org/abs/2011.12139>.
The latest working copy of the document can be found at <http://sdc.uio.no/open/solarnet/>.
Please use the latest version when adding comments/suggested changes (using track changes), before sending to s.v.h.haugan@astro.uio.no.

Preface

This document started as a deliverable in the SOLARNET project but is now a living document that will be used and maintained even after SOLARNET ends. Part A of the original document described various aspects of (ideal) Solar Virtual Observatories. Although that part is still relevant as a rationale for this document's existence, it does not add to the usefulness of the current document and has been removed.

Abstract

Metadata descriptions of Solar observations have so far only been standardized for space-based observations, but the standards have been mostly within a single space mission at a time, at times with significant differences between different mission standards. In the context of ground-based Solar observations, data has typically not been made freely available to the general research community, resulting in an even greater lack of standards for metadata descriptions. This situation makes it difficult to construct multi-instrument archives/virtual observatories with anything more than the most basic metadata available for searching, as well as making it difficult to write generic software for instrument-agnostic data analysis. This document describes the metadata recommendations developed under the SOLARNET EU project, which aims foster more collaboration and data sharing between both ground-based and space-based Solar observatories. The recommendations will be followed by data pipelines developed under the SOLARNET project as well as e.g. the Solar Orbiter SPICE pipeline, the SST CHROMIS/CRISP common pipeline, and the Alma pipeline for Solar data. These recommendations are meant to function as a common reference to which even existing diverse data sets may be related, for ingestion into solar virtual observatories and for analysis by generic software.

Table of Contents

| | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| Part A. Description of FITS keywords | 4 |
| 1. About file formats | 5 |
| 2. Header and Data Units (HDUs) in FITS files | 5 |
| 2.1. Naming of HDUs in SOLARNET FITS files..... | 6 |
| 2.2. Fully and partially SOLARNET-compliant Obs-HDUs | 6 |
| 2.3. Other HDUs | 7 |
| 3. The World Coordinate System (WCS) and related keywords | 7 |
| 3.1. Fundamental WCS coordinate specification..... | 7 |
| 3.2. WCS positional keywords and relative radial velocity | 9 |
| 4. Time-related keywords | 10 |
| 4.1. Specifying WCS time coordinates..... | 10 |
| 5. Description of data contents | 10 |
| 5.1. Data type/units (BTYPE/BUNIT) | 10 |
| 5.2. Exposure time, binning..... | 11 |
| 5.3. Cadence | 11 |
| 5.4. Instrument/data characteristics etc. | 12 |
| 5.5. Quality aspects..... | 13 |
| 5.6. Data statistics | 14 |
| 5.6.1. Missing and saturated pixels, spikes/cosmic rays, padding, etc. | 15 |
| 5.6.2. Explicit listing of missing, saturated, spike/cosmic ray pixels etc..... | 15 |
| 6. Metadata about affiliation, origin, acquisition, etc. | 16 |
| 7. Grouping | 16 |
| 8. Pipeline processing applied to the data | 17 |
| 8.1. Basic description of processing software | 18 |
| 8.2. Detailed description of all processing steps | 19 |
| 9. Integrity and administrative information | 20 |
| 10. Reporting of events detected by the pipeline/spacecraft | 21 |
| 11. References | 44 |
| Part B. Lists of mandatory and optional FITS keywords with example values | 46 |
| 12. Mandatory keyword for all HDUs (Section 4.1) | 46 |
| 13. Mandatory keywords for all Obs-HDUs (Section 4.2) | 46 |
| 14. Mandatory WCS keyword for all HDUs with a UTC (time) coordinate (Section 6.1) | 46 |
| 15. Mandatory keywords for fully SOLARNET-compliant Obs-HDUs | 46 |
| 15.1. Mandatory general keywords (Sections 10 and Appendix V-a) | 46 |
| 15.2. Fundamental WCS coordinate keywords (Section 5.1)..... | 47 |
| 15.3. Mandatory WCS positional keywords (Section 5.2)..... | 47 |
| 15.3.1. Mandatory for ground based observatories (Section 5.2) | 47 |
| 15.3.2. Mandatory for Earth orbiting satellites (Section 5.2) | 47 |
| 15.3.3. Mandatory for deep space missions (not Earth orbiting satellites) (Section 5.2) | 47 |
| 15.4. Mandatory data description keywords (Sections 7.1, 7.2 and 7.6.2)..... | 47 |
| 15.5. Mandatory keywords identifying the origin of the observations (Sections 8 and 9) | 48 |
| 15.6. Mandatory keywords for spectrographs and filter instruments (Sections 5.2 and 7.4) | 48 |
| 15.7. Mandatory keyword for spectrographs (Section 7.4)..... | 48 |
| 15.8. Mandatory keyword for grouping (Sections 9 and Appendix V-b) | 49 |
| 16. Mandatory keyword for SOLARNET HDUs that contain keywords with a definition in conflict with the specifications in this document (Section 4.2) | 49 |
| 17. Mandatory keywords for all HDUs that uses any of the variable-keyword, pixel list or meta-observation mechanism (Sections 4.1, 4.2, 4.3, Appendix I, Appendix II and Appendix III) | 49 |
| 17.1. Mandatory keyword for binary table extension value columns with a coordinate that should not be used in coordinate association (Appendix I-b) | 49 |

17.2. *Mandatory keyword for binary table extension value columns that uses pixel-to-pixel association (Appendix I-d)* 50

18. Optional keywords for all Obs-HDUs 50

18.1. *Optional keywords describing cadence (Section 7.3)*..... 50

18.2. *Optional keywords characterising the instrument/data (Sections 5.1, 7.4 and 7.5)*..... 50

18.3. *Optional quality aspects keywords (Sections 5.1 and 7.5)*..... 50

18.4. *Optional data statistics keywords (Section 7.6)*..... 51

18.5. *Optional keywords for missing and saturated pixels (Section 7.6.1)*..... 51

18.6. *Optional keywords identifying the origin of the observations (Section 8 and Appendix V-b)* 51

18.7. *Optional pipeline processing keywords (Sections 10, 10.1 and 10.2)*..... 51

18.8. *Optional keyword for administrative information (Section 9)*..... 52

18.9. *Optional date and time keywords (Section 4)*..... 52

Part C. Alphabetical listings of FITS keywords with section references..... 53

19. Alphabetical listing of all new SOLARNET keywords with section references 54

20. Alphabetical listing of all keywords with section references..... 55

Part A. Description of FITS keywords

1. About file formats

The most common practice in the solar remote sensing community is currently to use the FITS Standard file format for disseminating solar remote sensing observations. For this reason, this document describes how to include the metadata content through keywords inside FITS files, but *that does not preclude the use of other file formats*. In many ways, this document simply uses FITS notation as a language to express the underlying metadata requirements.

For a discussion about file names and how to group observational data between or inside different files, see Appendix V: [Other recommendations](#).

2. Header and Data Units (HDUs) in FITS files

FITS files may contain one or more Header and Data Units (HDUs) of different types, e.g. primary HDUs, image extensions, and binary table extensions, containing data and a header with metadata stored as keyword-value pairs. Primary HDUs and image extensions are for all practical purposes functionally identical: a primary HDU may simply be regarded as “the first image HDU”.

This document primarily describes how the SOLARNET recommendations apply when using primary and image HDUs to store observational data, but Appendix IV also explains how the recommendations may be applied to observational data stored in binary table extensions. In this context, observational data are data values derived from solar photons recorded by a detector. Other types of data, e.g. temperatures, voltages, atmospheric conditions etc., will be regarded as auxiliary data.

In this document, HDUs storing observational data will be referred to as Obs-HDUs.

There are many keywords that are not mentioned in this document which have an established definition in the FITS Standard, Papers I-V, Thompson (2006), other references, and existing or past projects. As far as possible, such keywords should be used when appropriate, and should not be used in conflict with that definition. New keywords should not be invented if there is already a keyword in established use that covers the needs of the keyword. In general, see the “Other sources of keywords with established use” from the References section before inventing a new keyword.

In the FITS file, keywords should have an appropriate short description in the comment field. Additional comments may be added using the `COMMENT` keyword or by leaving the keyword field blank – see Section 4.4.2.4 in the FITS Standard.

Any units for keyword values should be enclosed in square brackets at the beginning of the keyword comment (Section 4.3 of the FITS Standard).

By default, FITS string values are limited to 68 characters, but the [CONTINUE Long String Keyword Convention](#) (see References) may be used in order to allow keywords to contain strings longer than 68 characters. *Note that this convention must not be used with any of the mandatory or reserved keywords defined in the FITS standard.*

In order to prevent having to specify keyword information that is common to all HDUs in a file, keyword inheritance may be used according to the [FITS Header Inheritance Convention](#) (see References).

This document introduces three new mechanisms that are not part of the FITS standards, but may be useful in fully describing observations: Appendix I explains how to describe keywords that vary as a function of WCS coordinates, Appendix II explains how to pin-point and (optionally) associate values to specific pixels/locations inside a data cube, and Appendix III explains how to signal that an HDU is a part of a larger set of HDUs (e.g. a time series) contained in multiple files.

2.1. Naming of HDUs in SOLARNET FITS files

HDUs in SOLARNET FITS files *must* contain the string-valued keyword `EXTNAME`, and each `EXTNAME` value must be unique within the file. `EXTNAME` must not contain the characters comma or semicolon except as prescribed for the variable-keyword mechanism (Appendix I), the pixel list mechanism (Appendix II) and the meta-observation mechanism ([Appendix III](#)). In addition, `EXTNAME` must not start with a space, but any trailing spaces are ignored. Finally, the CONTINUE Long String Keyword Convention must not be used with `EXTNAME`, since this is a reserved keyword defined in the FITS standard.

Formatted: English (US)

2.2. Fully and partially SOLARNET-compliant Obs-HDUs

All fully SOLARNET-compliant *and* partially SOLARNET-compliant Obs-HDUs *must* contain (in addition to all mandatory FITS standard keywords) the following mandatory keywords:

```
EXTNAME
SOLARNET
OBS_HDU
DATE-BEG
```

Obs-HDUs cannot contain keywords with definitions in conflict with other SOLARNET-defined keywords, unless they occur in a comma-separated list in the keyword `SOLNETEX`. This mechanism may sometimes be necessary to ensure backwards compatibility with existing utilities. Keywords listed in `SOLNETEX` will be ignored by SOLARNET-aware utilities. The `SOLNETEX` mechanism must *not* be applied to FITS standard keywords.

The `SOLARNET` keyword is used to signal if an Obs-HDU is fully SOLARNET-compliant (`SOLARNET=1`) or partially SOLARNET-compliant (`SOLARNET=0.5`).

Fully SOLARNET-compliant Obs-HDUs *must* contain all mandatory SOLARNET/FITS standard keywords described in Section 15 that apply (depending on the nature of the observation), and *must not* have any of those mandatory keywords listed in `SOLNETEX`.

Partially SOLARNET-compliant Obs-HDUs need not contain all keywords summarised in Section 15.

Both fully and partially SOLARNET-compliant Obs_HDUs *must* have `OBS_HDU=1`.

2.3. Other HDUs

Other HDUs in the same file as an Obs-HDU may be used to store additional data that is required to describe the observations, to allow instrument-specific utilities to function correctly, to interpret the data correctly, or to enable further calibrations to be made. Specific cases of such HDUs are variable-keyword HDUs (Appendix I), pixel list HDUs (Appendix II) and meta-HDUs (Appendix III).

Obs-HDUs that are neither fully nor partially SOLARNET-compliant may use the mechanisms described in Appendix I, Appendix II or Appendix III, if they have `SOLARNET=-1`. In fact, the HDUs described in these appendices may themselves use these mechanisms, if they have `SOLARNET=-1`.

3. The World Coordinate System (WCS) and related keywords

The World Coordinate System (WCS) is a very comprehensive standard that should be used for the description of physical data coordinates in Obs-HDUs.

In some earlier data sets, the data coordinates are not specified using the WCS standard, but rather through e.g. `XCEN`, `YCEN`, `FOVX`, and `FOVY`, etc. Future pipelines, however, should *only* use the full, recommended WCS standard, without any deprecated features (e.g. `CROTAi`) or any instrument- or mission-specific practices¹.

All keywords described in this Section are defined by the FITS Standard and Papers I-V. See also Thompson (2006).

3.1. Fundamental WCS coordinate specification

As a reference, the most commonly used conversion from the set of pixel indices ($p_1, p_2, p_3, \dots, p_N$) to a physical WCS coordinate c_i is given by the following formula:

$$c_i(p_1, p_2, p_3, \dots, p_N) = \text{CRVAL}_i + \text{CDELTA}_i \sum_{j=1}^N \text{PC}_{i-j}(p_j - \text{CRPIX}_j)$$

N is normally equal to `NAXIS`, the number of dimensions in the data cube. However, it may be overridden by setting `WCSEXES > NAXIS`, e.g.

- when trailing singular dimensions are being suppressed in the writing of the file, as happens in IDL
- when there are more coordinates that vary throughout the data cube than there are data cube dimensions, e.g. in a raster scan with (x,y,lambda) coordinates, with a time coordinate that is also a function of the x coordinate
- when dummy coordinates are used in table lookup of coordinates, in order to minimise the storage space requirements

`CTYPEi` is used to specify the nature of the coordinates and their projections. In solar observations, the most appropriate values include `HPLN-TAN` and `HPLT-TAN` (solar coordinates;

¹ If a full description seems impossible through the existing WCS framework, please contact s.v.h.haugan@astro.uio.no.

Thompson 2006), `wAVE` (wavelengths in vacuum; Paper III), `UTC` (time; Paper IV), and `STOKES` (Stokes parameter; Paper I).

Coordinates may also be given in table lookup form (Section 6 in Paper III), for use with e.g. Fabry-Pérot imaging spectroscopy with uneven spacing in the wavelength direction. Also, WCS even allows for specifications of distortions down to a pixel-by-pixel level basis if required (see Paper V).

In ground-based observations, image restoration techniques such as MOMFBD leave behind apparent local movements of image features. Such residual effects represent local errors/distortions in the coordinate system specified by the HDU's WCS keywords. In the FITS Standard Section 8.2, it is specified that a "representative average" of such random errors may be given in the keywords `CRDERi` (for axis number i).

Likewise, representative averages for systematic errors in the coordinates may be given in the keywords `CSYERi` (for coordinate number i). Thus, `CSYERi` should be used to represent the uncertainty in the pointing/position of the image as a whole, and uncertainties in the wavelength calibration for spectrometric data.

If a coordinate system has been determined or refined through the use of some external reference image(s) or other source(s), or even been adjusted manually, the keyword `PREFNAME` `COORDREF` should be used to give a comma-separated list of the images/sources/people, see Section 8. If it is not possible to give specific image names/references, the name of the instrument, filter, etc. should be given. Since such images must obviously be (near) co-temporal with the data in the Obs-HDU, this should not introduce much ambiguity.

WCS allows *multiple* sets of coordinate systems to be specified for each data cube (see the FITS Standard). In particular, this can be used to correctly describe data such as rasters, with one system describing the spatial-wavelength coordinate system (x , y , λ), and another describing the temporal-spatial-wavelength coordinate system ($time$, y , λ). Imaging observations scanning through the wavelength dimension could have a primary system describing (x , y , λ) and a second coordinate system (x , y , $time$).

However, in many such cases it is simpler and more appropriate to use a single coordinate system with four coordinates, e.g. (x , y , λ , $time$). This comes naturally if the observations are repeated and concatenated in time (i.e. resulting in a 4-dimensional data cube), but can also be used when scans are stored individually, (i.e. as 3-dimensional data cubes). In such cases, it is necessary to specify the number of coordinates with the keyword `WCSAXES=4` in order to account for the time coordinate that is not represented by a dimension in the data cube.

For rotating FOVs in a time series, the table lookup algorithm for coordinates (see Paper III Section 6) must be used, with a joint table lookup of coordinates `HPLN-TAN`, `HPLT-TAN` and `UTC`. Since table lookup of WCS coordinates is performed with linear interpolation, it is normally possible to represent such a rotating FOV with a coordinate table that has size $(x, y, t) = (2, 2, \tau)$, where τ may be significantly smaller than the number of time steps in the time series. For highly non-linear rotation rates the indexed form of the table lookup algorithm may be used to vary the sampling of the FOV coordinates with time.

For observations (instruments) where the plate scale/pointing is derived from measurements of the apparent solar radius versus the physical size, the keywords `RSUN_REF` should be used to report the reference value for the physical radius used in the calculations (see Thompson 2010, Section 8).

For descriptions of distortions of coordinates in complex data sets, e.g. cavity errors, see Appendix VI.

3.2. WCS positional keywords and relative radial velocity

Ground based observatories must report their geographical location using the keywords `OBSGEO-x`, `OBSGEO-y`, and `OBSGEO-z`, implicitly stating that the observer is following Earth rotation (see Precision effects for solar image coordinates within the FITS world coordinate systems, Section 3; Paper III Section 7). In principle, the coordinates should be given in ITRF geocentric coordinates. However, for SOLARNET purposes, GPS coordinates are an acceptable proxy.

Earth-orbiting satellites must report their position through `GEOX_OBS`, `GEOY_OBS`, and `GEOZ_OBS`. Contrary to the `OBSGEO-x/y/z` keywords, these keywords do *not* implicitly imply that the coordinates are fixed w.r.t. Earth's rotation, but are otherwise identically defined (ITRF, but GPS is an acceptable proxy). For many observations, these keywords must be reported using the variable-keyword mechanism (Appendix I) since the spacecraft might move considerably during the observation.

For deep space missions, the keywords `DSUN_OBS` (distance from Sun centre in metres), `HGLN_OBS` (longitude), and `HGLT_OBS` (latitude) *must* be used to report the instrument position in the Stonyhurst Heliographic system (see Thompson 2006, Sections 2.1 and 9.1). The distance from the Sun centre in astronomical units may be reported in `DSUN_AU` (in addition to `DSUN_OBS`). Note that the Solar B angle is identical to `HGLT_OBS`, and although it is a duplication of information, it may be reported also in `SOLAR_B0` for convenience.

If other coordinate systems or positional information are given for the observer position, they should follow the specifications in Thompson (2006), Sections 2.1 and 9.1.

For spectrometers (and for some narrow-band imagers), the radial velocity between the instrument and the Sun may be important. Unfortunately, WCS does not have a mechanism for specifying this without also correcting the wavelength scale to account for the Doppler shift (see Paper III, Section 7). Such a correction is not traditionally applied in FITS files within the solar physics community. To specify that no such wavelength correction has been done, `SPECSYSa` must be set to 'TOPOCENT' and `VELOSYSa` must be set to 0.0. In order to specify the observer's radial velocity relative to the Sun, the non-WCS keyword `OBS_VR` (given in m/s) must be used (possibly as a variable keyword). Positive velocities are outward from the Sun (i.e. $OBS_VR = dr/dt$).

As mentioned also in Section 2, many keywords already established elsewhere but not mentioned in this document may apply. Such keywords should never be used in conflict with established use. In particular, see "Other sources of keywords with established use" under References. A few that are related to those defined in this section are: `SOLAR_P0` (apparent angle from observer location between celestial north and solar north), `SOLAR_EP` (apparent angle from observer location between celestial north and ecliptic north), `RSUN_ARC` (apparent photospheric solar radius in arc seconds), and `CAR_ROT` (Carrington rotation number for the reference pixel pointed to by `CRPIXj` values).

4. Time-related keywords

`DATE-BEG` *must* be given, referring to the start time of the data acquisition in the time system specified by the `TIMESYS` keyword, which has the default of `'UTC'`. The `TIMESYS` value applies to all `DATE-` keywords, `DATEREF` (see Section 4.1), and several other date-valued keywords.

`DATE-END` may be given, referring to the end of data acquisition.

`DATE-AVG` may be used to give the average date of the observation. However, there is no unambiguous definition of the average when applied to observations with varying cadence or varying exposure times.

Note that we do *not* recommend using the `DATE-OBS` keyword mentioned in the FITS Standard, since this is not explicitly defined there, and has a history of somewhat ambiguous use (see Paper IV).

The observer's position may be important when comparing the times of observations from different vantage points – in particular when at least one of the observations is space based. Thus, the keywords `DSUN_OBS`, `HGLN_OBS`, and `HGLT_OBS` (Section 3.2) may be important w.r.t the timing of the observations.

4.1. Specifying WCS time coordinates

The literature describing all the possible methods of specifying WCS time coordinates is very complex, but except in unusual circumstances, the following prescription should be sufficient:

`CTYPE1='UTC'` should be used as the name of the WCS time coordinate. However, applications should also recognize the value `'TIME'` as having the same meaning, for historical reasons.

Also, `DATEREF` *must* be set to the zero point of the WCS time coordinate. I.e. for pixels that have the `CTYPE1='UTC'` coordinate equal to zero, the time is the value given in `DATEREF`. In most cases the values of `DATEREF` and `DATE-BEG` will be identical, but note that *according to the FITS standard, DATE-BEG is not a default value for DATEREF*, thus `DATEREF` may not be omitted. The existence of both keywords allows e.g. midnight to be used as a zero point for the time coordinate for multiple observations recorded during the following day, each having different values of `DATE-BEG`.

5. Description of data contents

A description of the actual data contents is important for the interpretation of an observation. Such a description is also important for finding relevant observations in an SVO.

5.1. Data type/units (BTYPE/BUNIT)

The `BUNIT` keyword should be used to indicate the units of the values in the data cube. The units should follow the rules in Section 4.3 of the FITS Standard.

It is of course also important that each Obs-HDU has a description of *what* the data array itself represents. For this, the `BTYPE` keyword should be used, even though it is not mentioned in any FITS standard document. It is, however, a natural analogy to the `CTYPE1` keywords used to

indicate the WCS coordinate type. When possible, we recommend using the Unified Content Descriptors (UCD) version 1+ (see References) when specifying `CTYPE`. The keyword comment may also be used to provide additional information.

It may be that the UCD scheme does not cover all data types encountered in solar observation. Thus, it may be necessary for the solar community to decide upon other values for this keyword. This is currently an unresolved issue.

5.2. Exposure time, binning

The exposure time used in the acquisition of an Obs-HDU should be given in the keyword `XPOSURE` - not in `EXPTIME`. The reason why `EXPTIME` should not be used is that in *some cases* it has been used for individual exposure times in summed multi-exposure observations, introducing an ambiguity. According to the recommendation in Paper IV, `XPOSURE` should always contain the *accumulated* exposure time whether or not the data stems from single exposures or summed multiple exposures.

When the data are a result of multiple summed exposures with identical exposure times, the keywords `NSUMEXP` and `TEXPOSUR` can be used to indicate the number of summed exposures and the single-exposure time, respectively.

When the `XPOSURE` or `TEXPOSUR` values vary as a function of time or any other of the Obs-HDU's dimension(s), the variable-keyword mechanism can be used to specify their exact values as a function of those dimensions (see Appendix I for further details). This would typically be the case when Automatic Exposure Control is used - both `XPOSURE` and `TEXPOSUR` could vary as a function of time.

Note that if the data has been binned, the `XPOSURE` keyword should reflect the *physical* exposure time, not the sum of exposure times of the binned pixels. Binning should be specified by the keywords `NBINj`, where j is the dimension number (analogous to the `NAXISj` keywords). E.g. for an observational array with dimensions (x, y, λ, t) where 2x2 binning has been performed in the y and λ directions (as is sometimes done with slit spectrometers), `NBIN2` and `NBIN3` should be set to 2. The default value for `NBINj` is 1, so `NBIN1` and `NBIN4` may be left unspecified.

In order to provide a simple way to determine the combined binning factor (for archive searches), the keyword `NBIN` should be set to the product of all specified `NBINj` keywords.

5.3. Cadence

Cadence may be a very important search term. A meta-Obs-HDU may be used to report such attributes even if it is impossible to do so in the constituent HDUs (Appendix III).

The planned/commanded cadence (frame-to-frame spacing measured in seconds) should be reported in `CADENCE`. The average (actual) cadence should be reported in `CADAVG`.

The cadence *regularity* is also important: The keywords `CADMAX` and `CADMIN` should be set to the maximum and minimum frame-to-frame spacing. `CADVAR` should be set to the variance of the frame-to-frame spacings.

Some instruments take interleaved observation series with a difference in cadence between different filters (“A” and “B”), e.g. AAABAAAB. For such a series, `CADENCE` for the A series should be the planned median spacing between A exposures.

For e.g. on-going synoptic observation series stored with single exposures in separate files (thus separate HDUs) it may be impossible to use the Meta-observation mechanism. The `CADENCE` keyword should be set to the planned series’ cadence. The rest of the keywords should be set based on the available history of the synoptic series.

5.4. Instrument/data characteristics etc.

In order to characterise the spectral range covered by an Obs-HDU, the keywords `WAVEMIN` and `WAVEMAX` should be used to specify the minimum and maximum wavelengths.

The magnitude of the wavelength related keywords mentioned in this section (`WAVExxx`) must be specified in `WAVEUNIT`, given as the power of 10 by which the metre is multiplied, e.g. `WAVEUNIT=-9` for nanometre. We recommend that `WAVEUNIT` corresponds to the `CUNITi` value of the WCS wavelength coordinate, if any, e.g. if `CUNITi='Angstrom'` then `WAVEUNIT=-10`.

`WAVEREF AIRORVAC` should be set to `'air'` or `'vacuum'` to signal whether wavelengths are given for air or vacuum. We recommend that `WAVEREF` corresponds to the `CTYPEi` value of the WCS wavelength coordinate, if any. E.g. if `CTYPEi='AWAV'` then `WAVEREF='air'`.

For spectrometers, the `WAVEMIN/WAVEMAX` values represent the range of wavelengths covered by the Obs-HDU. For filter images, the definition is somewhat up to the discretion of the pipeline constructor, since effective response curves are never a perfect top-hat function. Bear in mind that these two keywords are primarily meant to be used for search purposes. E.g. if someone wants an observation covering a specific wavelength λ , the search can be formulated as `“WAVEMIN < lambda < WAVEMAX”`. In other words, it might be wise to include more than the “intended” or “nominal” min/max wavelengths of a filter: sometimes parts of an extended tail should be included if it covers a potentially interesting emission line that is normally very weak, but may be strong under certain conditions. We suggest that the wavelengths at which the response function is 0.1 times the peak might be a good choice, unless other considerations make other choices more appropriate. This should be based on a measured response function if available – otherwise it should be based on a design specification or theoretical basis. We reiterate, though, that the criteria are up to the discretion of the pipeline designers. The criteria used to set these keywords should in all cases be specified in the keywords’ comment.

For filter images, the `WAVELNTH` keyword may be set to the “characteristic wavelength” at which the observation was taken. For EUV imagers, this keyword typically identifies the most prominent emission line in the bandpass. For a spectrometer `WAVELNTH` might also be the middle of the wavelength range of the HDU, but we leave the exact definition up to the pipeline designers.

In addition, the keyword `WAVEBAND` may be used for a human-readable description of the waveband, typically the (expected) strongest emission/absorption line in HDUs containing spectrometer observations (or specifying the continuum region), or the most dominant contributing line in filter images.

For filter observations where a more thorough specification of the response curve is required for a proper analysis or for search purposes, the variable keyword `RESPONSE` may be used – see Appendix I

The `RESPONSE` keyword should also be used for spectrometers where there are significant variations in the response across the dataset.

If the data has already been corrected for a variable response, the response function that has been applied should instead be given in the variable keyword `RESPAPPL`.

For spectrometric data, the resolving power R should be given in the keyword `RESOLVPW`. For slit spectrometers, the slit width in arc seconds should be given in `SLIT_WID`.

For Stokes vectors, existing conventions use celestial coordinates (RA/DEC) for their reference systems, but for Solar observations this is not practical. SOLARNET-compliant FITS files should use a right-handed reference system (x, y, z) with the z coordinate oriented towards the observer. The image axes are specified by the keyword `POLCONV` in the form “(+/- x , +/- y)” where e.g. `POLCONV=’(+HPLT, -HPLN)’` means that x is parallel to the HPLT axis (Solar North) and y is antiparallel to the HPLN axis.

5.5. Quality aspects

Many quality aspects of ground-based observations change rapidly, even from one exposure to the next. Keywords that describe such quality aspects must therefore often use the variable-keyword mechanism to specify the time evolution of such values, see Appendix I. This mechanism may be used to specify quality-related values for single exposures, average or effective values for composite images, while also allowing an average or effective scalar value to be given in the header.

Until now, there has been little effort in order to characterise quality aspects of ground-based observations in a manner that is *consistent* between different telescopes, and even between different setups at the same telescope. In FITS files from ESO (European Southern Observatory), the keyword `PSF_FWHM` is used to give the full width at half maximum in arc seconds for the point spread function. However, this quantity is generally not available for solar observations. Some adaptive optics systems, however, may record parameters like the atmospheric coherence length r_0 . If available, the value of r_0 should be stored in the keyword `ATMOS_R0`. Since there are multiple ways of measuring this value, its only use should be to reflect the quality of the observing conditions whenever the measurements are performed in the same (or similar enough) way.

If you have suggestions for consistent methods of measuring parameters describing the spatial resolution of observations (or a proxy for it), please contact s.v.h.haugan@astro.uio.no, so that we can include this method in a later version of the document.

The keyword `AO_LOCK` should be used to indicate the status of any adaptive optics. When specified for individual exposures, the value should be either 0 or 1, but as mentioned above, averages may also be specified as appropriate.

The keyword `AO_NMODE` should be used to indicate the number of adaptive optics modes corrected. As mentioned above, averages may also be specified as appropriate. The type of the modes (e.g. Zernike, Karhunen-Loeve, etc.) should be given in the keyword comment.

The keyword `FT_LOCK` is used to indicate the status of any feature tracking `FT_LOCK=0` (no feature tracking lock) or `FT_LOCK=1` (feature tracking lock) for individual exposures, with appropriate averages as mentioned above.

The keyword `ROT_COMP` should be set to 1 if automated solar rotation compensation was in effect during the observation, and to 0 if not. The keyword `ROT_MODL` should be set to specify the rotation model used for rotation compensation². It can refer to specific, predefined models such as `ALLEN` (Allen, *Astrophys. Quantities*, 1979), `HOWARD` (Howard *et al.*), `SIDEREAL`, `SYNODIC`, `CARRINGTON`, or `aaa.a` – arcseconds per hour (units [arcsec/h]). See also the SolarSoft routine `diff_rot.pro`. If other models have been used, please contact s.v.h.haugan@astro.uio.no, or set `ROT_MODL` to `FORMULA`, and specify the formula in the keyword `ROT_FORM`.

If other relevant values to specify e.g. the rotation compensation implementation, we recommend using keywords starting with "ROT_".

`ELEV_ANG`: This keyword should be used to quote the telescope's elevation angle at the time of data acquisition, in degrees.

In some cases, lossy compression has been applied to the data. Depending on the type of compression, different quality aspects will be introduced that should somehow be specified. Since any significant on-board processing should be considered as a processing step in the pipeline, lossy compression may be listed using the `PRxxxxx` keywords described in Section 8.

However, for searching and sorting purposes it would be useful to have a generic numeric keyword describing the loss of quality due to lossy compression. The keyword `COMPQUAL` should be set to a number between 0.0 and 1.0, where 1.0 indicates lossless compression (if any) and 0.0 indicates "all information is lost". In practice, however, the actual value is not crucial, as long as a higher value corresponds to a higher data quality. If there is a choice between different compression algorithms for this instrument, the name of the algorithm should be given in `COMP_ALG`.

`OBS_LOG`: Location of the log file that is relevant to this observation, if available, given as a URL.

`COMMENT`: May be used to include the relevant parts of the `OBS_LOG`, and any other relevant comments about the HDU that may be useful for the interpretation of the data.

5.6. Data statistics

It may be useful to have statistics about the data array of a Obs-HDU in order to search for "particularly interesting" files (or to filter out particularly *uninteresting* files for that matter).

`DATAMIN` – the minimum data value
`DATAMAX` – the maximum data value
`DATAMEAN` – the average data value
`DATAMEDN` – the median data value.

² This might be important when comparing observations where cross-correlation cannot be used for alignment - e.g. coronal observations vs. photospheric observations. In such cases, different rotation models might cause a drift between the two. The information in this keyword can be used to prevent misunderstandings and misinterpretations in such situations.

DATA P_{nn} – the nn percentile (where nn is e.g. 01, 02, 05, 10, 25, 75, 90, 95, 98, and 99).

DATAN P_{nn} – $DATA_{P_{nn}}/avg(x)$

DATARMS – the RMS deviation from the mean $\sqrt{\text{sum}((x-avg(x))^2)/N }$

DATANRMS – $DATARMS/avg(x)$

DATAMAD – the mean absolute deviation, $avg(abs(x-avg(x)))$

DATANMAD – $DATAMAD/avg(x)$

DATAKURT – the kurtosis

DATASKEW – the skewness

Note that no keyword is defined for the standard deviation, since this value is almost indistinguishable from the root mean square for solar observation data sets, and its value can be calculated using **DATARMS** and **NDATAPIX**.

Note that the calculation of these keywords should be based only on pixels containing actual observational data – not including e.g. padding due to rectification, etc.

5.6.1. *Missing and saturated pixels, spikes/cosmic rays, padding, etc.*

In some data sets, the data in the HDU may be affected by missing/lost telemetry, acquisition system glitches, cosmic rays/noise spikes, or saturation, hot/cold pixels etc. Some keywords are useful to find/exclude files based on how many such pixels there are. In order to allow such searches, the following keywords should be used:

NTOTPIX – the number of *expected usable data pixels*, not including e.g. padding.

NLOSTPIX – the number of lost pixels due to telemetry/acquisition glitches

NSATPIX – the number of saturated pixels

NSPIKPIX – the number of identified spike pixels

NMASKPIX – the number of dust-affected/hot/cold/*padded* pixels etc.

NDATAPIX – the number of usable pixels: $NTOTPIX - NLOSTPIX - NSATPIX - NSPIKPIX - NMASKPIX$

Corresponding percentages relative to **NTOTPIX** should be given in **PCT_LOST**, **PCT_SAT**, **PCT_SPIK**, **PCT_MASK**, and **PCT_DATA**.

5.6.2. *Explicit listing of missing, saturated, spike/cosmic ray pixels etc.*

Bad pixels may be handled in one of three ways: they can be left untouched, they can be filled with the value of **BLANK** (integer-valued HDUs) or *NaN* (floating-point-valued HDUs), or they can be filled in with estimated values.

For some purposes, it may be useful to keep lists of such pixels using the pixel list mechanism, see Appendix II. This is especially important when the pixels have been filled in with estimated values, storing the original values in the pixel list. Pixel lists that flag individual lost, saturated, spike or masked pixels, should have **EXTNAME_s** equal to **LOSTPIXLIST**, **SATPIXLIST**, **SPIKPIXLIST**, or **MASKPIXLIST** respectively. Original values (when appropriate) should be given in the pixel list's attribute column with **TTYPEN='ORIGINAL'** – see Appendix II. for details. For cosmic ray/spike detection, a confidence level (between 0.0 and 1.0) may also be given in an attribute column with **TTYPEN='CONFIDENCE'**. In order to ensure unique **EXTNAME_s** for pixel lists belonging to different Obs-HDUs, the pixel list **EXTNAME_s** may have a trailing “tag”, see Appendix II. Pixel lists with other **EXTNAME_s** than **LOSTPIXLIST** etc. may of course be used for other purposes, e.g. storing the pixel indices and classification of sun spots, the latter stored as a string valued attribute.

6. Metadata about affiliation, origin, acquisition, etc.

The keywords in this section describe metadata regarding the origin, acquisition, and affiliation of the data. Although not generally required for the *use* of the data, such metadata are very useful w.r.t. e.g. searching, grouping, counting, and reporting. Some of the keywords will not make sense for all data sets, because the nature and nomenclature of the observational scenarios vary. In such cases, leave them out. Also, some of the keywords will have different meanings within different settings, in many cases based on tradition.

We therefore refrain from giving explicit instructions on the usage of many of the keywords. An SVO should allow searching on such keywords by asking for “observations where `PROJECT=xxx`”, but it should also be possible to search for “observations where `xxx` occurs in any of the keywords mentioned below”.

In general, all keywords below may contain comma-separated lists when necessary. In some cases, it may be a good idea to use both the full name and an acronym.

We *strongly* recommend that all such “free-text” keywords are filled in from lists of predefined texts, strictly controlled by each individual pipeline/instrument team. Experience has shown that free-text fields will be filled in incredibly inconsistently, even the writer’s own name. Of course, it would be even better if a community-wide service could be established to homogenise such controlled lists, but this may never happen.

PROJECT: Name(s) of the project(s) affiliated with the data
MISSION: Typically used only in space-based settings (e.g. the SOHO or STEREO mission)
OBSRVTRY: Name of the observatory
TELESCOP: Name of the telescope.
TELCONF: Telescope configuration.
INSTRUME: Name of the instrument.
CAMERA: Name of the camera.
GRATING: Name of the grating used (when there are more than one available).
FILTER: Name(s) of the filter(s) used during the observation.
DETECTOR: Name of the detector.
OBS_MODE: A string (from a limited/discrete list) uniquely identifying the mode of operation.
SETTINGS: Other settings – numerical values can be given as “parameter1=n, parameter2=m”.
OBSERVER: Who acquired the data.
PLANNER: Observation planner(s).
REQUESTR: Who requested this particular observation.
AUTHOR: Who designed the observation
CAMPAIGN: Coordinated campaign name/number, including instance number, when applicable.

Note also this catch-all keyword:

DATATAGS: Used for any additional search terms that do not fit in any of the above keywords.

7. Grouping

It is very important for an SVO to be able to group search results in a meaningful way.

E.g. if a search matches 1000 Obs-HDUs, but they are part of only 5 different observation series, it makes sense to have a grouping mechanism to collapse the result listing into only 5 lines, showing some form of summary of the underlying Obs-HDUs for each series.

For this to work, the concept of a “pointing id” has proven to be useful in e.g. the Hinode archive – it serves to separate otherwise identical observations into groups in a logical way. This is particularly useful wherever multiple instruments share the same pointing platform/telescope, but also during coordinated campaigns between different telescopes.

We therefore introduce the keyword `POINT_ID`, to be given a new, unique string value (e.g. a string giving the date and time of the repointing) every time the telescope is “*significantly repointed*” - not counting feature tracking or rotation compensation. In other words, successive files “interrupted” by sudden (not continuous) solar rotation compensation/feature tracking jumps may share the same `POINT_ID`. HDUs sharing the same `POINT_ID` *may* have quite different fields of view, e.g. a series of overview images of a sunspot and a simultaneous series with a much smaller field of view focussing on the edge of the sunspot. The exact criteria used for changing the `POINT_ID` value are left up to the pipeline designers/observers, but we would like to stress the importance of this particular keyword for VSO/archive purposes.

For fixed-pointing instruments this concept may of course not be relevant, but if there is any logical grouping of observation series, unique values of the `POINT_ID` keyword should be used to separate them.

8. Pipeline processing applied to the data

The concept of “data level” is often used to label data with a particular degree of processing, from raw data up to complex data products.

However, definitions of data levels are extremely instrument-/mission-/pipeline-dependent, and not very useful in terms of explaining what processing has been applied. This concept is useful, however, to ensure that files with different processing level have unique file names. For this reason, the keyword `LEVEL` may be used, as a string value to capture sub-levels such as quick-look versions.

`VERSION` should be set to the processing version of the file, an integer that should be increased whenever a reprocessing is performed in order to improve the data set (e.g. with a better flat-field, better detection of cosmic rays, [more complete telemetry](#), etc). The version numbers in files published through an SVO may increase by more than one for each new published “generation”, allowing the use of intermediate values for internal/experimental use.

`ORIGIN` should be set to a character string identifying the organization or institution responsible for creating the FITS file. `DATE` should be set to the date of creation of the FITS file.

In addition to the `LEVEL`, `VERSION` and `ORIGIN` keywords, we recommend that some additional keywords are used in order to indicate the processing steps that has been applied to the data. The four keywords described in Section 8.1 may be used instead of or in addition to the more complex set of keywords described in Section 8.2.

8.1. Basic description of processing software

The name and version of the processing software should be specified by those of the following keywords that might apply:

```
CREATOR = 'ZUN_MOMF'           / Name of software that produced the FITS file
VERS_SW = '2.5'                / Version of software applied
VERS_CAL= '2.4'                / Version of calibration pack applied
```

In addition, `PRSTEPn` should specify the nature of the processing steps, if any, that has been applied to the data. Each `PRSTEPn` may contain a comma separated list if multiple processing steps are inseparable. The number `n` specifies the step number and should reflect the order in which the steps have been performed, e.g.:

```
PRSTEP1 = 'FIXED-PATTERN,FLATFIELDING' / First processing steps (inseparable)
PRSTEP2 = 'CALIBRATION'                / Second processing step
PRSTEP3 = 'DISTORTION-CORRECTION'     / Third processing step
```

Below is a list of recommendations for descriptions of processing steps. If desirable, further specifications may be added, e.g. instead of “`LINE-FITTING`” one may want to use “`GAUSSIAN-LINE-FITTING`” versus “`VOIGT-LINE-FITTING`”. Note that distortion corrections come in two flavours: applied to the data (regridding) or applied to the coordinates. In the latter case, `COORDINATE` should be a part of the processing step description. If you need to add to this list, please contact s.v.h.haugan@astro.uio.no.

```
BIAS-CORRECTION
DARK-SUBTRACTION
FLATFIELDING
FIXED-PATTERN-REMOVAL
PIXEL-FILLING
SPATIAL-DISTORTION-CORRECTION
SPECTRAL-DISTORTION-CORRECTION
SPATIAL-COORDINATE-DISTORTION-CORRECTION
SPECTRAL-COORDINATE-DISTORTION-CORRECTION
BINNING
CALIBRATION
DESPIKING
SUMMING
SPECKLE-DECONVOLUTION
WFS-DECONVOLUTION
SHACK-HARTMANN-DECONVOLUTION
MOMFED
SUBTRACTION
MULTIPLICATION
FILTERING
EDGE-DETECTION
THRESHOLDING
BINARIZATION
FLOOR
CELL
ROUND
DEMODULATION
DEROTATION
INVERTING
STOKES-INVERSION
ATMOSPHERIC-INVERSION
DESTRETCHING
LINE-FITTING
COMPRESSION
```

CONCATENATION
 POINTING
 ALIGNMENT
 SPATIAL-ALIGNMENT
 SPECTRAL-ALIGNMENT
 CALIBRATION-PREPARATION
 RECIPROCAL
 ATMOSPHERIC-INVERSION
 STOKES-INVERSION
 STOKES-CROSSTALK-CORRECTION
 DATA-CURATION
 SMOOTHING
 BLURRING
 CROPPING
 BZERO-BSCALE-TRUNCATION
 PADDING-CONVERSION (Lofdahl)

8.2. Detailed description of all processing steps

Each processing step may be described in further detail using all or some of the keywords `PRPROCn`, `PRVERn`, `PRMODEn`, `PRPARAn`, `PRLIBna`, `PRVERna` and `PRBRAna`. The letter `a` stands for an optional character (blank or `A-Z`), used to distinguish between multiple libraries used for step `n`. E.g.:

```

PRSTEP1 = 'MOMFBD'           / Processing step type
PRPROC1 = 'ZUN_MOMF'        / Name of procedure performing PRSTEP1
PRPVER1 =                   0.8 / Version of procedure PRPROC1
PRMODE1 = 'BALANCED'       / Processing mode
PRREF1 = 'balanced_definition_573.sav' / Balanced mode definition file
PRREF1A = 'John Doe (john.doe@email.pr)' / Person doing manual adjustments
PRPARA1 = 'ITER=5,FG=7,FILL=1' / List of parameters/options for PRPROC1
PRLIB1 = 'ZUNRED '         / Software library containing ZUN_MOMF
PRVER1 =                   1.5 / Library version/MJD of last update
PRBRA1 = 'Master '        / GIT repository branch
PRLIB1A = 'SSW '          / Additional software library
PRVER1A =                   59214 / Additional library version/MJD of last update
  
```

In this example, the `ZUN_MOMF` routine is part of the `ZUNRED` reduction package and relies on SolarSoft library routines. If further libraries had been used in processing step 1, they would be specified in `PRLIB1B`, etc. Libraries should be listed in the order they appear in the path. If it is necessary to specify the repository branch in order to identify the software used, `PRBRAna` may be used.

If a single procedure performs multiple steps, it is ok to list each step separately, using the same value in e.g. `PRPROC1` and `PRPROC2`, but different values for `PRSTEP1` and `PRSTEP2`.

The version keywords `PRPVERn`/`PRVERna` should be numerically increasing with increasing maturity of the pipeline. When using libraries with no (numeric) version numbers, the Modified Julian Day (MJD) of the time the library was last mirrored/changed could be used as a version number.

`PRMODEn` is meant for pipelines that may be run with different trade-offs between e.g. signal to noise ratio versus spatial resolution or contrast. This should already be apparent from the other keywords, but `PRMODEn` provides a much simpler way of identifying data processed in a particular way (e.g. “BALANCED” or “HIGH CONTRAST”). Note that a single observation may be

registered multiple times in an SVO with different values of `PRMODEn` - but then a `PRMODEn`-specific identifier in the file name is necessary in order to ensure uniqueness.

`PRREFna` is a catch-all keyword that can be used to refer to up to 26 types of other factors/inputs influencing a processing step, e.g. references to images used for pointing adjustments. If a process has been performed manually or been influenced by one or more persons, this keyword can be used to give the name(s). If the step involves multiple inputs of the same type (e.g. concatenation of multiple files), they can be entered in a single instance of the keyword (e.g. `PRREF1A`) using a comma separated list, whereas multiple types of inputs should use different letters `a`, e.g. the name of a person in `PRREF1A` and the name of a reference image in `PRREF1B`. Listing hundreds or thousands of files is not very practical and may be skipped. However, [glob-like](#) patterns are allowed, e.g. `inputs_*.dat` and `inputs_[2400-3400].dat`.

For some data sets, it may be desirable to include information about how the calibration data has been created/processed. In such cases, the same mechanism should be used, even though the observational data in the HDU is not altered by that processing in itself. The processing steps for the calibration data should have a lower `n` than those steps that use the calibration data (e.g. `PRSTEP1='CALIBRATION-PREPARATION'` and `PRSTEP2='CALIBRATION'`).

9. Integrity and administrative information

The `DATASUM` and `CHECKSUM` keywords (see the [Checksum Keyword Convention](#)) should be set in all HDUs to allow a check on whether the data file has been modified from the original or has become corrupted. However, their values in a meta-HDU (see Appendix III) will be recomputed when constituent HDUs have been combined into a single HDU (after checking the constituent HDUs `DATASUM` and `CHECKSUM`).

`INFO_URL` should point to a human-readable web page describing “everything” about the data set: what it is, how to use it, links to e.g. user guides, instrument/site/telescope descriptions, descriptions of caveats, information about data rights, preferred acknowledgements, whom to contact if you have questions, and repositories of observing/engineering logs.

Upon ingestion of (meta)data into an SVO, the material pointed to by `INFO_URL` and `OBS_LOG` (Section 5.5) might be “harvested” and preserved in such a way that it is possible to retrieve a copy even if the original source is no longer available. It might be possible for an SVO to recursively harvest pages/documents and even auxiliary data such as flat-fields being linked to from `INFO_URL`. The harvesting will have to be restricted somehow - presumably limited to links pointing beside or below `INFO_URL`³ and `OBS_LOG`.

Any other administrative information pertaining to the file should also be included at the `INFO_URL`.

Proprietary data should be marked by setting the keyword `RELEASE` to the date at which the data can be freely distributed. The keyword `RELEASEC` may be used to give contact information for one or more people (name/email addresses, comma separated) administering the release details.

³ E.g. with `INFO_URL='http://some.site/this/guide.html'`, documents `http://some.site/this/manual.pdf` and `http://some.site/this/subdirectory/auxiliary.dat` might be harvested if it is (recursively) referenced from `guide.html`, but not `http://some.site/other/use.pdf`.

10. Reporting of events detected by the pipeline/spacecraft

If the pipeline uses event/feature detection algorithms that will only work on the raw data, not the final pipeline product, detected events/features should be reported in pixel lists (see Appendix II). If possible, events that are detected during acquisition of the data but are not detectable in the acquired data should also be reported (e.g. on-board-detected events in spacecraft).

When possible, such events/features should also be reported to relevant registries following the appropriate standards (e.g. VOEvents).

Appendix I. Variable-keyword mechanism

In many cases, auxiliary data such as detector temperatures, atmospheric conditions, variable exposure times, or adaptive optics performance is recorded alongside the observations. In some cases, other kinds of information such as the instrument response as a function of wavelength or a collection of instrument temperatures may be significant for correct interpretation of the data. In these cases, the variable-keyword mechanism described below can be used to link the observational data and the auxiliary data together.

Since this mechanism may be used by any HDU with a non-zero `SOLARNET` keyword, we will from now on simply use the term “referring HDU” for an HDU that uses this mechanism. The actual values of a variable keyword are stored in a binary table column (see Cotton et al. 1995), which we will call “value columns” in the description below.

To use this mechanism, the referring HDU must contain the keyword `VAR_KEYS` declaring the `EXTNAME` of the binary table extension containing the variable values (i.e. the value columns), followed by a semicolon, and then a comma-separated list of the variable keyword(s).

When multiple extensions are used for storing value columns, this is signalled by a comma behind the last keyword of one extension, then a new `EXTNAME` followed by a semicolon, then a comma-separated list of keywords stored in that extension. The `EXTNAME` is freely chosen as long as it adheres to the `EXTNAME` rules given in Section 2.

Each keyword name may be followed by a “tag” – an expression of the form “[...]”, i.e. a square bracket containing a string. The tag string itself may not contain semicolons, commas, or square brackets. The tag’s only function is to distinguish between different value columns containing values for the same keyword but for different referring HDUs. However, multiple HDUs may refer to a single value column, even if it has a tag.

The value columns must have `TTYPEN` equal to the keyword name plus any tag. Column numbers (`n`) do not matter in the linking of value columns to keyword names. Note that the CONTINUE Long String Keyword Convention must not be used with `TTYPEN`, since this is a reserved keyword defined in the FITS standard.

When appropriate, it is highly recommended that the referring HDU also contains a representative scalar value of a variable keyword, although this is not mandatory. How the representative value is chosen depends on the nature of the variable keyword, though the average of the variable values is usually the appropriate choice. Variable keywords may also have string values.

As an example, the header of a referring HDU might contain the following entries:

```
EXTNAME = 'He_I' / Referring HDU extension name
VAR_KEYS= 'VAR-EXT-1;KEYWD_1,KEYWD_2[He_I],VAR-EXT-2;KEYWD_3' / Variable keywords
KEYWD_1 = 5.2 / Average of KEYWD_1 values
          / No scalar value for KEYWD_2
KEYWD_3 = 5 / Minimum of KEYWD_3 values
```

This means that the values of the variable keywords `KEYWD_1` and `KEYWD_2` are stored in two separate columns in the `VAR-EXT-1` binary table extension, and that the `KEYWD_3` values are stored in the `VAR-EXT-2` binary table extension. The “tag” `[He_I]` is used to distinguish between columns in the `VAR-EXT-1` extension storing different `KEYWD_2` values for different referring HDUs.

The line with a “blank keyword” used to comment on `KEYWD_2` is a valid FITS construct, with the same effect as `COMMENT`. The `VAR-EXT-1` binary table header might contain the following entries (header examples from binary tables are shown in grey in this appendix):

```
EXTNAME = 'VAR-EXT-1'           / Variable keyword binary table extension name
TTYPE5  = 'KEYWD_1'           / Column 5 contains variable KEYWD_1 values
TTYPE6  = 'KEYWD_2[He_I]'     / Column 6 contains variable KEYWD_2 values for He_I
:
TTYPE8  = 'KEYWD_2[C_II]'     / Column 8 contains variable KEYWD_2 values for C_II
```

The `TTYPES` entry is included only to illustrate the need for the `[He_I]` tag in `TTYPE6`. The `VAR-EXT-2` binary table extension might contain the following entries:

```
EXTNAME = 'VAR-EXT-2'           / Variable keyword binary table extension name
TTYPE1  = 'KEYWD_3'           / Column 1 contains variable KEYWD_3 values
```

There are two ways in which the values of the variable keyword data cube may be associated with the data cube in the referring HDU: association by coordinates (Appendix I-a) and pixel-to-pixel association (Appendix I-d).

This mechanism may also be used to store a set of values that do not vary as a function of any coordinate or dimension of the referring HDU. Such constant, multi-valued keywords are described in Appendix I-c.

In all the examples below, the referring HDU is an image sequence with coordinates and dimensions $(x, y, t) = (\text{HPLN-TAN}, \text{HPLT-TAN}, \text{UTC}) = (512, 512, 60)$, with a header containing the following entries relevant to the examples in this appendix (note the formatting of `VAR_KEYS` for readability):

```
DATEREF = '2018-01-01T00:00:00' / Time coord. zero point (time reference, mandatory)
DATE-BEG= '2018-01-01T12:12:12' / Beginning of data acquisition (mandatory)
DATE-END= '2018-01-01T12:43:29' / End of data acquisition
CTYPE1  = 'HPLN-TAN'           / Coord. 1 is "solar x"
CTYPE2  = 'HPLT-TAN'           / Coord. 2 is "solar y"
CTYPE3  = 'UTC'                / Coord. 3 is time in seconds relative to DATEREF
NAXIS1  =                      512 / Size of dimension 1
NAXIS2  =                      512 / Size of dimension 2
NAXIS3  =                      60 / Size of dimension 3
:
VAR_KEYS= 'MEASUREMENTS;'      &' / Extension containing measured auxiliary values
CONTINUE '  ATMOS_R0,'         &' / ATMOS_R0 values
CONTINUE '  TEMPS,'           &' / Temperature history
CONTINUE 'PARAMETERS;'       &' / Extension containing instrument parameters
CONTINUE '  GAINS'            / Gain settings
```

Appendix I-a. Association by coordinates

The variable-keyword mechanism using association by coordinates is fully analogous to the matching up of two separate observations – it is their shared coordinates that describe how to align the two in space, time, wavelength etc. In general, two Obs-HDUs do not necessarily have all coordinates in common. Examples are images vs. spectral rasters, or polarimetric data vs. images. Of course, the order of the WCS coordinates in the two Obs-HDUs does not matter, and e.g. the spatial, temporal, and spectral sampling of the observations may be entirely different, and the coordinates may even be irregular.

Likewise, when using association by coordinates for variable keywords, each value column has its own set of WCS keywords defining their WCS coordinates. These coordinates specify where each value in the value column data cube is located in relation to the referring HDU's WCS coordinates.

As is the case for the alignment of e.g. images vs. spectra, the value columns do not need to specify all of the coordinates in the referring HDU (e.g. a time series of temperatures vs. a sequence of images), and may have coordinates that are not present in the referring HDU (e.g. a time series of temperatures vs. a single image). Furthermore, *it is the coordinate name that is used to establish the association, after any projection has been taken into account*. E.g. `HPLN-TAN` and `HPLN-TAB` are both recognised as just `HPLN` with respect to association.

If the value column contains no other coordinates than those present in the referring HDU, and no dimensions without a coordinate, only a single keyword value is associated with any pixel in the referring HDU. This is because the association uniquely determines the position within the value column based on the position in the referring HDU. Such variable keywords are called single-valued – see Examples 1 and 3.

However, if the value column contains coordinates that are not present in the referring HDU, or dimensions without an assigned coordinate, there are multiple values within the value column that apply to any given pixel in the referring HDU. Such variable keywords are called multi-valued – see Example 2.

Example 1 – Single-valued variable keyword associated by a single shared coordinate

Let us assume that the atmospheric coherence length `ATMOS_R0` is recorded during the observations described by the example header above, with a different cadence than the observations.

For each exposure in the observation series, there is a single value of `ATMOS_R0` that applies to all pixels in that exposure. Thus, the `ATMOS_R0` value column should be one-dimensional, and the only coordinate that needs to be specified is time.

The header of the corresponding binary table extension `MEASUREMENTS` might contain the following entries:

```
EXTNAME = 'MEASUREMENTS'      / Extension containing measured auxiliary values
DATEREFP = '2018-01-01T12:00:00' / Time coord. zero point (time reference, mandatory)
TTYP5 = 'ATMOS_R0'           / Column 5 contains values for ATMOS_R0
1CTYP5 = 'UTC'                / Time coordinate
TDIM5 = '(4700)'             / Array dimensions for column 5
```

As we can see, the values of `1CTYP5` is identical to the value of `CTYPE3` in the referring HDU described above. This is the **only** basis for the association of coordinates. The coordinate numbers (`i=1` vs `i=3`) are irrelevant in the association, and it is the sum of the `UTC` coordinate and `DATEREFP` in the respective extensions that matters in the matching up of the two data cubes. Time spans, dimension sizes (`NAXIS3` vs `TDIM5`), and other characteristics given by WCS keywords, e.g. cadence (`CDEL3` vs `1CDLT5`; not shown), are also irrelevant.

Now, in order to find the value of `ATMOS_R0` for a given point in the referring data cube, the time corresponding to that point must be calculated. A reverse calculation is done for the value column to locate the point where its time coordinate has the same value. Then, the `ATMOS_R0` value can be extracted from that point in the value column (using linear interpolation as specified in the FITS standard).

Note that the zero point for the time coordinate (`DATEREF`) *must* be given for both extensions when one of the specified coordinates are `UTC`, but `DATE-BEG` is mandatory only for the Obs-HDU. In fact, if `DATE-BEG` is present in a binary table extension, it applies to all columns in this extension, which implies that the acquisition of data starts at the same time for all columns. When it is desirable to give specific, different `DATE-BEG` values for each column, the value columns must instead be stored in separate extensions. The `DATEREF` value also applies to all columns with a `UTC` coordinate, but is only used as a zero-point for that coordinate and has no other meaning.

Example 2 – Multi-valued keyword associated by a single shared coordinate

Let us assume that for the same observation, the gain of each of four ADC converters (one for each detector quadrant) varies throughout the image sequence, and these values are also recorded. In this case, there are 4 values that may be associated with each image, stored in the variable keyword `GAINS`.

The data cube of this value column is three-dimensional, with the `UTC` coordinate varying along one of the dimensions. This is the only shared coordinate, and it will be used for the association between each image and the measured gains. The other two value column dimensions represent the 2x2 detector quadrants, and the value of `GAINS` for a given image will be a 2x2 array.

If desired, coordinates may be specified for non-shared dimensions. Such coordinates may be regular WCS coordinates (e.g. `WAVE`, `STOKES`, `HPLN/HPLT` etc.), but they may also be *ad hoc* coordinates with no meaning defined within the WCS framework. In this example, we use the *ad hoc* coordinates `QUADRANT_X` and `QUADRANT_Y`.

The `GAINS` value column might be specified by having the following entries (among others) in the binary table extension `PARAMETERS`:

```
EXTNAME = 'PARAMETERS' / Extension containing instrument parameters
DATEREF = '2018-01-01T00:00:00' / Time coord. zero point (time reference, mandatory)
TTYPE2 = 'GAINS' / Column 2 contains values for GAINS
1CTYP2 = 'QUADRANT_X' / Quadrant number (X)
2CTYP2 = 'QUADRANT_Y' / Quadrant number (Y)
3CTYP2 = 'UTC' / Time coordinate
TDIM2 = '(2,2,30)' / Array dimensions for column 2
```

The associated data retain any coordinates that are not used in the association. Thus, in this example, the 2x2 `GAINS` array associated with an image has its own coordinates, `QUADRANT_X` and `QUADRANT_Y`, as specified by `1CTYP2` and `2CTYP2` (and any other relevant WCS keywords such as `1CDLT2` and `2CDLT2` etc.).

Note that it is possible to explicitly mark coordinates that should not be used in the association, by setting the `iCNAME` keywords to a string starting with `'UNASSOCIATED'`, see Appendix I-b.

Example 3 – Single-valued keyword associated by multiple shared coordinates

Of course, the referring HDU and the value column may have more than one shared coordinate. Since each quadrant from the previous example covers different areas on the Sun, it is possible (though perhaps not very useful) to assign them Solar coordinates. As an example, the binary table extension from Example 2 might instead have the following `iCTYPn` entries:

```

1CTYP2 = 'HPLN-TAN'      / Solar X coordinate
2CTYP2 = 'HPLT-TAN'      / Solar Y coordinate
3CTYP2 = 'UTC'           / Time coordinate

```

Since `CTYPE1='HPLN-TAN'` and `CTYPE2='HPLT-TAN'` in the referring HDU, this would have a dramatic effect on the interpretation of the values in the value column! Since the value column now has all of its coordinates in common with the referring HDU, a lookup of `GAINS` for a specific pixel in a given image would first calculate the `HPLN-TAN`, `HPLT-TAN`, and `UTC` values for that pixel (taking `DATEREF` into account), and these values would be used in the reverse calculation for the value column. The result would be a *single number* as in Example 1, not a 2x2 array as in Example 2.

As a side remark, it is of course possible to write a general look-up utility function that takes an explicit subset of those coordinates that are to be used in the association. It would then be possible to specify that only the time should be used in the look-up if it were desirable to retrieve the full 2x2 array for a given image.

Example 4 – Real-life example with multi-valued keyword associated by a single shared coordinate, and with table look-up of coordinates

While we are waiting for headers from Mats Löfdahl: below is a preliminary draft outlining some of the keywords that might end up in this example:

Primary header:

```

CTYPE3 = 'UTC'           / Coord. 3 is time in seconds relative to DATEREF
VAR_KEYS= 'VAR-EXT-ATMOS_R0;ATMOS_R0' / Extension storing ATMOS_R0 values

```

Binary table extension header:

```

XTENSION= 'BINTABLE'      / Written by IDL: Mon Sep  4 17:56:40 2017
:
TFIELDS =                  3 / Number of columns
EXTNAME = 'VAR-EXT-ATMOS_R0' / Variable keyword, coordinate-tabulated assoc.
DATEREF = '2016-09-19T00:00:00.000' / Time reference is midnight
COMMENT
COMMENT Column 1: Tabulated ATMOS_R0
COMMENT
TFORM1 = '86E'            / Real*4 (floating point)
TTYPE1 = 'ATMOS_R0'       / Table of SOLARNET variable-keyword
TDIM1 = '(2,43)'         / Array dimensions for column 1
TUNIT1 = 'm'             / Units of column 1
1CNAM1 = 'WFS subfield size for ATMOS_R0' /
1CTYP1 = 'WFSSZ-TAB'      / Not a WCS coordinate.
1CUNI1 = 'pix'           /
1PS1_0 = 'VAR-EXT-ATMOS_R0' / Axis 1 in col. 1 is in VAR-EXT-ATMOS_R0
1PS1_1 = 'WFSSZ-ATMOS_R0' / Col. with val. for axis 1 of col. 1
1PVI_3 =                  1 / WFSSZ 1st (only) coord. in WFSSZ-ATMOS_R0
2CNAM1 = 'Coord. 2 for col. 1 (ATMOS_R0) is time.' /
2CTYP1 = 'UTC--TAB'      / WCS time coordinate.
2CUNI1 = 's'             /
2PS1_0 = 'VAR-EXT-ATMOS_R0' / Axis 2 in col. 1 is in VAR-EXT-ATMOS_R0
2PS1_1 = 'TIME-ATMOS_R0' / Col. with val. for axis 2 of col. 1
2PVI_3 =                  1 / UTC 1st (only) coord. in TIME-ATMOS_R0
COMMENT
COMMENT Column 2: WFS subfield size (coord. 1 for ATMOS_R0)
COMMENT
TFORM2 = '2I'            / Integer*2 (short integer)
TTYPE2 = 'WFSSZ_ATMOS_R0' / Tabulations of WFSSZ for ATMOS_R0

```

```

TUNIT2 = 'pix      ' / Units of column 2
TDIM2  =          2 /
COMMENT
COMMENT Column 3: Measurement times (coord. 2 for ATMOS_R0)
COMMENT
TFORM3 = '43D      ' / Real*8 (double precision)
TTYPE3 = 'TIME-ATMOS_R0' / Tabulations of TIME for ATMOS_R0
TUNIT3 = 's        ' / Units of column 3
TDIM3  =          43 /

```

Appendix I-b. Preventing unwanted coordinate associations

In some cases, it is useful to specify value columns with coordinates that are common with coordinates in the referring HDU, but without implying an association by that coordinate. An example would be a time series of temperatures needed for calibration purposes, where the entire temperature history applies to every pixel in the referring HDU. Following the normal association rules, however, only a single value in the temperature series would be returned if a lookup is performed for a specific image, assuming that both the referring HDU and the value column has a `UTC` time coordinate. To prevent unwanted coordinate association, value column coordinates for which association is not wanted should have the corresponding coordinate name `iCNAN` start with `'UNASSOCIATED'`.

Example 5 – Preventing association of a common coordinate

Given a time series of temperatures (`TEMPS`) with a `UTC` coordinate as described above, the header of the corresponding binary table extension `MEASUREMENTS` might contain the following entries:

```

EXTNAME = 'MEASUREMENTS' / Extension containing measured auxiliary values
DATEREf = '2018-01-01T12:00:00' / Time coord. zero point (time reference, mandatory)
TTYPE6  = 'TEMPS      ' / Column 6 contains values for TEMPS
TDIM6   = '(60)      ' / Array dims for column 6
ICTYP6  = 'UTC       ' / 1st dimension is time
ICNA6   = 'UNASSOCIATED Time' / i=1 is an unassociated coordinate or dimension

```

This means that no coordinates in the `TEMPS` value column will be associated with coordinates in the referring HDU, and the 60 array stored in the value column data cube therefore applies to every (x, y, t) pixel in the referring HDU. Therefore, `TEMPS` is said to be a constant, multi-valued keyword in Appendix I-c below.

Appendix I-c. Constant, multi-valued keywords

As exemplified in Appendix I-b, it is possible to use the variable-keyword mechanism to specify keywords that are multi-valued but entirely independent of the referring HDU's WCS coordinates. This is simply a matter of not having any associated coordinates (using the `UNASSOCIATED` mechanism when necessary), and by not invoking the pixel-to-pixel association explained in Appendix I-d. In Example 5 above, `TEMPS` is such a constant, multi-valued keyword.

Appendix I-d. Pixel-to-pixel association

Some variable keywords encode discrete-valued properties. In such cases, it might be important to ensure an exact correspondence between pixels in the referring HDU's data cube and pixels

in the value column's data cube, without any round-off errors in the floating-point calculations of WCS coordinates.

When standard WCS calculations are used in the association between the referring HDU and the value column, such round-off errors will interfere with any exact pixel-to-pixel correspondence, and linear interpolation will be used. I.e. if a variable keyword represents a discrete-valued property, association by coordinates may result in non-discrete values. If instead a direct pixel-to-pixel association is desirable, the variable-keyword mechanism may be used as described below.

Even for non-discrete-valued keywords it may be simpler and more illustrative to use a pixel-to-pixel association in some cases. This is typically the case for values that have been measured in sync with the observations. Another example could be values varying along one detector dimension, e.g. one value per detector row.

In order to signal such an exact pixel-to-pixel association, the `WCSNn` keyword for the value column must start with `'PIXEL-TO-PIXEL'`. In this case, no coordinate specified for the value column will be used in the association. Also, all dimensions of the data cube in the referring HDU must be present in the value column (in the same order). Dimensions in the referring HDU for which the variable keyword has a constant value should be collapsed into a singular dimensions. Trailing dimensions may be added in order to specify variable keywords with multiple values for each pixel in the referring HDU (see Example 7), and coordinates tied to these dimensions may be specified if desirable.

Example 6 – Single-valued keyword with pixel-to-pixel association

If the `ATMOS_R0` values from Example 1 in Appendix I-a had been recorded in sync with the 60 images, i.e. a single `ATMOS_R0` value is recorded for each image, the binary table extension might instead contain the following entries:

```
EXTNAME = 'MEASUREMENTS'      / Extension name of binary table extension
WCSN5   = 'PIXEL-TO-PIXEL'    / Column 5 uses pixel-to-pixel association
TTYPE5  = 'ATMOS_R0'         / Column 5 contains values for ATMOS_R0
TDIM5   = '(1,1,60)'         / Array dimensions for column 5
```

This means that the `ATMOS_R0` value for any referring HDU pixel (x, y, t) is found in pixel $(1, 1, t)$ of the `ATMOS_R0` value column data cube.

The pixel-to-pixel association may also be used if `ATMOS_R0` had been recorded with a lower cadence than the images. If e.g. `ATMOS_R0` was recorded for every 20th image then the value found in in pixel $(1, 1, 0)$ of the `ATMOS_R0` column data cube applies to the first 20 images. A total of 3 `ATMOS_R0` measurements would have been made, and `TDIM5='(1,1,3)'`.

Generically, when `WCSNn='PIXEL-TO-PIXEL'`, if the size of a dimension j in the variable keyword data cube is $1/N$ of the corresponding dimension of the data cube of the referring HDU, the pixel index $p_{j,v}$ for the variable keyword data cube can be found from the referring HDU's data cube pixel index $p_{j,d}$ through the formula $p_{j,v} = \text{floor}((p_{j,d} - 1)/N) + 1$.

Example 7 – Multi-valued keyword with pixel-to-pixel association

If the variable keyword is multi-valued for each point in the referring HDU's data cube, trailing dimensions may be added to the value column data cube. Accompanying WCS coordinate keywords may also be added.

Following Example 2 in Appendix I-a, but with the `GAINS` values recorded in sync with the observation time series, the binary table extension might contain the following entries:

```
EXTNAME = 'PARAMETERS'           / Extension name of binary table extension
WCSN2   = 'PIXEL-TO-PIXEL'       / Column 2 uses pixel-to-pixel association
TTYPE2  = 'GAINS'                / Column 2 contains values for GAINS
TDIM2   = '(1,1,60,2,2)'         / Array dimensions for column 2
4CTYP2  = 'QUADRANT_X'          / Quadrant number (X)
5CTYP2  = 'QUADRANT_Y'          / Quadrant number (Y)
```

This means that the 2x2 array of `GAINS` values for any referring HDU pixel (x, y, t) is found in pixel $(1, 1, t, *, *)$ of the `GAINS` value column data cube.

Appendix II. Pixel list mechanism

In some cases, it is useful to store attributes (numbers or strings) that apply only to specific pixels within an Obs-HDU, or to e.g. simply flag certain pixels of the HDU (see Section 5.6.2). One example is to store the location of hot/cold pixels. Another example is to store the location and original values (as attributes) of pixels affected by cosmic rays/spikes. Yet another example might be to highlight or label (even with a string) specific points within the data cube – such as where a reduction algorithm has broken down.

Since the pixel list mechanism described here may be used by any HDU with a non-zero `SOLARNET` keyword, we will from now on simply use the term “referring HDU” for the HDU that uses this mechanism.

This mechanism uses a specific implementation of the pixel list FITS standard (Paper I, Section 3.2), where binary table extensions are used to store pixel indices and attributes (if any) associated with each pixel. Pixel indices and attributes alike are stored in separate binary table columns, with one row per listed pixel.

As an example, if we want to use a pixel list to store an attribute associated with 2 pixels of a 3-dimensional data cube, the pixel list might contain the following table values:

| | Column 1 (pixel index 1) | Column 2 (pixel index 2) | Column 3 (pixel index 3) | Column 4 (attribute) |
|------------------|-----------------------------|-----------------------------|-----------------------------|-------------------------|
| Row 1 (pixel #1) | 5 | 10 | 1 | 500 |
| Row 2 (pixel #2) | 5 | 13 | 43 | 489 |

This should be interpreted such that the attribute associated with pixel (5,10,1) has the value 500, and that the attribute associated with pixel (5,13,43) has the value 489.

In the general case, for a referring HDU with an n -dimensional data cube, the first n columns of a corresponding pixel list contain pixel indices, labelled by `TTYPEn= 'DIMENSIONk'`, where k is the dimension number in the referring HDU. Any subsequent columns contain the attributes associated with those pixels. Columns containing attribute values will be called “attribute columns” below. The column names (`TTYPEn`) for attribute columns must be equal to the name of the attribute.

In order to establish the connection between the referring HDU and a pixel list, the referring HDU must contain the keyword `PIXLISTS`. The syntax of `PIXLISTS` is analogous to that of `VAR_KEYS` (see Appendix I): it must declare the `EXTNAME` of the extension containing the pixel list, followed by a semicolon, then a comma-separated list of any attribute names. When multiple pixel lists are used, this is signalled by adding a comma, the `EXTNAME` of the next pixel list extension followed by a semicolon, etc. Note that even when a pixel list does not contain any attributes, a comma is needed before the `EXTNAME` of any subsequent pixel list.

The `EXTNAME` of pixel lists may carry meaning (e.g. `LOSTPIXLIST`, see Section 5.6.2). But if a pixel list `EXTNAME` ends with a “tag” (see Appendix I), this does not change its meaning. Thus, such tags may be used to distinguish between different extensions containing pixel lists of the same type/meaning for different referring HDUs. Multiple referring HDUs may refer to the same pixel list, even if it has a tag.

As an example, in order to refer to all types of pixel lists mentioned in Section 5.6.2, the referring HDU's `PIXLISTS` could contain the following:

```
PIXLISTS= 'LOSTPIXLIST;, MASKPIXLIST;,      &' / Lost and masked pixels
CONTINUE 'SATPIXLIST[He_I];ORIGINAL,      &' / He_I saturated pixels w/original values
CONTINUE 'SPIKEPIXLIST[He_I];ORIGINAL,CONFIDENCE, &' / Spike pixels for He_I
CONTINUE 'SUNSPOTS;CLASSIFICATION' / Sunspot locations and classification
```

The pixel list name `SUNSPOTS` used above is arbitrarily chosen as an example, i.e. this `EXTNAME` does not carry any predefined meaning in a SOLARNET context.

Example 1 – Pixel list with attribute columns

The header of an Obs-HDU with dimensions $(\lambda, x, y) = (20, 100, 100)$ might contain the following entry:

```
PIXLISTS= 'SPIKEPIXLIST;ORIGINAL,CONFIDENCE' / List of spike pixels
```

This means that `SPIKEPIXLIST` is a pixel list with two attribute columns, `ORIGINAL` and `CONFIDENCE`. The header of this binary table extension might include the following entries:

```
EXTNAME = 'SPIKEPIXLIST' / Extension name
TTYPE1  = 'DIMENSION1' / Col.1 is index into data cube dimension 1
TTYPE2  = 'DIMENSION2' / Col.2 is index into data cube dimension 2
TTYPE3  = 'DIMENSION3' / Col.3 is index into data cube dimension 3
TTYPE4  = 'ORIGINAL' / Col.4 contains original values of listed pixels
TTYPE5  = 'CONFIDENCE' / Col.5 contains confidence values of spike detection
TCTYP1  = 'PIXEL ' / Indicates that col. 1 is a pixel index
TCTYP2  = 'PIXEL ' / Indicates that col. 2 is a pixel index
TCTYP3  = 'PIXEL ' / Indicates that col. 3 is a pixel index
TPC1_1  = 1 / Indicates that col. 1 is a pixel index
TPC1_2  = 0 / Indicates that col. 1 is a pixel index
TPC1_3  = 0 / Indicates that col. 1 is a pixel index
TPC2_1  = 0 / Indicates that col. 2 is a pixel index
TPC2_2  = 1 / Indicates that col. 2 is a pixel index
TPC2_3  = 0 / Indicates that col. 2 is a pixel index
TPC3_1  = 0 / Indicates that col. 3 is a pixel index
TPC3_2  = 0 / Indicates that col. 3 is a pixel index
TPC3_3  = 1 / Indicates that col. 3 is a pixel index
```

The presence of the `TCTYPn` and `TPCnk` elements for n and k between 1 and 3 signals that this binary table extension is a pixel list, and that columns 1 to 3 are pixel indices. Conversely, the absence of these keywords for columns 4 and 5 indicate that they are attribute columns that do not contain pixel indices but rather associated attributes (see Paper I, end of Section 3.2). The use of 'PIXEL' as a coordinate name (`TCTYPn`) is taken from Wells et al. (1981), Appendix A, Section III-F.

The general rule is that when using binary table extensions in this manner, it is a two-dimensional table of cells, with $n+m$ *columns* and x *rows*, where n is the number of dimensions in the referring HDU's data array, m is the number of attribute columns, and x is the number of listed pixels. Each cell may only contain a single number or a string.

Thus, if we want to flag 3 pixels in the above mentioned referring HDU data cube, and store the values `ORIGINAL` and `CONFIDENCE` for each pixel, the pixel list might contain the following table values (column headings are `TTYPEn` values):

| | DIMENSION1 | DIMENSION2 | DIMENSION3 | ORIGINAL | CONFIDENCE |
|------------------|------------|------------|------------|----------|------------|
| Row 1 (pixel #1) | 5 | 10 | 1 | 500 | 0.91 |
| Row 2 (pixel #2) | 5 | 11 | 1 | 489 | 0.91 |
| Row 3 (pixel #3) | 8 | 55 | 73 | 1405 | 0.98 |

Example 2 – Pixel list with no attribute columns

We could list pixels that were lost during acquisition but were later filled in with estimated values. In this case, there is no original value, thus there are no attributes to associate with the pixels. An Obs-HDU might then contain:

```
PIXLISTS= 'LOSTPIXLIST[He_I];' / Lists of lost pixels for Obs-HDU He_I
```

In the header of the pixel list binary table with `EXTNAME='LOSTPIXLIST[He_I]'`, only the first 3 columns would be present (`N=3, m=0`) and the table values might be:

| | DIMENSION1 | DIMENSION2 | DIMENSION3 |
|------------------|------------|------------|------------|
| Row 1 (pixel #1) | 1 | 10 | 3 |
| Row 2 (pixel #2) | 2 | 10 | 3 |
| Row 3 (pixel #3) | 3 | 10 | 3 |

Appendix II-a. Flagging *N*-dimensional slices of a data cube

One problem with the above usage of pixel lists arises when it is desirable to flag e.g. an entire exposure. When specifying pixel indices for all dimensions, that would require flagging each pixel in that exposure individually, i.e. having one row for each pixel. This would typically mean several million rows per flagged exposure – and each row requires several times more storage space than the pixel being flagged!

However, in such a case it is possible to simply omit the table columns that specify the indices for dimension 1 and 2 of the referring HDU, since these are not needed in order to pin-point the exposure.

The remaining column(s), containing indices for relevant dimension(s), must still be labelled with `TTYPEn='DIMENSIONk'`, where, `k` indicates the dimension number in the referring HDU's data cube.

Example 3 – Pixel list flagging a 2-dimensional slice of a 3-dimensional data cube

E.g. to flag specific exposures in a data cube with dimensions (`x, y, t`), the pixel list header might contain the following:

```
EXTNAME = 'BAD_EXPOSURES' / Extension name
TTYPE1 = 'DIMENSION3' / Col.1 is index into data cube dimension 3
TTYPE2 = 'REASON' / Col.2 contains reason for bad exposure
TCTYP1 = 'PIXEL' / Indicates that col. 1 is a pixel index
TPC1_1 = 1 / Indicates that col. 1 is a pixel index
```

And the table values of the pixel list might be:

Commented [SVHH1]: Bill Thompson: I think it would also be useful to have a way to flag pixel ranges, e.g. a block within the image that was degraded by a lost telemetry packet.

Commented [SVHH2R1]: Indeed, that would be useful. One idea might be to flag the lower left & upper right pixels (can be extended to multi-dimensional data).

Would it require some other referring keyword than `PIXLISTS`? Or just annotate the bintable extension with some keyword specifying that the pixel "list" is in fact given in this form? With columns labelled with `TTYPEn` values (`DIMENSION1-LL, DIMENSION2-LL, ...DIMENSIONX-LL, DIMENSION1-UR, DIMENSION2-UR, ...DIMENSIONX-UR`)? Of course, with the possibility of additional columns containing details for the flagging.

| | DIMENSION3 | REASON |
|------------------------------------------|------------|-------------------------|
| Row 1 (1 st flagged exposure) | 8 | 'SHUTTER FAILURE' |
| Row 2 (2 nd flagged exposure) | 21 | 'WRONG FILTER POSITION' |

This should be interpreted such that exposure 8 is flagged as bad because of a shutter failure (i.e. pixels (*,*,8) are bad), and exposure 21 is bad because of a wrong filter position (pixels (*,*,21) are bad).

Appendix III. Meta-observation mechanism

Most users expect to be able to analyse at least one file at a time on a laptop, preferably with all of the data loaded into memory. Thus, at some point, files become too large for comfort⁴ when following the guidelines for what to store in a single file/single Obs-HDU in a strict sense.

An obvious solution to this problem for a file that contains multiple Obs-HDUs would be to split it into multiple files containing only a single HDU each. However, high-cadence, high-resolution observations often produce extremely large amounts of data, and at some point this strategy will not be enough to keep file sizes reasonable. Thus, the issue of prohibitively large files *must* be dealt with in a more generic way while preserving the “spirit” of the guidelines for what should be stored together.

We do this by providing this mechanism to logically connect HDUs stemming from a single observation series that *ought* to be put in a single HDU according to the guidelines. It allows HDUs to be split into smaller constituent HDUs, stored in separate files and individually recorded in an SVO as separately retrievable observation units, whilst *also* recording the entire observation as a meta-observation unit, without duplicating the data. The meta-observation unit is represented by a meta-HDU that contains header keywords representing the observation’s global attributes like duration, data statistics etc. for searchability reasons, but the data are only retrievable as a collection of constituent files.

Note that it may be impossible for the pipeline to calculate values like cadence (and cadence variation) during the writing of each constituent HDU. However, this information may still be reported in the meta-HDU.

In general, a meta-HDU may contain keywords that are not present in the constituent HDUs.

Although we recommend having a copy of the meta-HDUs in each constituent file⁵, this is not a requirement. In fact, for some pipelines, it makes sense to have meta-HDUs only in the last file, since many of the global attributes are not known until the last constituent HDUs have been processed.

In this appendix, we only discuss splitting observations in the *time dimension*. Although it is possible to use the mechanism to split observational data in any other dimension, this seems likely to be confusing for users, and we therefore do not recommend it.

To give an example: A series of 10000 images might have to be split over 10 files, each having a single Obs-HDU containing 1000 images, with any accompanying specifications of variable keywords, pixel lists, or other HDUs. All HDUs in those 10 constituent files should be self-consistent, and it should be possible to analyse each file independently. In fact, each file should be created just as if it were not part of such a meta-observation⁶, except for some additional information and some additional rules/metadata relating to the meta-observation.

⁴ Processing easily doubles the size of data to be kept in memory. Data cubes prepared for immediate visualisation are often accessed using memory-mapped files, though, so files of “arbitrary size” are ok.

⁵ A Meta-HDU does not take up much space, as it only contains header keywords and a singular data array.

⁶ The file names should also reflect this - e.g. the date/time part should reflect the start of the observations contained in each constituent file.

The mechanism is designed to make it possible to construct a *stitching utility* that can collect constituent HDUs into “ideal” HDUs that *would* have been created if it were not for file size considerations. Here we describe the additional rules and metadata required to make such a stitching utility work.

First of all, constituent HDUs must have the same `EXTNAME` in order to be stitched together. I.e. the HDUs with `EXTNAME=He_I` inside the constituent files will be stitched together into a new `He_I` HDU, and `c_I` HDUs will be stitched together into a new `c_I` HDU.

In addition, *all* HDUs that belong to a meta-observation *must have the following additional keywords*:

- `METAFIL` must be set to the name of the *first* file that contains parts of the observation unit (excluding any file type suffixes).
- `METADIM` set to the dimension that has been split. E.g. when splitting an array (x, y, t) into time chunks, `METADIM=3`. Note that an accompanying auxiliary HDU with dimensions (t, y) would set `METADIM=1`. `METADIM=0` should be used for e.g. auxiliary HDUs whose data array dimensions does not contain the split dimension. It is allowed to have `METADIM=NAXIS+1`. E.g. if constituent HDUs have dimensions (x, y, λ) and `METADIM=4`, the output meta-HDU will have dimensions (x, y, λ, t) . Note that the meta-HDU in the constituent files must have a set of WCS keywords that correctly describe the resulting array, including any added dimensions.

However, in order to correctly register a meta-observation unit, the global values of all keywords (including instrument-specific) must also be available.

This is of course only a problem for those keywords that are not constant among all the constituent HDUs. E.g. `DATE-BEG`, `DATE-END` and other keywords that vary as a function of the split dimension, or as a function of the data itself (e.g. `DATAMAX` and `DATAMIN`) must be given explicitly for the meta-observation. Also, the global scalar value of any variable keyword should be given if it has been specified in the constituent HDUs.

This is achieved through a *meta-Obs-HDU* containing the meta-observation unit’s keyword values - i.e. the values that would have resulted from storing the meta-observation in a single HDU. The `NAXISj` keywords are exempt, since the meta-Obs-HDUs should contain a singular data array. This meta-HDU should occur in each file containing any constituent HDUs⁷.

The `EXTNAME` of such a meta-Obs-HDU *must* be the same as the (common) `EXTNAME` of the constituent HDUs, with the string “;METAHDU” appended.

Using the keywords given above and the meta-Obs-HDU, it is now possible to reconstruct/stitch together constituent Obs-HDUs into an ideal Obs-HDU with a correct header. It is also possible to reconstruct HDUs containing their corresponding variable keyword specifications and pixel lists.

⁷ Although from an SVO standpoint, such (identical) meta-HDUs are only required in one of the files, they do not take up much storage space since they only contain keyword values, and they might be useful to people who are analysing constituent files one file at a time.

The logic behind the stitching is as follows, given a collection of files that make up a meta-observation:

The name of the output file containing the stitched meta-observation will be “<METAFILE>_META.fits”. Below is a pseudo-code description of the subsequent steps in the stitching:

```
FOR each file (in the order given):
  FOR each HDU/EXTNAME:
    IF SOLARNET<>0 OR HDU is a META-HDU:
      IF HDU is a pixel list HDU:
        Adjust index in the split dimension
      ENDIF
      Collect/aggregate HDU along split dimension (given by
      METADIM) with earlier HDUs with same EXTNAME
    ENDIF ELSE:
      Keep HDU, replacing any earlier HDU with same EXTNAME
    ENDELSE
  ENDFOR
ENDFOR
```

For all types of HDUs, only the last encountered header is preserved, but the `NAXIS` and `NAXISj` keywords are recomputed to match the stitched (aggregated) data array.

When finished, there is one HDU for each distinct `EXTNAME` that occurred in the file collection.

The final step is to go through all meta-HDUs, i.e. HDUs with `EXTNAME`s ending in “;METAHDU”. The “;METAHDU” string is clipped off the `EXTNAME`. If one of the processed HDUs has the resulting string as its `EXTNAME`, its keywords will be replaced by the keywords in the meta-HDU. However, FITS keywords in the aggregated HDU describing the data array size (`NAXIS`, `NAXISj`) are not replaced, since they must represent the size of the data array in the target HDU (not the size of the singular data array in the meta-HDU).

In other words, the guidelines for what to store together in a single file/single Obs-HDU may be followed strictly, interpreting them as guidelines for what to store together in a single meta-file/single meta-Obs-HDU.

Appendix IV. Adaptation to binary table extensions

This section outlines how to adapt the SOLARNET recommendations for data stored as columns in binary table extensions (Cotton et al.). It is written for an audience that already has experience in using binary table extensions for this purpose, so many details are deliberately left out.

First of all, for any column we consider the combination of column-specific keywords (`TTYPEn`, `TDIMn`, etc), general header keywords (`FILENAME`, `CREATOR`, etc), and the associated (column) data as a self-contained quasi-HDU, entirely analogous to the normal concept of an HDU. Thus, whenever the term HDU (as in Obs-HDU) is used elsewhere in this document, it may be taken to refer to such a quasi-HDU instead of an actual HDU.

However, for such quasi-HDUs, column-specific keywords replace general header keywords according to established standards and conventions for binary tables. E.g. for column `n`, `TDIMn` replaces `NAXIS` and `NAXISj`, `TZEROn` replaces `BZERO` etc. In particular, almost all WCS keywords for image extensions have binary table column equivalents. For WCS keywords without a column-specific form, the value applies to all columns. Thus, if different values of such WCS keywords are necessary for separate columns, the data *must* be placed in separate binary table extensions.

The column-specific keyword `TTYPEn` is normally used analogously to how `EXTNAME` is used for image extensions, but binary table extensions must also have an `EXTNAME` keyword set according to the rules in Section 2.

The column-specific keywords `TVARn` replaces `VAR_KEYS`, and `TPXLn` replaces `PIXLISTS` (see Appendix I and Appendix II).

The naming conventions for column-specific keywords (starting with `T` and allowing for 3-digit column numbers) leaves only 4 letters to carry meaning, which easily leads to the creation of very awkward column-specific keyword names. In order to alleviate this problem for keywords that must have different values for different columns, the column-specific keyword `TKEYSn` is introduced, listing pairs of keyword names and values inside a string. The [CONTINUE Long String Keyword Convention](#) may of course be used to improve readability and add comments, e.g.:

```
TKEYS3 = 'OBS_HDU=1,                &' / Contains observational data
CONTINUE 'DETECTOR="ZUN_A_HIGHSPEED2", &' / Detector 2
CONTINUE 'WAVELNTH=1280            ' / [Angstrom] Principal wavelength
```

The syntax is relatively straightforward – a comma-separated list of keyword-value pairs, with string values in *double* quotes. Spaces are ignored (except inside strings).

Warning: non-SOLARNET-aware FITS reading software will *not* recognize values inside `TKEYSn`. Thus, FITS standard keywords – including WCS keywords – must never be given in `TKEYSn`. If no no appropriate column-specific variant is valid and different values are necessary for different columns, the columns *must* instead be stored in separate binary table extensions. Thus, `TKEYSn` should be used only for project-specific and SOLARNET-specific keywords.

Appendix V. Other recommendations or suggestions

Appendix V-a. File naming suggestions

Although file-naming recommendations are of little consequence to an SVO, they can be a big help for users to get an overview of files stored locally, so we will give some (hopefully helpful) suggestions below.

The file name should be given in the keyword `FILENAME` (this is useful in case the actual file name is changed). `FILENAME` is mandatory for fully SOLARNET-compliant Obs-HDUs.

We recommend that file names only contain letters `A-Z` and `a-z`, digits `0-9`, periods, underscores and plus/minus signs. Each component of the file name should be separated with an underscore – not a minus sign. In this regard, a range may be considered a single component with a minus sign between the min and max values (such as start/end date). File name components with numerical values should be a) preceded with one or more identifying letters, and b) given in a fixed-decimal format, e.g. `(00.0300)`. Variable-length string values should be post-fixed with underscores to a fixed length.

Another common practice has been to start the file name with the “instrument name” – although typically defined in a consistent manner only on a *per mission* or *per observatory* basis - i.e. collisions may appear with other missions. Thus, we recommend prefixing the instrument name with a mission or observatory identifier (e.g. `iris` for IRIS or `sst` for SST).

After the instrument name, the data level is normally encoded as e.g. “I0” and “I1” for level 0 and 1. Note, however, that the definitions of data levels are normally entirely project/instrument-specific and does not by itself uniquely identify what kinds of processing have been applied.

Within each data set it is often very useful to have file names that can be sorted by time when subject to a lexical sort (such as with “`1s`”). This requires that the next item in the file name should be the date and time (`YYYYMMDD_HHMMSS[.d]`). The “`d`” part is fractional seconds, with enough digits to distinguish between any two consecutive observations.

If the data might be made available (simultaneously) with e.g. different processing emphasis (e.g. trade-offs between resolution and noise level), an alphanumeric identifier⁸ of the processing mode should be added in order to ensure uniqueness of the file name.

What comes next is highly instrument-specific, but attributes that specify the type of content should definitely be encoded, e.g. which filter and wavelength has been used, which type of optical set-up has been used, etc. In particular, the wavelength is very useful for those who are not familiar with a data set.

Note that if file names (including file type suffixes) are longer than 68 characters, it will have to be split over two or more lines in a FITS header using the OGIP 1.0 long string convention. Likewise, if file name lengths exceed 67 characters, a comma-separated list of file names cannot be represented with one line per file. Thus, file names using 67 or fewer characters is preferable for human readability of FITS headers.

⁸ E.g. a short form of the contents of `PR_MODE`, see Section 8.

The names of files containing different observations *must* be unique - i.e. all files must be able to coexist in a single directory. If the above recommendations do not result in a unique name, some additional information *must* be added. We recommend that file names for different versions of the same data should be identical – i.e. downloading a new version will overwrite any older version in the same directory.

Appendix V-b. Storing data in a single file or in separate files

From an SVO point of view, *each Obs-HDU represents a single “observation unit”*, and will be registered separately. Thus, the choice of putting such observation units into separate files or not does not matter to an SVO in terms of searchability, but it does matter in terms of the file sizes of data that may be served. However, the meta-observation mechanism may be used to circumvent this issue (see Appendix III).

However, some “typical use” aspects should still be considered - including how most existing utilities⁹ interact with observations of a particular type:

As a general rule, Obs-HDUs that would typically be analysed/used together and are seldom used as stand-alone products should be stored in the same file, whereas Obs-HDUs that are often analysed/used as stand-alone products should be stored in separate files. Furthermore, Obs-HDUs with data of fundamentally different types (e.g. filter images vs. spectra vs. Fabry-Pérot data vs. spectropolarimetry) should *not* be put in the same file.

Obviously, data processed by separate pipelines cannot be stored in a single file (unless they are combined at a later stage).

Examples and arguments in favour of a single file:

- Data from different Stokes parameters (in the same wavelength) are normally analysed together and should be put together in a single file.
- Data from a spectrometer raster¹⁰ are normally stored in a single file, even though the data may contain information from multiple detector readout windows. They have normally been acquired in a synchronous fashion, and they may be analysed together in order to have better estimates of continuum values when performing line fitting.
- Pointing adjustments *in order to track* solar features by means of solar rotation compensation or feature tracking should not automatically cause the data to be stored in separate files. This is somewhat dependent upon the frequency and magnitude of the tracking movements relative to the field of view, and the magnitude of any time gaps relative to the cadence, but we leave it up to the discretion of pipeline designers to determine when it is appropriate to split image sequences in such cases: if the data are suitable for making a single movie, store them in a single file (and a single HDU).

⁹ Some utilities may prefer different grouping of HDUs with respect to separate vs. single files, but that issue may be solved by a utility program that is able to join HDUs in separate files into a single file and vice versa.

¹⁰ Rasters are observations (usually spectrometric) collected by stepping a slit across the observation area.

- Having too many files can lead to inefficiency both on a file system basis and on the level of utilities.
- Not least, having too many files will also be an inconvenience to users who want to look at file lists manually.

Examples and arguments in favour of separate files:

- Observations with different `POINT_ID` values (see Section 7) should not be stored in the same file.
- Images/movies in different filters are often used as stand-alone products, even if a parallel observation in another filter exists. Thus, observations from different filters should be put in separate files.
- In a similar fashion, Fabry-Pérot scans of separate wavelength regions should go in separate files. The same applies to similar observations such as from spectropolarimetry.
- Some observation series with very low cadence should be stored with each image in a separate file. The definition of “very low cadence”, however, is somewhat dependent on the type of data, the resolution, and the variability time scale of resolved features. The “normal use” of the data also matters: If images are largely used as stationary context for other observations, they should definitely go into separate files. This is typically the case for synoptic observation series, which are also normally of indefinite length and therefore must be split up one way or another anyway.
- Observations with significantly different starting and/or end times should not be stored in a single file. “Significantly different” in this context means on the order of a few times the cadence/exposure time or larger - since there may be technical reasons for differences smaller than this.

An additional aspect is that grouping data into a single file makes it impossible to download “only the interesting part” of a data set for a given analysis purpose. However, given the guidelines above, we think this is unlikely to be an issue. Also, a future SVO could provide file splitting “services”, such as selecting only specific HDUs from multi-HDU files.

When an Obs-HDU (partially) overlaps in time and space with one or more HDUs stored in other files, `CCURRENT` (concurrent) may be set to a comma-separated list of its own file name plus the names of all files containing concurrent Obs-HDUs¹¹.

`CCURRENT` serves as a pointer to other concurrent (and probably relevant) observations, but it also serves a purpose in grouping search results (see Section 7).

Appendix V-c. Obs-HDU content guidelines

In addition to guidelines determining how data should be stored in single vs. multiple files, we here give guidelines for what should be considered as single vs. multiple observational units - i.e. what should be stored in a single vs. multiple Obs-HDUs within each file.

¹¹ This is of course on a “best effort” basis for the pipelines!

Such guidelines can only be given heuristically, due to the large diversity of possible data sets.

Each Obs-HDU will be registered as an individual observation unit, with attributes such as duration, minimum/maximum wavelength etc. Such attributes may be important search terms, and this must be taken into account when considering what should be collected into a single Obs-HDU or not.

As with the guidelines for keeping data in single or multiple files, some “typical use” aspects must also be considered - including how most existing utilities¹² interact with observations of a particular type. In addition, any “user convenience” issues should be taken into account.

Some examples and arguments in favour of a single HDU:

- If a Fabry-Pérot scan is stored with each exposure (each wavelength) as a separate HDU, a search made for data covering a particular wavelength inside the scan’s min/max range will not match any of the HDUs if the desired wavelength falls between the bandpasses of the individual exposures.
- Different Stokes parameters are so intrinsically tied to each other that they should be stored together in a single HDU. There is also an existing convention for how to do this, see Section 8.5 in the FITS Standard.
- If an observation is repeated with a more or less fixed cadence (except for small cadence variations caused by e.g. technical issues/limitations), this will not be immediately apparent if each repetition is stored as a separate HDU.
- Observations are often visualised by displaying slices of a multi-dimensional data array, sometimes also scanning through one of the dimension in order to visualise it as a movie (though not necessarily in the time dimension). Data that are likely to be visualised in such ways should be put into a single HDU. In other words, all dimensions through which slicing or scanning may be desirable should be included in a single HDU¹³.
- Pointing changes in e.g. slit-jaw movies due to slit movements should not cause the movie to be split into separate HDUs, even if there are relatively large pointing changes associated with the starting of new rasters in a series.
- Uneven spatial sampling, e.g. dense in the centre and sparse in the periphery, should not cause the data to be separated into multiple HDUs, though note that a table lookup form of WCS coordinates must be used.
- Variable exposure times due to *Automatic Exposure Control (AEC)* should not cause the exposures to be stored in separate HDUs - as long as the settings for the AEC is constant.

¹² Some utilities may prefer different practices, but a relatively simple program that is able to split or join HDUs in specific dimensions would solve the problem.

¹³ In particular, the time dimension should be included when observations are repeated and are suitable to be presented as a movie. Repeated rasters have traditionally not been collected into single HDUs, but that may be because of their low cadence - causing relatively few files to be created during a single observation run. This is changing, however, and we recommend that repeated rasters should be joined into single HDUs which include the time dimension, e.g. (x, y, λ, t) .

Examples and arguments in favour of separate HDUs:

- Observation units stored in separate files according to the guidelines Appendix V-b are of course stored as separate HDUs.
- If the readout of a spectrometer has gaps (i.e. only small portions of the spectrum are extracted, in “wavelength windows”), the different wavelength windows should not be stored in a single HDU, since that would falsely indicate that the observation unit covers the entire spectrum between the minimum and maximum wavelengths.
- Some observation series are made with alternating long and short exposure times. These should not be collected in a single Obs-HDU, because of the resulting difficulty in describing the exposure time¹⁴ as well as the complexity that would be required in utilities in order to handle/display such data correctly. Instead, the data should be separated into one HDU with long exposures and one HDU with short exposures.
- Data that are often displayed side by side, such as images in different filters should be split into separate HDUs.

For very closely connected, parallel observations, it is preferable to handle the grouping of data into Obs-HDUs in the same way for all of the observations, even if they are not of the same type (e.g. repeated rasters and corresponding slit-jaw movies).

Bearing all of the above in mind, observations that fit the following description should be collected into a single HDU:

An array of data that has (quasi-)uniform spacing in each physical dimension, e.g. x, y, lambda, and time, and also has (quasi-)constant attributes such as pointing, exposure times, gain, filter, and other relevant settings.

¹⁴ The variable-keyword mechanism could be used for e.g. `XPOSURE` if such data are stored in a single HDU, e.g. `(x, y, t, 2)`, with `XPOSURE` stored as `(1, 1, 1, 2)`. But this is much less self-explanatory than having two separate HDUs, and it would require users/utilities to be aware of the mechanisms and how to use them. Since it is not absolutely necessary to do it this way, the variable-keyword mechanism should not be used.

Appendix VI. Extended mechanism for distortion corrections

For some complicated data sets, the existing FITS WCS mechanism for distortion corrections becomes untenable. This is particularly true when describing cavity errors. We have therefore extended the formalism given in Paper V to allow an arbitrary number of distortion corrections, calculated on the basis of any coordinate stage (pixel coordinates, intermediate pixel coordinates, and intermediate world coordinates) and applied to any other subsequent coordinate stage.

While the formalism for this mechanism extension is largely worked out, it is also very complex, and considered unfinished until fully implemented in at least one pipeline – most likely the CHROMIS pipeline will be the first one.

For this reason, we refrain from including a full specification and explanation in the current version of the document.

However, if anyone has a need to describe complex distortions, please contact s.v.h.haugan@astro.uio.no for advice.

11. References

- [The FITS Standard, version 3.0 \(Pence et al. 2010, A&A, 524, A42, 40 pp., https://www.aanda.org/articles/aa/pdf/2010/16/aa15362-10.pdf\)](https://www.aanda.org/articles/aa/pdf/2010/16/aa15362-10.pdf)
- Paper I: [Representations of World Coordinates in FITS \(Greisen & Calabretta, 2002, A&A, 395, 1061-1075, http://www.aanda.org/articles/aa/pdf/2002/45/aah3859.pdf\)](http://www.aanda.org/articles/aa/pdf/2002/45/aah3859.pdf)
- Paper II: [Representations of celestial coordinates in FITS \(Calabretta & Greisen, 2002, A&A, 395, 1077-1122, http://www.aanda.org/articles/aa/pdf/2002/45/aah3860.pdf\)](http://www.aanda.org/articles/aa/pdf/2002/45/aah3860.pdf)
- Paper III: [Representations of spectral coordinates in FITS \(Greisen, Calabretta, Valdes & Allen, 2006, A&A, 446, 747-771, http://www.aanda.org/articles/aa/pdf/2006/05/aa3818-05.pdf\)](http://www.aanda.org/articles/aa/pdf/2006/05/aa3818-05.pdf)
 - Authors' web sites, supplemental background: [Eric Greisen, Mark Calabretta, http://www.atnf.csiro.au/people/mcalabre/WCS/index.html](http://www.atnf.csiro.au/people/mcalabre/WCS/index.html)
 - [An unofficial errata for Papers I, II, and III \(Calabretta & Greisen, http://fits.gsfc.nasa.gov/wcs/errata_20071222.pdf\)](http://fits.gsfc.nasa.gov/wcs/errata_20071222.pdf)
- Paper IV: [Representations of Time Coordinates in FITS \(Rots, 2015, A&A, 574, A36, https://www.aanda.org/articles/aa/pdf/2015/02/aa24653-14.pdf\)](https://www.aanda.org/articles/aa/pdf/2015/02/aa24653-14.pdf)
- Paper V: [Representations of distortions in FITS world coordinate systems \(Calabretta, Valdes, Greisen, Allen, ADASS, 2004, 314, http://fits.gsfc.nasa.gov/wcs/dcs_20040422.pdf\)](http://fits.gsfc.nasa.gov/wcs/dcs_20040422.pdf)
- [Coordinate systems for solar image data \(Thompson, 2006, A&A, 449, 791-803, http://www.aanda.org/articles/aa/pdf/2006/14/aa4262-05.pdf\)](http://www.aanda.org/articles/aa/pdf/2006/14/aa4262-05.pdf)
- [FITS: A Flexible Image Transport System \(Wells et al. 1981, A&AS, 44, 363\)](#)
- [Precision effects for solar image coordinates within the FITS world coordinate system \(Thompson, 2010, A&A, 515, A59, http://www.aanda.org/articles/aa/pdf/2010/07/aa10357-08.pdf\).](http://www.aanda.org/articles/aa/pdf/2010/07/aa10357-08.pdf)
- [The SolarSoft WCS Routines: A Tutorial \(Thompson, http://hesperia.gsfc.nasa.gov/ssw/gen/idl/wcs/wcs_tutorial.pdf\)](http://hesperia.gsfc.nasa.gov/ssw/gen/idl/wcs/wcs_tutorial.pdf)

- Binary table extension to FITS (Cotton et al., 1995, A&AS, **113**, 159-166, <http://adsabs.harvard.edu/abs/1995A%26AS..113..159C>).
- [Checksum Keyword Convention](http://fits.gsfc.nasa.gov/registry/checksum.html) (<http://fits.gsfc.nasa.gov/registry/checksum.html>)
- [The FITS Header Inheritance Convention](https://fits.gsfc.nasa.gov/registry/inherit/fits_inheritance.txt) (https://fits.gsfc.nasa.gov/registry/inherit/fits_inheritance.txt)
- [The CONTINUE Long String Keyword Convention](https://fits.gsfc.nasa.gov/registry/continue_keyword.html) (https://fits.gsfc.nasa.gov/registry/continue_keyword.html)
- [Space Physics Archive Search and Extract \(SPASE\) instrument types](http://www.spase-group.org/data/reference/spase-2_2_8/spase-2_2_8_xsd.htm) (http://www.spase-group.org/data/reference/spase-2_2_8/spase-2_2_8_xsd.htm - InstrumentType and http://www.spase-group.org/docs/dictionary/spase-2_2_8.pdf)
- [Recommendations for Data & Software Citation in Solar Physics](#) (2012AAS...22020127H)
- [Best Practices for FITS Headers](#) (2012AAS...22020128H, http://sdac.virtualsolar.org/docs/SPD2012/2012_SPD_FITS_headers.pdf)
- VSO Checklists, <http://virtualsolar.org/checklists>
- VSO Minimum Information for Solar Observations, <http://docs.virtualsolar.org/wiki/MinimumInformation>
- The Unified Content Descriptors, Version 1+ (UCD1+) <http://www.ivoa.net/documents/latest/UCDlist.html>

Other sources of keywords with established use:

- Solar Orbiter FITS keyword definitions (in preparation, contact s.v.h.haugan@astro.uio.no for latest version)
- STEREO Standard FITS keywords (http://isoc.stanford.edu/doc/keywords/STEREO/STEREO_site_standard_fits_keywords.txt)
- SDO/AIA FITS keyword definitions (https://www.lmsal.com/sdodocs/doc?cmd=dcur&proj_num=SDOD0019&file_type=pdf)
- IRIS FITS keyword definitions (http://www.lmsal.com/iris_science/irisfitskeywords.pdf)

Part B. Lists of mandatory and optional FITS keywords with example values

12. Mandatory keyword for all HDUs (Section 2.1)

In addition to all keywords required by the FITS Standard, all HDUs in SOLARNET FITS files must contain the keyword `EXTNAME`, with a value that is unique within the file.

```
EXTNAME = 'He_I' / Name of HDU
```

13. Mandatory keywords for all Obs-HDUs (Section 2.2)

```
SOLARNET= 0.5 / Fully SOLARNET-compliant=1.0, partially=0.5
OBS_HDU = 1 / This HDU contains observational data
DATE-BEG= '2020-12-24T17:12:00.5' / Date of start of observation
```

14. Mandatory WCS keyword for all HDUs with a UTC (time) coordinate (Section 4.1)

```
DATEREF = '2020-12-24T00:00:00' / Time coordinate zero point
```

15. Mandatory keywords for fully SOLARNET-compliant Obs-HDUs

The keywords listed in this section are mandatory for fully SOLARNET-compliant Obs-HDUs. However, most keywords are only "conditionally mandatory", depending on the data content, which mechanisms have been used, instrument type, etc.

15.1. Mandatory general keywords (Sections 8 and Appendix V-a)

```
FILENAME= 'sleep_a_zen_12_20201224_170000.1_balanced.fits'
DATASUM = '2503531142' / Data checksum
CHECKSUM= 'hcHjic9ghcEghc9g' / HDU checksum
DATE = '2020-12-31T23:59:59' / Date of FITS file creation
ORIGIN = 'University of Oslo' / Location where FITS file has been created
```

15.2. Fundamental WCS coordinate keywords (Section 3.1)

Obs-HDUs must contain all WCS coordinate specifications that are required to *adequately describe the observations* for their normal use. This includes e.g. the use of extra coordinates for singular dimensions when necessary (i.e. `WCSAXES` may be greater than `NAXIS`), or the use of alternate WCS coordinate systems (with WCS keywords ending in a letter A-Z). Normally, WCS keywords that must be included are `CRVALi`, `CDELTi`, `CRPIXj`, `CUNITi`, `CTYPEi`, and when necessary also e.g. `WCSAXES`, `PCi_j` (or `CDi_j`), `CRDERi`, and `CSYERi`.

The example below is a brief excerpt of a header describing an observation where the random and systematic errors in the time coordinate are so large that they may be important for certain types of analysis.

```
CTYPE3 = 'UTC'          / Coordinate 3 is time
CUNIT3 = 's'           / Units for time coordinate
CRPIX3 =               1 / Reference pixel: time starts at first image
CRVAL3 =               0 / [s] Offset from DATEREF of reference pixel
CDEL3 =               0.1 / [s] Sampling is 0.1 seconds
CRDER3 =               0.03 / [s] Large random clock error (sample-to-sample)
CSYER3 =               5 / [s] Large systematic error, clock may be off by 5s
```

15.3. Mandatory WCS positional keywords (Section 3.2)

15.3.1. Mandatory for ground based observatories (Section 3.2)

```
OBSGEO-X= 5327395.9638 / [m] Observer's fixed geographic X coordinate
OBSGEO-Y= -1719170.4876 / [m] Observer's fixed geographic Y coordinate
OBSGEO-Z= 3051490.766 / [m] Observer's fixed geographic Z coordinate
```

15.3.2. Mandatory for Earth orbiting satellites (Section 3.2)

```
GEOX_OBS= 1380295.0032 / [m] Observer's non-fixed geographic X coordinate
GEOY_OBS= 57345.1262 / [m] Observer's non-fixed geographic Y coordinate
GEOZ_OBS= 9887953.9454 / [m] Observer's non-fixed geographic Z coordinate
```

15.3.3. Mandatory for deep space missions (not Earth orbiting satellites) (Section 3.2)

```
HGLN_OBS= -0.0572950 / Observer's Stonyhurst heliographic longitude
HGLT_OBS= 5.09932 / Observer's Stonyhurst heliographic latitude
DSUN_OBS= 1498142450 / [m] Distance from instrument to Sun centre
```

15.4. Mandatory data description keywords (Sections 5.1, 5.2 and 5.6.2)

```
BTYPE = 'Intensity' / Description of what the data array represents
BUNIT = 'DN' / Units of data array
XPOSURE = 2.44 / [s] Accumulated exposure time
```

When the data are a result of multiple summed exposures with identical exposure times, the keyword `TEXPOSUR` and `NSUMEXP` must be set:

```
TEXPOSUR= 1.22 / [s] Single-exposure time
NSUMEXP = 2 / Number of summed exposures
```

`NBINj` and `NBIN` is mandatory if the data has been binned:

```
NBIN1 = 2 / Binning in dimension 1
NBIN2 = 4 / Binning in dimension 2
```

NBIN = 8 / Product of all NBIN_j

Missing or blank pixels in floating-point-valued HDUs should be set to *NaN*, but missing or blank pixels in integer-valued HDUs must be given the value of **BLANK**:

BLANK = -100 / Value of missing pixels (integer HDU)

15.5. Mandatory keywords identifying the origin of the observations (Sections 6 and 7)

A *subset* of the following keywords is mandatory in the sense that the subset must be sufficient to uniquely identify the origin of the observations, and they should be present to the extent that they make sense for the given observations (e.g. **MISSION** might not make sense for ground based observations, or there might be no sensible value for **PROJECT**).

```
PROJECT = 'Living With a Star' / Name of project
MISSION = 'SLEEP ' / Name of mission
OBSRVTRY= 'SLEEP A ' / Name of observatory
TELESCOP= 'ZUN ' / Name of telescope
TELCONF= 'STANDARD' / Telescope configuration
INSTRUME= 'ZUN ' / Name of instrument
CAMERA = 'cam1 ' / Name of camera
GRATING = 'GRISM_1 ' / Name of grating/grism used
FILTER = 'Al_med, open' / Name of filter(s)
DETECTOR= 'ZUN_A_HIGHSPEED1' / Name of detector
OBS_MODE= 'lo-res-hi-speed12b' / Name of predefined settings used during obs.
SETTINGS= 'fpos=123,vpos=3' / Additional instrument/acquisition settings
DATATAGS= '"ESA", "NASA", "ESA/NASA"' / Additional information
```

Note that e.g. **DETECTOR**, **GRATING**, and **FILTER** this might seem unnecessary for instruments with an *a priori* single fixed value, but for ground based observatories, upgrades of an instrument might include a change of e.g. filters.

15.6. Mandatory keywords for spectrographs and filter instruments (Sections 3.2 and 5.4)

```
WAVEUNIT= -10 / Wavelength related kwds have unit: 10^(WAVEUNIT) m
WAVEREF = 'vac ' / Wavelength related kwds in vacuum

WAVEMIN = 582.10 / [Angstrom] Min wavelength covered by filter
WAVEMAX = 586.62 / [Angstrom] Max wavelength covered by filter
```

For spectrographs, and narrow-band filter instruments (when radial velocity is of importance when interpreting the observations), the following keyword is mandatory:

OBS_VR = 36.62 / [km/s] Observer's outward velocity w.r.t. Sun

Also, to signal that no wavelength correction has been applied (even if the observer moves with respect to the Sun with a velocity given by **obs_vr**) the following WCS keywords are mandatory:

```
SPECSYS = 'TOPOCENT' / Coordinate reference frame = observer
VELOSYS = 0.0 / [m s-1] No velocity correction applied to WAVE coord.
```

15.7. Mandatory keyword for spectrographs (Section 5.4)

SLIT_WID= 0.5 / [arcsec] Slit width

15.8. Mandatory keyword for grouping (Sections 7 and Appendix V-b)

```
POINT_ID= '20201224_165812_200' / Unique (re-)pointing ID
```

16. Mandatory keyword for SOLARNET HDUs that contain keywords with a definition in conflict with the specifications in this document (Section 2.2)

If a SOLARNET HDU contains SOLARNET keywords with definitions that are in conflict with the definitions in this document, those keywords *must* be listed as a comma-separated list in the keyword SOLNETEX, e.g.:

```
SOLNETEX= 'PLANNER, ATMOS_R0' / Exception: kws with conflicting definitions
```

Note that *none* of the keywords that are mandatory for a given HDU may have conflicting definitions¹⁵.

17. Mandatory keywords for all HDUs that uses any of the variable-keyword, pixel list or meta-observation mechanism (Sections 2.1, 2.2, 2.3, Appendix I, Appendix II and Appendix III)

Any HDU using one of these mechanisms must have SOLARNET set to a non-zero value, even non-Obs-HDUs (which should use a value of -1). In addition, EXTNAME must be set according to the guidelines in Section 2. Finally, the respective VAR_KEYS, PIXLISTS or METADIM/METAFIL must be set – see Appendix I, Appendix II and Appendix III for details.

```
SOLARNET = -1.0 / SOLARNET mechanisms may be used
EXTNAME = 'zunhousekeeping' / Name of HDU
VAR_KEYS= 'He_I_T3;TEMPS' / Variable keyword used by this Aux-HDU
PIXLISTS= 'He_I_T3;T_IDX' / Pixel list used by this Aux-HDU
METADIM = 4 / Split dimension of meta-observation
METAFIL = 'sleep_a_zun_12_20201224_165812.2_balanced.fits' / First file in meta-obs
```

17.1. Mandatory keyword for binary table extension value columns with a coordinate that should not be used in coordinate association (Appendix I-b)

The value of iCNA_n must start with “UNASSOCIATED” to signal that coordinate *i* is not to be used in the association:

```
2CNA6 = 'UNASSOCIATED Time' / i=2 is an unassociated coordinate or dimension
```

¹⁵ If some existing utility requires a different definition of a mandatory keyword, we recommend that the value for the non-SOLARNET definition be given in a new keyword, and that the software be modified.

17.2. Mandatory keyword for binary table extension value columns that uses pixel-to-pixel association (Appendix I-d)

The value of `wcsnn` must start with “`PIXEL-TO-PIXEL`” to signal that a direct pixel-to-pixel association applies:

```
WCSN5 = 'PIXEL-TO-PIXEL' / Column 5 uses pixel-to-pixel association
```

18. Optional keywords for all Obs-HDUs

The keywords in this section are optional for both fully and partially SOLARNET-compliant Obs-HDUs.

18.1. Optional keywords describing cadence (Section 5.3)

```
CADENCE = 2.5 / [sec] Planned cadence
CADAVG = 2.45553 / [sec] Average actual cadence
CADMIN = 2.27943 / [sec] Minimum actual frame-to-frame spacing
CADMAX = 2.69162 / [sec] Maximum actual frame-to-frame spacing
CADVAR = 0.0118546 / [sec] Variance of frame-to-frame spacing
```

18.2. Optional keywords characterising the instrument/data (Sections 3.1, 5.4 and 5.5)

```
WAVEBAND= 'He I 584.58 A' / Strongest emission line in data
WAVELNTH= 584.61 / [Angstrom] Characteristic wavelength in vacuum
RESOLVPW= 6530 / Resolving power of spectrograph
COORDREF= 'hisol_12_20201224_104332_012_195.fits,John-Doe' / Coord system refs
FT_LOCK = 1 / Feature tracking on
ROT_COMP= 1 / Solar rotation compensation on (1)/off (0)
ROT_MODL= 'ALLEN' / Model used for rotation compensation
AO_LOCK = 0.9 / Adaptive optics status, 0.0=no lock, 1.0=lock
AO_NMODE= 2 / Type of modes: Zernike, Karhunen-Loeve
```

If desirable, `AWAVLNTH` may be used to specify the characteristic wavelength in air instead of vacuum. The magnitude of `WAVELNTH` and `AWAVLNTH` must be specified in `WAVEUNIT`, given as the power of 10 to which the metre is raised, e.g. `WAVEUNIT=-10` for Angstrom.

The response function may be given in the variable keyword `RESPONSE`. If the data has been corrected for a variable response, the response function that has been applied should instead be given in `RESPAPPL`:

```
RESPONSE= 0.76 / Mean of response function
OR
RESPAPPL= 0.52 / Mean of applied response function
```

18.3. Optional quality aspects keywords (Sections 3.1 and 5.5)

```
ATMOS_R0= 15 / [cm] Atmospheric coherence length
ELEV_ANG= 76.2 / [deg] Telescope elevation angle.
OBS_LOG = 'logs/2020/12/24/' / URL of observation log
RSUN_REF= 6.95508E+08 / [m] Solar rad. used for calc. px scale
COMPQUAL= 0.75 / Quality of data after lossy compression
```

```
COMP_ALG= 'jpeg2000'      / Name of lossy compression algorithm
```

18.4. Optional data statistics keywords (Section 5.6)

```
DATAMIN =          -23 / [DN] Minimum of data
DATAMAX =          4077 / [DN] Maximum of data
DATEMEAN=         144.794 / [DN] Mean of data
DATEMEDN=         11.0000 / [DN] Median of data
DATAP01 =          -14 / [DN] 1st percentile of data
DATAP10 =           -9 / [DN] 10th percentile of data
DATAP25 =           -4 / [DN] 25th percentile of data
DATAP75 =           70 / [DN] 75th percentile of data
DATAP90 =           304 / [DN] 90th percentile of data
DATAP95 =           704 / [DN] 95th percentile of data
DATAP98 =           2085 / [DN] 98th percentile of data
DATAP99 =           2844 / [DN] 99th percentile of data
DATARMS =          471.524 / [DN] Root mean square of data
DATAKURT=          24.8832 / Kurtosis of data
DATASKEW=          4.84719 / Skewness of data
```

18.5. Optional keywords for missing and saturated pixels (Section 5.6.1)

```
NTOTPIX =          262144 / Expected number of data pixels
NLOSTPIX=           512 / Number of lost pix. b/c acquisition problems
NSATPIX =           4 / Number of saturated pixels
NSPIKPIX=           7 / Number of noise spike pixels
NMASKPIX=           71 / Number of masked pixels
NDATAPIX=          261550 / Number of usable pix. excl lost/miss/spik/sat
PCT_LOST=           0.195312 / NLOSTPIX/NTOTPIX*100
PCT_SATP=           0.00152588 / NSATPIX/NTOTPIX*100
PCT_SPIK=           0.00267029 / NSPIKPIX/NTOTPIX*100
PCT_MASK=           0.02708435 / NMASKPIX/NTOTPIX*100
PCT_DATA=           99.7734070 / NDATAPIX/NTOTPIX*100
```

18.6. Optional keywords identifying the origin of the observations (Section 6 and Appendix V-b)

```
OBSERVER= 'John Doe'      / Operator(s) who acquired the data
PLANNER = 'Jane Doe'      / Observations planner(s)
REQUESTR= 'Jane Doe'      / Name(s) of person requesting this particular
                           observation
CAMPAIGN= 'FlareHunt791,JOP922' / Coordinated campaign name(s)
CCURRENT= 'sleep_a_zen_l2_20201224_170000_120_balanced.fits,&' / Concurrent,
CONTINUE 'sleep_b_zen_l2_20201224_170001_045_balanced.fits,&' / overlapping
CONTINUE 'sleep_a_magneto_l2_20201224_170001_030_full.fits,&' / observations
CONTINUE 'sleep_b_magneto_l2_20201224_170001_808_full.fits,&' / (multiple files)
CONTINUE 'sleep_b_magneto_l2_20201224_170001_909_full.fits' /
```

18.7. Optional pipeline processing keywords (Sections 8, 8.1 and 8.2)

```
LEVEL = '3'            / Data level of fits file
VERSION = 2            / FITS file processing generation/version

CREATOR = 'ZUN_MOMF'   / Name of software that produced the FITS file
VERS_SW = '2.5'        / Version of software applied
VERS_CAL= '2.4'        / Version of calibration pack applied
PRSTEP1 = 'MOMFBD'    / Processing step name
```

```
PRPROC1 = 'ZUN_MOMF'           / Name of procedure used
PRMODE1 = 'BALANCED'          / Processing mode
PRPARA1 = 'ITER=5,FG=7,FILL=1' / List of parameters/options for PRPROCn
PRLIB1  = 'ZUNRED '           / Software library containing ZUN_MOMF
PRVER1  =                      1.5 / Library version/MJD of last update
PRBR1   = 'Master '           / GIT repository branch
PRLIB1A = 'SSW '              / Additional software library
PRVER1A =                      59214 / Additional library version/MJD of last update
```

18.8. Optional keyword for administrative information (Section 9)

```
INFO_URL= 'http://sleep.esa.int/zun/info.html' / Data set resource web page
```

18.9. Optional date and time keywords (Section 4)

```
DATE-END= '2020-12-24T17:00:02.5' / Date of end of observation
DATE-AVG= '2020-12-24T17:00:01.3' / Average date of observation
TIMESYS = 'UTC '                  / Time scale of the time-related keywords.
```

***Part C. Alphabetical listings of FITS
keywords with section
references***

Commented [TF3]: The section references are no longer correct – they will be updated in a future version of this document.

19. Alphabetical listing of all new SOLARNET keywords with section references

Below is an alphabetical listing of all SOLARNET keywords that are not part of the FITS standard or any widely accepted FITS convention, keywords that have been used in the past that do not have widely accepted definitions, or previously defined keywords that need to take special values in SOLARNET files:

AO_LOCK 7.5., 20.2.
 AO_NMODE 7.5., 20.2.
 ATMOS_RO 7.5., Appendix I., Appendix I-a., Appendix I-d., 20.3.
 AWAVMAX 7.4., 17.6.
 AWAVMIN 7.4., 17.6.
 CADAVG 7.3., 20.1.
 CADENCE 7.3., 20.1.
 CADMAX 7.3., 20.1.
 CADMIN 7.3., 20.1.
 CADVAR 7.3., 20.1.
 CAMERA 8., 17.5.
 CAMPAIGN 8., 20.6.
 CAR_ROT 5.2.
 CCURRENT Appendix V-b., 20.6.
 COMPQUAL 7.5., 20.3.
 COMP_ALG 7.5., 20.3.
~~COORDREF 5.1., 20.2.~~
 CREATOR 10.1., Appendix IV., 20.7.
 DATAKURT 7.6., 20.4.
 DATAMEAN 7.6., 20.4.
 DATAMEDN 7.6., 20.4.
 DATAPnn 7.6., 20.4.
 DATARMS 7.6., 20.4.
 DATASKEW 7.6., 20.4.
 DATATAGS 8., 17.5.
 DATEREF 6., 6.1., Appendix I., Appendix I-a., Appendix I-b., 16.
 ELEV_ANG 7.5., 20.3.
 EXTNAME 4.1., 4.2., 7.6.2., Appendix I., Appendix I-a., Appendix I-b., Appendix I-d., Appendix II., Appendix II-a., Appendix III., Appendix IV., 14., 19.
 FILTER 8., 9., Appendix II-a., 17.5.
 FT_LOCK 7.5., 20.2.
 iCNAn Appendix I-a., Appendix I-b., 19.1.
 iCDLTn Appendix I-a.
 iCTYPn Appendix I-a., Appendix I-b., Appendix I-d.
 iCUNIn Appendix I-b.
 iPSn_m Appendix I-b.
 iPVn_m Appendix I-b.
 INFO_URL 11., 20.8.
 LEVEL 10., 20.7.
 METADIM Appendix III., 19.
 METAFIL Appendix III., 19.
 MISSION 8., 9., 17.5.
 NBINj 7.2., 17.4.
 NDATAPIX 7.6.1., 20.5.
 NLOSTPIX 7.6.1., 20.5.
 NMASKPIX 7.6.1., 20.5.
 NSATPIX 7.6.1., 20.5.
 NSPIKPIX 7.6.1., 20.5.
 NSUMEXP 7.2., 17.4.
 NTOTPIX 7.6.1., 20.5.

Commented [TF4]: The section references are no longer correct – they will be updated in a future version of this document.

OBSRVTRY 8., 9., 17.5.
 OBS_HDU 4.2., Appendix IV., 15.
 OBS_LOG 7.5., 11., 20.3.
 OBS_VR 5.2., 17.6.
 PCT_DATA 7.6.1., 20.5.
 PCT_LOST 7.6.1., 20.5.
 PCT_MASK 7.6.1., 20.5.
 PCT_SATP 7.6.1., 20.5.
 PCT_SPIK 7.6.1., 20.5.
 PIXLISTS Appendix II., Appendix IV., 19.
 POINT_ID 9., Appendix V-b., 17.8.
 PRBRANa 10.2., 20.7.
 PRLIBna 10.2., 20.7.
 PRMODEn 10.2., 20.7.
 PROJECT 8., 9., 17.5.
 PRPARAn 10.2., 20.7.
 PRPROCn 10.2., 20.7.
 PRSTEPn 10.1., 10.2., 20.7.
 PRREFna
 PRVERn 10.2., 20.7.
 REQUEST 8., 20.6.
 RESOLVFW 7.4., 20.2.
 RESPAPPL 7.4., 20.2.
 RESPONSE 7.4., 20.2.
 ROT_COMP 7.5., 20.2.
 ROT_MODL 7.5., 20.2.
 SETTINGS 8., 9., 17.5.
 SLIT_WID 7.4., 17.7.
 SOLARNET 4.2., 4.3., Appendix I., Appendix II., Appendix III., 15., 19.
 SOLNETEX 4.2., 18.
 TELCONFG 8., 17.5.
 TEXPOSUR 7.2., 17.4.
 TKEYSn Appendix IV.
 TPXLSn Appendix IV.
 TVARKn Appendix IV.
 VAR_KEYS Appendix I., Appendix II., Appendix IV., 19.
 VERSION 10., 20.7.
 VERS_CAL 10.1., 20.7.
 VERS_SW 10.1., 20.7.
 WAVE 5.1., 7.4., Appendix I-a., 17.2.
 WAVEBAND 7.4., 20.2.
 WAVEMAX 7.4., 17.6.
 WAVEMIN 7.4., 17.6.
 WAVEUNIT 7.4., 17.6.
 WCSNn Appendix I-d., 19.2.

20. Alphabetical listing of all keywords with section references

Below is an alphabetical listing of all keywords used in this document, both SOLARNET keywords and FITS standard/widely accepted FITS convention keywords.

~~AIRORVAC 7.4., 17.6.~~
 AO_LOCK 7.5., 20.2.
 AO_NMODE 7.5., 20.2.
 ATMOS_R0 7.5., Appendix I., Appendix I-a., Appendix I-d., 20.3.
 AWAVMAX 7.4., 17.6.
 AWAVMIN 7.4., 17.6.
 BLANK 7.6.2., 17.4.
 BTYPE 7.1., 17.4.

Commented [TF5]: The section references are no longer correct – they will be updated in a future version of this document.

BUNIT 7.1., 17.4.
 BZERO Appendix IV.
 CADAvg 7.3., 20.1.
 CADENCE 7.3., 20.1.
 CADMAX 7.3., 20.1.
 CADMIN 7.3., 20.1.
 CADVAR 7.3., 20.1.
 CAMERA 8., 17.5.
 CAMPAIGN 8., 20.6.
 CAR_ROT 5.2.
 CCURRENT Appendix V-b., 20.6.
 CDELt_i Appendix I-a., 17.2.
 CDi_j 17.2.
 CHECKSUM 11., 17.1.
 COMMENT 4., 7.5., Appendix I.
 COMQUAL 7.5., 20.3.
 COMP_ALG 7.5., 20.3.
 CONTINUE 4., Appendix I., Appendix II., Appendix IV., 13., 20.6.
 COORDREF 5.1., 20.2.
 CRDER_i 5.1., 17.2.
 CREATOR 10.1., Appendix IV., 20.7.
 CRPIX_j 5.2., 17.2.
 CRVAL_i 17.2.
 CSYER_i 5.1., 17.2.
 CTYPE_i 5.1., 6.1., 7.1., Appendix I., Appendix I-a., 17.2.
 CUNIT_i 7.4., 17.2.
 DATAKURT 7.6., 20.4.
 DATAMAX 7.6., Appendix III., 20.4.
 DATAMEAN 7.6., 20.4.
 DATAMEDN 7.6., 20.4.
 DATAMIN 7.6., Appendix III., 20.4.
 DATAP_{nn} 7.6., 20.4.
 DATARMS 7.6., 20.4.
 DATASKEW 7.6., 20.4.
 DATASUM 11., 17.1.
 DATATAGS 8., 17.5.
 DATE 10., 17.1.
 DATE-AVG 6., 20.9.
 DATE-BEG 4.2., 6., 6.1., Appendix I., Appendix I-a., Appendix III., 15.
 DATE-END 6., Appendix I., Appendix III., 20.9.
 DATEREF 6., 6.1., Appendix I., Appendix I-a., Appendix I-b., 16.
 DETECTOR 8., Appendix IV., 17.5.
 DSUN_OBS 5.2., 6., 17.3.3.
 ELEV_ANG 7.5., 20.3.
 EXTNAME 4.1., 4.2., 7.6.2., Appendix I., Appendix I-a., Appendix I-b., Appendix I-d., Appendix II., Appendix II-a., Appendix III., Appendix IV., 14., 19.
 FILENAME Appendix IV., Appendix V-a., 17.1.
 FILTER 8., 9., Appendix II-a., 17.5.
 FT_LOCK 7.5., 20.2.
 GEOX_OBS 5.2., 17.3.2.
 GEOY_OBS 5.2., 17.3.2.
 GEOZ_OBS 5.2., 17.3.2.
 GRATING 8., 17.5.
 HGLN_OBS 5.2., 6., 17.3.3.
 HGLT_OBS 5.2., 6., 17.3.3.
 INFO_URL 11., 20.8.
 iCNAn Appendix I-a., Appendix I-b., 19.1.
 iCDLT_n Appendix I-a.
 iCTYP_n Appendix I-a., Appendix I-b., Appendix I-d.
 INSTRUME 8., 9., 17.5.
 LEVEL 10., 20.7.
 METADIM Appendix III., 19.
 METAFIL Appendix III., 19.

MISSION 8., 9., 17.5.
 NAXIS 5.1., 7.2., Appendix I., Appendix I-a., Appendix III., Appendix IV., 17.2.
 NAXISj 7.2., Appendix I., Appendix I-a., Appendix III., Appendix IV.
 NBIN 7.2., 17.4.
 NBINj 7.2., 17.4.
 NDATAPIX 7.6.1., 20.5.
 NLOSTPIX 7.6.1., 20.5.
 NMASKPIX 7.6.1., 20.5.
 NSATPIX 7.6.1., 20.5.
 NSPIKPIX 7.6.1., 20.5.
 NSUMEXP 7.2., 17.4.
 NTOTPIX 7.6.1., 20.5.
 OBSERVER 8., 20.6.
 OBSGEO-X 5.2., 17.3.1.
 OBSGEO-Y 5.2., 17.3.1.
 OBSGEO-Z 5.2., 17.3.1.
 OBSRVTRY 8., 9., 17.5.
 OBS_HDU 4.2., Appendix IV., 15.
 OBS_LOG 7.5., 11., 20.3.
 OBS_MODE 8., 9., 17.5.
 OBS_VR 5.2., 17.6.
 ORIGIN 10., 17.1.
 PCT_DATA 7.6.1., 20.5.
 PCT_LOST 7.6.1., 20.5.
 PCT_MASK 7.6.1., 20.5.
 PCT_SATP 7.6.1., 20.5.
 PCT_SPIK 7.6.1., 20.5.
 Pci_j 17.2.
 PIXLISTS Appendix II., Appendix IV., 19.
 PLANNER 8., 18., 20.6.
 POINT_ID 9., Appendix V-b., 17.8.
 PRBRAn 10.2., 20.7.
 PRLIBna 10.2., 20.7.
 PRMODEn 10.2., 20.7.
 PROJECT 8., 9., 17.5.
 PRPARAn 10.2., 20.7.
 PRPROCn 10.2., 20.7.
 PRSTEPn 10.1., 10.2., 20.7.
 PRVERn 10.2., 20.7.
 REQUESTR 8., 20.6.
 RESOLVPW 7.4., 20.2.
 RESPAPPL 7.4., 20.2.
 RESPONSE 7.4., 20.2.
 ROT_COMP 7.5., 20.2.
 ROT_MODL 7.5., 20.2.
 RSUN_ARC 5.2.
 RSUN_REF 5.1., 20.3.
 SETTINGS 8., 9., 17.5.
 SLIT_WID 7.4., 17.7.
 SOLARNET 4.2., 4.3., Appendix I., Appendix II., Appendix III., 15., 19.
 SOLAR_B0 5.2.
 SOLAR_EP 5.2.
 SOLAR_P0 5.2.
 SOLNETEX 4.2., 18.
 SPECSYSa 5.2., 17.6.
 TCTYPn Appendix II., Appendix II-a.
 TDIMn Appendix I-a., Appendix I-b., Appendix I-d., Appendix IV.
 TELECONF 8., 17.5.
 TELESCOP 8., 9., 17.5.
 TEXPOSUR 7.2., 17.4.
 TIMESYS 6., 20.9
 TKEYSn Appendix IV.
 TPCn_k Appendix II., Appendix II-a.

TPXLSn Appendix IV.
TTYPEn 7.6.2., Appendix I., Appendix I-a., Appendix I-b., Appendix I-d., Appendix II., Appendix II-a., Appendix IV.
TVARKn Appendix IV.
TZEROn Appendix IV.
VAR_KEYS Appendix I., Appendix II., Appendix IV., 19.
VELOSYSa 5.2., 17.6.
VERSION 10., 20.7.
VERS_CAL 10.1., 20.7.
VERS_SW 10.1., 20.7.
WAVE 5.1., 7.4., Appendix I-a., 17.2.
WAVEBAND 7.4., 20.2.
WAVLNTH 7.4., Appendix IV., 20.2.
WAVEMAX 7.4., 17.6.
WAVEMIN 7.4., 17.6.
WAVEUNIT 7.4., 17.6.
WCSAXES 5.1., 17.2.
WCSNn Appendix I-d., 19.2.
XPOSURE 7.2., 17.4.