4. Headers

The first two sections of this chapter define the structure and content of header keyword records. This is followed in Sect. 4.3 with recommendations on how physical units should be expressed. The final section defines the mandatory and reserved keywords for primary arrays and conforming extensions.

4.1. Keyword records

4.1.1. Syntax

Each 80-character header keyword record *shall* consist of a keyword name, a value indicator (only required if a value is present), an *optional* value, and an *optional* comment. Keywords *may* appear in any order except where specifically stated otherwise in this standard. It is *recommended* that the order of the keywords in *FITS* files be preserved during data processing operations because the designers of the *FITS* file may have used conventions that attach particular significance to the order of certain keywords (e.g., by grouping sequences of `COMMENT` keywords at specific locations in the header, or appending `HISTORY` keywords in chronological order of the data processing steps) or using `CONTINUE` keywords to generate long-string keyword values.

A formal syntax, giving a complete definition of the syntax of *FITS* keyword records, is given in Appendix A. It is intended as an aid in interpreting the text defining the standard.

In earlier versions of this standard a FITS keyword, assumed as an item whose value is to be looked up by name (and presumably assigned to a variable) by a FITS reading program, was associated one to one to a single header keyword record. With the introduction of (continued) long-string keywords (see 4.2.1.2), such FITS keywords may span more than one header keyword record, and the value *shall* be created by concatenation as explained in 4.2.1.2.

4.1.2. Components

4.1.2.1. Keyword name (bytes 1 through 8)

The keyword name *shall* be a left justified, 8-character, space-filled, ASCII string with no embedded spaces. All digits 0 through 9 (decimal ASCII codes 48 to 57, or hexadecimal 30 to 39) and upper case Latin alphabetic characters `'A'` through `'Z'` (decimal 65 to 90 or hexadecimal 41 to 5A) are permitted; lower case characters *shall not* be used. The underscore (`'_'`, decimal 95 or hexadecimal 5F) and hyphen (`'-'`, decimal 45 or hexadecimal 2D) are also permitted. No other characters are permitted.[1] For indexed keyword names that have a single positive integer index counter appended to the root name, the counter *shall not* have leading zeroes (e.g.,`NAXIS1`, not `NAXIS001`). Note that keyword names that begin with (or consist solely of) any combination of hyphens, underscores, and digits are legal.

4.1.2.2. Value indicator (bytes 9 and 10)

If the two ASCII characters `'=␣'` (decimal 61 followed by decimal 32) are present in bytes 9 and 10 of the keyword record this indicates that the keyword has a value field associated with it, unless it is one of the commentary keywords defined in Sect. 4.4.2 (i.e., a `HISTORY`, `COMMENT`, or completely blank keyword name) which by definition have no value.

4.1.2.3. Value/comment (bytes 11 through 80)

In keyword records that contain the value indicator in bytes 9 and 10, the remaining bytes 11 through 80 of the record *shall* contain the value, if any, of the keyword, followed by *optional* comments. In keyword records without a value indicator, bytes 9 through 80 *should* be interpreted as commentary text, however, this does not preclude conventions that interpret the content of these bytes in other ways.

The value field, when present, *shall* contain the ASCII text representation of a literal string constant, a logical constant, or a numerical constant, in the format specified in Sect. 4.2. The value field *may* be a null field; i.e., it *may* consist entirely of spaces, in which case the value associated with the keyword is undefined.

The mandatory *FITS* keywords defined in this standard *must not* appear more than once within a header. All other keywords that have a value *should not* appear more than once. If a keyword does appear multiple times with different values, then the value is indeterminate.

If a comment follows the value field, it *must* be preceded by a slash (`'/'`, decimal 47 or hexadecimal 2F).[1] A space between the value and the slash is strongly *recommended*. The comment *may* contain any of the restricted set of ASCII text characters, decimal 32 through 126 (hexadecimal 20 through 7E). The ASCII control characters with decimal values less than 32 (including the null, tab, carriage return, and line feed characters), and the delete character (decimal 127 or hexadecimal 7F) *must not* appear anywhere within a keyword record.

4.2. Value

The structure of the value field depends on the data type of the value. The value field represents a single value and not an array of values.[1] The value field *must* be in one of two formats: fixed or free. The fixed-format is required for values of mandatory keywords and is *recommended* for values of all other keywords.

4.2.1. Character string

4.2.1.1 Single record string keywords

A character string value *shall* be composed only of the set of restricted ASCII text characters, decimal 32 through 126 (hexadecimal 20 through 7E) enclosed by single quote characters (“`'`”, decimal 39, hexadecimal 27). A single quote is represented within a string as two successive single quotes, e.g., O’HARA = `'O''HARA'`. Leading spaces are significant; trailing spaces are not. This standard imposes no requirements on the case sensitivity of character string values unless explicitly stated in the definition of specific keywords.

If the value is a fixed-format character string, the starting single quote character *must* be in byte 11 of the keyword record and the closing single quote *must* occur in or before byte 80.

[1] This requirement differs from the wording in the original *FITS* papers. See Appendix H.

COPY

Earlier versions of this standard also *required* that fixed-format characters strings *must* be padded with space characters to at least a length of eight characters so that the closing quote character does not occur before byte 20. This minimum character string length is no longer required, except for the value of the XTENSION keyword (e.g., 'IMAGE␣␣␣' and 'TABLE␣␣␣'; see Sect. 7) which *must* be padded to a length of eight characters for backward compatibility with previous usage.

Free-format character strings follow the same rules as fixed-format character strings except that the starting single quote character *may* occur after byte 11. Any bytes preceding the starting quote character and after byte 10 *must* contain the space character.

Note that there is a subtle distinction between the following three keywords:

```
KEYWORD1= ''               / null string keyword
KEYWORD2= '   '            / empty string keyword
KEYWORD3=                  / undefined keyword
```

The value of KEYWORD1 is a null, or zero length string whereas the value of the KEYWORD2 is an empty string (nominally a single space character because the first space in the string is significant, but trailing spaces are not). The value of KEYWORD3 is undefined and has an indeterminate data type as well, except in cases where the data type of the specified keyword is explicitly defined in this standard.

The maximum length of a string value that can be represented on a single keyword record is 68 characters, with the opening and closing quote characters in bytes 11 and 80, respectively. In general, no length limit less than 68 is implied for character-valued keywords.

Whenever a keyword value is declared 'string' or said to 'contain a character string', the length limits in this section apply. The next section 4.2.1.2 applies when the value is declared 'long-string'.

4.2.1.2 Continued string (long-string) keywords

Earlier versions of this Standard only defined single record string keywords as described in the previous section. The Standard now incorporates a convention (originally developed for use in *FITS* files from high energy astrophysics missions) for continuing arbitrarily long string values over a potentially unlimited sequence of multiple consecutive keyword records using the following procedure:

1. Divide the long string value into a sequence of smaller substrings, each of which is no longer than 67 characters in length. (Note that if the string contains any literal single quote characters, then these must be represented as a pair of single quote characters in the *FITS* keyword value, and these 2 characters must both be contained within a single substring).
2. Append an ampersand character ('&') to the end of each substring, except for the last substring. This character serves as a flag to *FITS* reading software that this string value *may* be continued on the following keyword in the header.
3. Enclose each substring with single quote characters. Non-significant space characters may occur between the ampersand character and the closing quote character.
4. Write the first substring as the value of the specified keyword.
5. Write each subsequent substring, in order, to a series of keywords that all have the reserved keyword name CONTINUE (see 4.4.2) in bytes 1 through 8 and have space characters in bytes 9 and 10 of the keyword record. The substring may be located anywhere in bytes 11 through 80 of the keyword record and may be preceded by non-significant space characters starting in byte 11. A comment string may follow the substring; if present, the comment string must be separated from the substring by a forward slash character ('/'). Also, it is *strongly recommended* that the slash character be preceded by a space character.

The CONTINUE keyword must not be used with of any of the mandatory or reserved keywords defined in this standard unless explicitly declared of type long-string.

The following keyword records illustrate a string value that is continued over multiple keyword records. (Note: the length of the substrings have been reduced to fit within the page layout):

```
WEATHER = 'Partly cloudy during the evening f&'
CONTINUE  'ollowed by cloudy skies overnight.&'
CONTINUE  ' Low 21C. Winds NNE at 5 to 10 mph.'
```

If needed, additional space for the keyword comment field can be generated by continuing the string value with one or more null strings, as illustrated schematically below:

```
STRKEY  = 'This keyword value is continued &'
CONTINUE  ' over multiple keyword records.&'
CONTINUE  '&' / The comment field for this
CONTINUE  '&' / keyword is also continued
CONTINUE  ''  / over multiple records.
```

FITS reading software can reconstruct the long string value by following an inverse procedure of checking if the string value ends with the '&' character and is immediately followed by a conforming CONTINUE keyword record. If both conditions are true, then concatenate the substring from the CONTINUE record onto the previous substring after first deleting the trailing '&' character. Repeat these steps until all subsequent CONTINUE records have been processed.

Note that if a string value ends with the '&' character, but is not immediately followed by a CONTINUE keyword that conforms to all the previously described requirements, then the '&' character should be interpreted as the literal last character in the string. Also, any 'orphaned' CONTINUE keyword records (formally not invalidating the FITS file, although likely representing an error with respect to what the author of the file meant) should be interpreted as containing commentary text in bytes 9 – 80 (similar to a COMMENT keyword).

4.2.2. Logical

If the value is a fixed-format logical constant, it *shall* appear as an uppercase T or F in byte 30. A logical value is represented in free-format by a single character consisting of an uppercase T or F as the first non-space character in bytes 11 through 80.

4.4.2.4. Commentary keywords

These keywords provide commentary information about the contents or history of the *FITS* file and *may* occur any number of times in a header. These keywords *shall* have no associated value even if the value indicator characters `=␣' appear in bytes 9 and 10 (hence it is *recommended* that these keywords not contain the value indicator). Bytes 9 through 80 *may* contain any of the restricted set of ASCII text characters, decimal 32 through 126 (hexadecimal 20 through 7E).

In earlier versions of this standard continued string keywords (see 4.2.1.2) could be handled as commentary keywords if the relevant convention was not supported. Now `CONTINUE` keywords *shall* be honoured as specified in Section 4.2.1.2.

`COMMENT` keyword. This keyword *may* be used to supply any comments regarding the *FITS* file.

`HISTORY` keyword. This keyword *should* be used to describe the history of steps and procedures associated with the processing of the associated data.

Keyword field is blank. This keyword *may* be used to supply any comments regarding the *FITS* file. It is frequently used for aesthetic purposes to provide a break between groups of related keywords in the header.

DRAFT

– The last paragraph of Sect. 4.1.2.3 was corrected to state that the ASCII text characters have hexadecimal values 20 through 7E, not 41 through 7E.

H.3. List of modifications to the latest FITS standard

1. The representation of time coordinates has been incorporated by reference from Rots et al. (2015) and is summarized in Sect. 9. Cross-references have been inserted in pre-existing sections of the Standard (namely in Sect. 4.2.7, 4.3, 4.4.2.1, 4.4.2.2 and 5.4, as well as in various places of Sect. 8, like 8.3 and 8.4.1). New keywords are listed in a rearranged Table 22. Contextually an erratum was applied in Sect. 8.4.1: keywords `OBSGEO-[XYZ]` were incorrectly marked as `OBSGEO-[XYZ]`*a*; the TAI-UTC difference in Table 30 was updated with respect to Rots et al. (2015) taking into account the latest leap second; the possibility of introducing more sources for the solar system ephemerides was re-worded (at the end of Sect.9.2.5 and in Table 31).
2. The continued string keywords described in Sect. 4.2.1.2 were originally introduced as a *FITS* convention since 1994, and registered in 2007. The text of the original convention is reported at `http://fits.gsfc.nasa.gov/registry/continue_keyword.html`. The differences with this standard concern:
   – In the convention, the `LONGSTRN` keyword was used to signal the possible presence of long strings in the HDU. The use of this keyword is no longer required or recommended.
   – Usage of the convention was *not recommended* for reserved or mandatory keywords. Now is explicilty forbidded unless keywords are explicitly declared long-string.
   – To avoid ambiguities in the application of the previous clause, the declaration of string keywords in sections 8, 9 and 10 has been reset from the generic 'character' to 'string'.
   – The description of continued comment field is new.

DRAFT

Table 21: WCS and celestial coordinates notation.

Variable(s)	Meaning	Related FITS keywords
i	Index variable for world coordinates	
j	Index variable for pixel coordinates	
a	Alternative WCS version code	
p_j	Pixel coordinates	
r_j	Reference pixel coordinates	CRPIX*ja*
m_{ij}	Linear transformation matrix	CD*i_ja* or PC*i_ja*
s_i	Coordinate scales	CDELT*ia*
(x, y)	Projection plane coordinates	
(ϕ, θ)	Native longitude and latitude	
(α, δ)	Celestial longitude and latitude	
(ϕ_0, θ_0)	Native longitude and latitude of the fiducial point	PV*i*_1*a*† , PV*i*_2*a*†
(α_0, δ_0)	Celestial longitude and latitude of the fiducial point	CRVAL*ia*
(α_p, δ_p)	Celestial longitude and latitude of the native pole	
(ϕ_p, θ_p)	Native longitude and latitude of the celestial pole	LONPOLE*a* (=PV*i*_3*a*†), LATPOLE*a* (=PV*i*_4*a*†)

Notes. † Associated with *longitude* axis *i*.

in the CRPIX*i* keywords, and the world coordinates at the reference point are encoded in the CRVAL*i* keywords. For additional details, see Greisen & Calabretta (2002).

The third step of the process, computing the final world coordinates, depends on the type of coordinate system, which is indicated with the value of the CTYPE*i* keyword. For some simple, linear cases an appropriate choice of normalization for the scale factors allows the world coordinates to be taken directly (or by applying a constant offset) from the x_i (e.g., some spectra). In other cases it is more complicated, and may require the application of some non-linear algorithm (e.g., a projection, as for celestial coordinates), which may require the specification of additional parameters. Where necessary, numeric parameter values for non-linear algorithms *must* be specified via PV*i_m* keywords and character-valued parameters will be specified via PS*i_m* keywords, where *m* is the parameter number.

The application of these formalisms to coordinate systems of interest is discussed in the following sub-sections: Sect. 8.2 describes general WCS representations (see Greisen & Calabretta 2002), Sect. 8.3 describes celestial coordinate systems (see Calabretta & Greisen 2002)), Sect. 8.4 describes spectral coordinate systems (see Greisen et al. 2006), and Sect. 9 describes the representation of time coordinates (see Rots et al. 2015).

8.2. World coordinate system representations

A variety of keywords have been reserved for computing the coordinate values that are to be associated with any pixel location within an array. The full set is given in Table 22; those in most common usage are defined in detail below for convenience. Coordinate system specifications may appear in HDUs that contain simple images in the primary array or in an image extension. Images may also be stored in a multi-dimensional vector cell of a binary table, or as a tabulated list of pixel locations (and optionally, the pixel value) in a table. In these last two types of image representations, the WCS keywords have a different naming convention which reflects the needs of the tabular data structure and the 8-character limit for keyword lengths, but otherwise follow exactly the same rules for type, usage, and default values. See reference Calabretta & Greisen (2002) for example usage of these keywords. All forms of these reserved keywords *must* be used only as specified in this Standard.

In the case of the binary table vector representation, all the images contained in a given column of the table may not necessarily have the same coordinate transformation values. For example, the pixel location of the reference point may be different for each image/row in the table, in which case a single 1CRPn keyword in the header is not sufficient to record the individual value required for each image. In such cases, the keyword must be replaced by a column with the same name (i.e. TTYPEm = '1CRPn') which can then be used to store the pixel location of the reference point appropriate for each row of the table. This convention for expanding a keyword into a table column (or conversely, collapsing a column of identical values into a single header keyword) is commonly known as part of the "Green Bank Convention"[9] for *FITS* keywords. This usage is illustrated in the example header shown in Table 9 of Calabretta & Greisen (2002),

The keywords given below constitute a complete set of fundamental attributes for a WCS description. Although their inclusion in an HDU is optional, *FITS* writers *should* include a complete set of keywords when describing a WCS. In the event that some keywords are missing, default values *must* be assumed, as specified below.

WCSAXES – [integer; default: NAXIS, or larger of WCS indexes *i* or *j*]. Number of axes in the WCS description. This keyword, if present, *must* precede all WCS keywords except NAXIS in the HDU. The value of WCSAXES *may* exceed the number of pixel axes for the HDU.

CTYPE*i* – [string; indexed; default: '␣' (i.e. a linear, undefined axis)]. Type for the intermediate coordinate axis *i*. Any coordinate type that is not covered by this standard or an officially recognized *FITS* convention *shall* be taken to be linear. All non-linear coordinate system names *must* be expressed in '4–3' form: the first four characters specify the coordinate type, the fifth character is a hyphen ('-'), and the remaining three characters specify an algorithm code for computing the world coordinate value. Coordinate types with names of less than four characters are padded on the right with hyphens, and algorithm codes with less than three charac-

[9] Named after a meeting held in Green Bank, West Virginia, USA in 1989 to develop standards for the interchange of single dish radio astronomy data.

DRAFT

Table 22 (continued)

Notes. The indexes *j* and *i* are pixel and intermediate world coordinate axis numbers, respectively. Within a table, the index *n* refers to a column number, and *m* refers to a coordinate parameter number. The index *k* also refers to a column number. The indicator *a* is either blank (for the primary coordinate description) or a character A through Z that specifies the coordinate version. See text.
(1) CROTA*i* form is deprecated but still in use. It *must not* be used with PC*i_j*, PV*i_m*, and PS*i_m*. (2) PC*i_j* and CD*i_j* forms of the transformation matrix are mutually exclusive, and *must not* appear together in the same HDU. (3) EPOCH is deprecated. Use EQUINOX instead. (4) These 8-character keywords are deprecated; the 7-character forms, which can include an alternate version code letter at the end, *should* be used instead. (5) For the purpose of time reference position, geodetic latitude/longitude/elevation OBSGEO-B, OBSGEO-L, OBSGEO-H or an orbital ephemeris keyword OBSORBIT can be also used (see Sect. 9.2.3). (6) [M]JDREF can be split in integer and fractional values [M]JDREFI and [M]JDREFF as explained in Sect. 9.2.2.

ters are padded on the right with blanks[10]. Algorithm codes *should* be three characters.

CUNIT*i* – [string; indexed; default: '␣' (i.e., undefined)]. Physical units of CRVAL and CDELT for axis *i*. Note that units *should* always be specified (see Sect. 4.3). Units for celestial coordinate systems defined in this Standard *must* be degrees.

CRPIX*j* – [floating point; indexed; default: 0.0]. Location of the reference point in the image for axis *j* corresponding to r_j in Eq. (9). Note that the reference point *may* lie outside the image and that the first pixel in the image has pixel coordinates (1.0, 1.0, . . .).

CRVAL*i* – [floating point; indexed; default: 0.0]. World Coordinate value at the reference point of axis *i*.

CDELT*i* – [floating point; indexed; default: 1.0]. Increment of the world coordinate at the reference point for axis *i*. The value *must not* be zero.

CROTA*i* – [floating point; indexed; default: 0.0]. The amount of rotation from the standard coordinate system to a different coordinate system. Further use of this of this keyword is deprecated, in favor of the newer formalisms that use the CD*i_j* or PC*i_j* keywords to define the rotation.

PC*i_j* – [floating point; defaults: 1.0 when $i = j$, 0.0 otherwise]. Linear transformation matrix between pixel axes *j* and intermediate coordinate axes *i*. The PC*i_j* matrix *must not* be singular.

CD*i_j* – [floating point; defaults: 0.0, but see below]. Linear transformation matrix (with scale) between pixel axes *j* and intermediate coordinate axes *i*. This nomenclature is equivalent to PC*i_j* when CDELT*i* is unity. The CD*i_j* matrix *must not* be singular. Note that the CD*i_j* formalism is an exclusive alternative to PC*i_j*, and the CD*i_j* and PC*i_j* keywords *must not* appear together within an HDU.

In addition to the restrictions noted above, if any CD*i_j* keywords are present in the HDU, all other unspecified CD*i_j* keywords *shall* default to zero. If no CD*i_j* keywords are present then the header *shall* be interpreted as being in PC*i_j* form whether or not any PC*i_j* keywords are actually present in the HDU.

Some non-linear algorithms that describe the transformation between pixel and intermediate coordinate axes require parameter values. A few non-linear algorithms also require character-valued parameters, e.g., table lookups require the names of the table extension and the columns to be used. Where necessary parameter values *must* be specified via the following keywords:

PV*i_m* – [floating point]. Numeric parameter values for intermediate world coordinate axis *i*, where *m* is the parameter number. Leading zeros *must not* be used, and *m* may have only values in the range 0 through 99, and that are defined for the particular non-linear algorithm.

PS*i_m* – [string]. Character-valued parameters for intermediate world coordinate axis *i*, where *m* is the parameter number. Leading zeros *must not* be used, and *m* may have only values in the range 0 through 99, and that are defined for the particular non-linear algorithm.

The following keywords, while not essential for a complete specification of an image WCS, can be extremely useful for readers to interpret the accuracy of the WCS representation of the image.

CRDER*i* – [floating point; default: 0.0]. Random error in coordinate *i*, which *must* be non-negative.

CSYER*i* – [floating point; default: 0.0]. Systematic error in coordinate *i*, which *must* be non-negative.

These values *should* give a representative average value of the error over the range of the coordinate in the HDU. The total error in the coordinates would be given by summing the individual errors in quadrature.

8.2.1. Alternative WCS axis descriptions

In some cases it is useful to describe an image with more than one coordinate type[11]. Alternative WCS descriptions *may* be added to the header by adding the appropriate sets of WCS keywords, and appending to all keywords in each set an alphabetic code in the range A through Z. Keywords that may be used in this way to specify a coordinate system version are indicated in Table 22 with the suffix *a*. All implied keywords with this encoding are *reserved keywords*, and *must* only be used in *FITS* HDUs as specified in this Standard. The axis numbers *must* lie in the range 1 through 99, and the coordinate parameter *m must* lie in the range 0 through 99, both with no leading zeros.

The *primary* version of the WCS description is that specified with *a* as the blank character[12]. Alternative axis descriptions are optional, but *must not* be specified unless the primary WCS description is also specified. If an alternative WCS description is specified, all coordinate keywords for that version *must* be given even if the values do not differ from those of the primary version. Rules for the default values of alternative coordinate descriptions

[10] Example: 'RA---UV '.

[11] Examples include the frequency, velocity, and wavelength along a spectral axis (only one of which, of course, could be linear), or the position along an imaging detector in both meters and degrees on the sky.

[12] There are a number of keywords (e.g. *ij*PC*na*) where the *a* could be pushed off the 8-char keyword name for plausible values of *i*, *j*, *k*, *n*, and *m*. In such cases *a* is still said to be 'blank' although it is not the blank character.

DRAFT

Table 23: Reserved celestial coordinate algorithm codes.

Code	Default ϕ_0	Default θ_0	Properties[1]	Projection name
			Zenithal (azimuthal) projections	
AZP	0°	90°	Sect. 5.1.1	Zenithal perspective
SZP	0°	90°	Sect. 5.1.2	Slant zenithal perspective
TAN	0°	90°	Sect. 5.1.3	Gnomonic
STG	0°	90°	Sect. 5.1.4	Stereographic
SIN	0°	90°	Sect. 5.1.5	Slant orthographic
ARC	0°	90°	Sect. 5.1.6	Zenithal equidistant
ZPN	0°	90°	Sect. 5.1.7	Zenithal polynomial
ZEA	0°	90°	Sect. 5.1.8	Zenithal equal-area
AIR	0°	90°	Sect. 5.1.9	Airy
			Cylindrical projections	
CYP	0°	0°	Sect. 5.2.1.	Cylindrical perspective
CEA	0°	0°	Sect. 5.2.2	Cylindrical equal area
CAR	0°	0°	Sect. 5.2.3	Plate carrée
MER	0°	0°	Sect. 5.2.4	Mercator
			Pseudo-cylindrical and related projections	
SFL	0°	0°	Sect. 5.3.1	Samson-Flamsteed
PAR	0°	0°	Sect. 5.3.2	Parabolic
MOL	0°	0°	Sect. 5.3.3	Mollweide
AIT	0°	0°	Sect. 5.3.4	Hammer-Aitoff
			Conic projections	
COP	0°	θ_a	Sect. 5.4.1	Conic perspective
COE	0°	θ_a	Sect. 5.4.2	Conic equal-area
COD	0°	θ_a	Sect. 5.4.3	Conic equidistant
COO	0°	θ_a	Sect. 5.4.4	Conic orthomorphic
			Polyconic and pseudoconic projections	
BON	0°	0°	Sect. 5.5.1	Bonne's equal area
PCO	0°	0°	Sect. 5.5.2	Polyconic
			Quad-cube projections	
TSC	0°	0°	Sect. 5.6.1	Tangential spherical cube
CSC	0°	0°	Sect. 5.6.2	COBE quadrilateralized spherical cube
QSC	0°	0°	Sect. 5.6.3	Quadrilateralized spherical cube
			HEALPix grid projection	
HPX	0°	0°	Sect. 6[2]	HEALPix grid

[1] Refer to the indicated section in Calabretta & Greisen (2002) for a detailed description. [2] This projection is defined in Calabretta & Roukema (2007).

are the same as those for the primary description. The alternative descriptions are computed in the same fashion as the primary coordinates. The type of coordinate depends on the value of CTYPE*ia*, and may be linear in one of the alternative descriptions and non-linear in another.

The alternative version codes are selected by the *FITS* writer; there is no requirement that the codes be used in alphabetic sequence, nor that one coordinate version differ in its parameter values from another. An optional keyword WCSNAME*a* is also defined to name, and otherwise document, the various versions of WCS descriptions:

WCSNAME*a* – [string; default for *a*: ' ' (i.e., blank, for the primary WCS, else a character A through Z that specifies the coordinate version]. Name of the world coordinate system represented by the WCS keywords with the suffix *a*. Its primary function is to provide a means by which to specify a particular WCS if multiple versions are defined in the HDU.

8.3. Celestial coordinate system representations

The conversion from intermediate world coordinates (x, y) in the plane of projection to celestial coordinates involves two steps: a spherical projection to native longitude and latitude (ϕ, θ), defined in terms of a convenient coordinate system (i.e., *native spherical coordinates*), followed by a spherical rotation of these native coordinates to the required celestial coordinate system (α, δ). The algorithm to be used to define the spherical projection *must* be encoded in the CTYPE*i* keyword as the three-letter algorithm code, the allowed values for which are specified in Table 23 and defined in references Calabretta & Greisen (2002) and Calabretta & Roukema (2007). The target celestial coordinate system is also encoded into the left-most portion of the CTYPE*i* keyword as the coordinate type.

For the final step, the parameter LONPOLE*a* must be specified, which is the native longitude of the celestial pole, ϕ_p. For certain projections (such as cylindricals and conics, which are less commonly used in astronomy), the additional keyword LATPOLE*a* must be used to specify the native latitude of the celestial pole. See Calabretta & Greisen (2002) for the transformation equations and other details.

The accepted celestial coordinate systems are: the standard equatorial (RA-- and DEC-), and others of the form *x*LON and *x*LAT for longitude-latitude pairs, where *x* is G for Galactic, E for ecliptic, H for helioecliptic and S for supergalactic coordinates. Since the representation of planetary, lunar, and solar coordinate

systems could exceed the 26 possibilities afforded by the single character *x*, pairs of the form *yz*LN and *yz*LT *may* be used as well.

RADESYS*a* – [string; default: FK4, FK5, or ICRS: see below]. Name of the reference frame of equatorial or ecliptic coordinates, whose value *must* be one of those specified in Table 24. The default value is FK4 if the value of EQUINOX*a* < 1984.0, FK5 if EQUINOX*a* ≥ 1984.0, or ICRS if EQUINOX*a* is not given.

EQUINOX*a* – [floating point; default: see below]. Epoch of the mean equator and equinox in years, whose value *must* be non-negative. The interpretation of epoch depends upon the value of RADESYS*a* if present: *Besselian* if the value is FK4 or FK4-NO-E, *Julian* if the value is FK5; *not applicable* if the value is ICRS or GAPPT.

EPOCH – [floating point]. This keyword is deprecated and *should not* be used in new *FITS* files. It is reserved primarily to prevent its use with other meanings. The EQUINOX keyword *shall* be used instead. The value field of this keyword was previously defined to contain a floating-point number giving the equinox in years for the celestial coordinate system in which positions are expressed.

DATE-OBS – [floating point]. This reserved keyword is defined in Sect. 4.4.2.

MJD-OBS – [floating point; default: DATE-OBS if given, otherwise no default]. Modified Julian Date (JD – 2,400,000.5) of the observation, whose value corresponds (by default) to the *start* of the observation, unless another interpretation is explained in the comment field. No specific time system (e.g. UTC, TAI, etc.) is defined for this or any of the other time-related keywords. It is *recommended* that the TIMESYS keyword, as defined in Sect. 9.2.1 be used to specify the time system. See also Sect. 9.5.

LONPOLE*a* – [floating point; default: ϕ_0 if $\delta_0 \geq \theta_0$, $\phi_0 + 180°$ otherwise]. Longitude in the native coordinate system of the celestial system's north pole. Normally, ϕ_0 is zero unless a non-zero value has been set for PV*i*_1*a*, which is associated with the *longitude* axis. This default applies for all values of θ_0, including $\theta_0 = 90°$, although the use of non-zero values of θ_0 are discouraged in that case.

LATPOLE*a* – [floating point; default: 90°, or no default if $(\theta_0, \delta_0, \phi_p - \phi_0) = (0, 0, \pm 90°)$]. Latitude in the native coordinate system of the celestial system's north pole, or equivalently, the latitude in the celestial coordinate system of the native system's north pole. May be ignored or omitted in cases where LONPOLE*a* completely specifies the rotation to the target celestial system.

8.4. Spectral coordinate system representations

This section discusses the conversion of intermediate world coordinates to spectral coordinates with common axes such as frequency, wavelength, and apparent radial velocity (represented here with the coordinate variables ν, λ, or v). The key point for constructing spectral WCS in *FITS* is that one of these coordinates *must* be sampled linearly in the dispersion axis; the others are derived from prescribed, usually non-linear transformations. Frequency and wavelength axes *may* also be sampled linearly in their logarithm.

Following the convention for the CTYPE*ia* keyword, when *i* is the spectral axis the first four characters *must* specify a code for the coordinate type; for non-linear algorithms the fifth character

Table 24: Allowed values of RADESYS*a*.

Value	Definition
ICRS	International Celestial Reference System
FK5	Mean place, new (IAU 1984) system
FK4[1]	Mean place, old (Bessel-Newcomb) system
FK4-NO-E[1]	Mean place: but without eccentricity terms
GAPPT	Geocentric apparent place, IAU 1984 system

(1) New FITS files should avoid using these older reference systems.

must be a hyphen, and the next three characters *must* specify a predefined algorithm for computing the world coordinates from the intermediate physical coordinates. The coordinate type *must* be one of those specified in Table 25. When the algorithm is linear, the remainder of the CTYPE*ia* keyword *must* be blank. When the algorithm is non-linear, the 3-letter algorithm code *must* be one of those specified in Table 26. The relationships between the basic physical quantities ν, λ, and v, as well as the relationships between various derived quantities are given in reference Greisen et al. (2006).

The generality of the algorithm for specifying the spectral coordinate system and its representation suggests that some additional description of the coordinate may be helpful beyond what can be encoded in the first four characters of the CTYPE*ia* keyword; CNAME*ia* is reserved for this purpose. Note that this keyword provides a name for an axis in a particular WCS, while the WCSNAME*a* keyword names the particular WCS as a whole. In order to convert between some form of radial velocity and either frequency or wavelength, the keywords RESTFRQ*a* and RESTWAV*a*, respectively, are reserved.

CNAME*ia* – [string; default: default: '␣' (i.e. a linear, undefined axis)]. Spectral coordinate description which *must not* exceed 68 characters in length.

RESTFRQ*a* – [floating point; default: none]. Rest frequency of the of the spectral feature of interest. The physical unit *must* be Hz.

RESTWAV*a* – [floating point; default: none]. Vacuum rest wavelength of the of the spectral feature of interest. The physical unit *must* be m.

One or the other of RESTFRQ*a* or RESTWAV*a* *should* be given when it is meaningful to do so.

8.4.1. Spectral coordinate reference frames

Frequencies, wavelengths, and apparent radial velocities are always referred to some selected standard of rest (i.e., reference frame). While the spectra are obtained they are, of necessity, in the observer's rest frame. The velocity correction from topocentric (the frame in which the measurements are usually made) to standard reference frames (which *must* be one of those given in Table 27) are dependent on the dot product with time-variable velocity vectors. That is, the velocity with respect to a standard reference frame depends upon direction, and the velocity (and frequency and wavelength) with respect to the local standard of rest is a function of the celestial coordinate within the image. The keywords SPECSYS*a* and SSYSOBS*a* are reserved and, if used, *must* describe the reference frame in use for the spectral axis coordinate(s) and the spectral reference frame that was held constant during the observation, respectively. In order to compute the velocities it is necessary to have the date and time of the

DRAFT

Table 25: Reserved spectral coordinate type codes.

Code[1]	Type	Symbol	Assoc. variable	Default units
FREQ	Frequency	ν	ν	Hz
ENER	Energy	E	ν	J
WAVN	Wavenumber	κ	ν	m^{-1}
VRAD	Radio velocity[2]	V	ν	m s^{-1}
WAVE	Vacuum wavelength	λ	λ	m
VOPT	Optical velocity[2]	Z	λ	m s^{-1}
ZOPT	Redshift	z	λ	...
AWAV	Air wavelength	λ_a	λ_a	m
VELO	Apparent radial velocity	v	v	m s^{-1}
BETA	Beta factor (v/c)	β	v	...

(1) Characters 1 through 4 of the value of the keyword CTYPE*ia*. (2) By convention, the 'radio' velocity is given by $c(\nu_0 - \nu)/\nu_0$ and the 'optical' velocity is given by $c(\lambda - \lambda_0)/\lambda_0$.

observation; the keywords DATE-AVG and MJD-AVG are reserved for this purpose. See also Sect. 9.5.

DATE-AVG – [string; default: none]. Calendar date of the mid-point of the observation, expressed in the same way as the DATE-OBS keyword.

MJD-AVG – [floating point; default: none]. Modified Julian Date (JD – 2,400,000.5) of the mid-point of the observation.

SPECSYS*a* – [string; default: none]. The reference frame in use for the spectral axis coordinate(s). Valid values are given in Table 27.

SSYSOBS*a* – [string; default: TOPOCENT]. The spectral reference frame that is constant over the range of the non-spectral world coordinates. Valid values are given in Table 27.

The transformation from the rest frame of the observer to a standard reference frame requires a specification of the location on Earth[13] of the instrument used for the observation in order to calculate the diurnal Doppler correction due to the Earth's rotation. The location, if specified, *shall* be represented as a geocentric Cartesian triple with respect to a standard ellipsoidal geoid at the time of the observation. While the position can often be specified with an accuracy of a meter or better, for most purposes positional errors of several kilometers will have negligible impact on the computed velocity correction. For details, see reference Greisen et al. (2006).

OBSGEO-X – [floating point; default: none]. $X-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

OBSGEO-Y – [floating point; default: none]. $Y-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

OBSGEO-Z – [floating point; default: none]. $Z-$coordinate (in meters) of a Cartesian triplet that specifies the location, with respect to a standard, geocentric terrestrial reference frame, where the observation took place. The coordinate *must* be valid at the epoch MJD-AVG or DATE-AVG.

[13] The specification of location for an instrument on a spacecraft in flight requires an ephemeris; keywords that might be required in this circumstance are not defined here.

Table 26: Non-linear spectral algorithm codes.

Code[1]	Regularly sampled in	Expressed as
F2W	Frequency	Wavelength
F2V		Apparent radial velocity
F2A		Air wavelength
W2F	Wavelength	Frequency
W2V		Apparent radial velocity
W2A		Air wavelength
V2F	Apparent radial vel.	Frequency
V2W		Wavelength
V2A		Air wavelength
A2F	Air wavelength	Frequency
A2W		Wavelength
A2V		Apparent radial velocity
LOG	Logarithm	Any four-letter type code
GRI	Detector	Any type code from Table 25
GRA	Detector	Any type code from Table 25
TAB	Not regular	Any four-letter type code

(1) Characters 6 through 8 of the value of the keyword CTYPE*ia*.

Information on the relative radial velocity between the observer and the selected standard of rest in the direction of the celestial reference coordinate *may* be provided, and if so *shall* be given by the VELOSYS*a* keyword. The frame of rest defined with respect to the emitting source may be represented in *FITS*; for this reference frame it is necessary to define the velocity with respect to some other frame of rest. The keywords SPECSYS*a* and ZSOURCE*a* are used to document the choice of reference frame and the value of the systemic velocity of the source, respectively.

SSYSSRC*a* – [string; default: none]. Reference frame for the value expressed in the ZSOURCE*a* keyword to document the systemic velocity of the observed source. Value *must* be one of those given in Table 27 *except* for SOURCE.

VELOSYS*a* – [floating point; default: none]. Relative radial velocity between the observer and the selected standard of rest in the direction of the celestial reference coordinate. Units *must* be m s^{-1}. The CUNIT*ia* keyword is not used for this purpose since the WCS version *a* might not be expressed in velocity units.

ZSOURCE*a* – [floating point; default: none]. Radial velocity with respect to an alternative frame of rest, expressed as a unitless redshift (i.e., velocity as a fraction of the speed of light in vacuum). Used in conjunction with SSYSSRC*a* to document the systemic velocity of the observed source.

DRAFT

All of the time part *may* be omitted (just leaving the date) or the decimal seconds *may* be omitted. Leading zeroes *must not* be omitted and timezone designators are *not allowed*. This definition is extended to allow five-digit years with a *mandatory* sign, in accordance with ISO-8601. That is, one *shall* use either the *unsigned* four-digit year format or the *signed* five-digit year format:

`[±C]CCYY-MM-DD[Thh:mm:ss[.s...]]`

Note the following:

- In counting years, ISO-8601 follows the convention of including year zero. Consequently, for negative year numbers there is an offset of one from BCE dates which do not recognize a year zero. Thus year 1 corresponds to 1 CE, year 0 to 1 BCE, year −1 to 2 BCE, and so on.
- The earliest date that may be represented in the four-digit year format is `0000-01-01T00:00:00` (in the year 1 BCE); the latest date is `9999-12-31T23:59:59`. This representation of time is tied to the Gregorian calendar. In conformance with the present ISO-8601:2004(E) standard (ISO 2004b) dates prior to 1582 *must* be interpreted according to the proleptic application of the rules of Gregorius XIII. For dates not covered by that range the use of Modified Julian Date (MJD) or Julian Date (JD) numbers or the use of the signed five-digit year format is *recommended*.
- In the five-digit year format the earliest and latest dates are `-99999-01-01T00:00:00` (i.e., −100 000 BCE) and `+99999-12-31T23:59:59`.
- The origin of JD can be written as: `-04713-11-24T12:00:00`.
- In time scale UTC the integer part of the seconds field runs from 00 to 60 (in order to accommodate leap seconds); in all other time scales the range is 00 to 59.
- The ISO-8601 *datetime* data type is *not allowed* in image axis descriptions since `CRVAL` is required to be a floating point value.
- ISO-8601 *datetime* does not imply the use of any particular time scale (see Section 9.2.1).
- As specified by Bunclark & Rots (1997), time zones are explicitly not supported in FITS and, consequently, appending the letter `'Z'` to a FITS ISO-8601 string is *not allowed*. The rationale for this rule is that its role in the ISO standard is that of a time zone indicator, not a time scale indicator. As the concept of a time zone is not supported in FITS, the use of time zone indicator is inappropriate.

9.1.2. Julian and Besselian epochs

In a variety of contexts *epochs* are provided with astronomical data. Until 1976 these were commonly based on the Besselian year (see Sect. 9.3), with standard epochs B1900.0 and B1950.0. After 1976 the transition was made to Julian epochs based on the Julian year of 365.25 days, with the standard epoch J2000.0. They are tied to time scales ET and TDB, respectively. Note that the Besselian epochs are scaled by the variable length of the Besselian year (see Sect. 9.3 and its cautionary note, which also applies to this context). The Julian epochs are easier to calculate, as long as one keeps track of leap days.

9.2. Time coordinate frame

9.2.1. Time scale

The *time scale* defines the temporal reference frame, and is specified in the header in one of a few ways, depending upon the context. When recorded as a global keyword, the time scale *shall* be specified by:

`TIMESYS` – [string; default: `'UTC'`]. The value field of this keyword *shall* contain a character string code for the time scale of the time-related keywords. The *recommended* values for this keyword in Table 30 have well defined meanings, but other values *may* be used. If this keyword is absent, `'UTC'` *must* be assumed.

In relevant contexts (e.g., time axes in image arrays, table columns, or random groups) `TIMESYS` *may* be overridden by a time scale recorded in `CTYPEia`, its binary table equivalents, or `PTYPEi` (see Table 22).

The keywords `TIMESYS`, `CTYPEia`, `TCTYPn`, and `TCTYna` or binary table equivalent *may* assume the values listed in Table 30. In addition, for backward compatibility, all except `TIMESYS` and `PTYPEi` *may* also assume the value `TIME` (case-insensitive), whereupon the time scale *shall* be that recorded in `TIMESYS` or, in its absence, its default value, `UTC`. As noted above, local time scales other than those listed in Table 30 *may* be used, but their use *should* be restricted to alternate coordinates in order that the primary coordinates will always refer to a properly recognized time scale.

See Rots et al. (2015), Appendix A, for a detailed discussion of the various time scales. In cases where high-precision timing is important one may append a specific realization, in parentheses, to the values in the table; e.g., `TT(TAI)`, `TT(BIPM08)`, `UTC(NIST)`. Note that linearity is not preserved across all time scales. Specifically, if the reference position remains unchanged (see Section 9.2.3), the first ten, with the exception of `UT1`, are linear transformations of each other (excepting leap seconds), as are `TDB` and `TCB`. On average `TCB` runs faster than `TCG` by approximately 1.6×10^{-8}, but the transformation from `TT` or `TCG` (which are linearly related) is to be achieved through a time ephemeris as provided by Irwin & Fukushima (1999).

The relations between coordinate time scales and their dynamical equivalents have been defined as:

$T(\mathrm{TCG}) = T(\mathrm{TT}) + L_G \times 86400 \times (JD(\mathrm{TT}) - JD_0)$

$T(\mathrm{TDB}) = T(\mathrm{TCB}) - L_B \times 86400 \times (JD(\mathrm{TCB}) - JD_0) + TDB_0$

where:

T is in seconds

$L_G = 6.969290134 \times 10^{-10}$

$L_B = 1.550519768 \times 10^{-8}$

$JD_0 = 2443144.5003725$

$TDB_0 = -6.55 \times 10^{-5}$ s

Linearity is virtually guaranteed since images and individual table columns do not allow more than one reference position to be associated with them, and since there is no overlap between reference positions that are meaningful for the first nine time scales on the one hand, and for the barycentric ones on the other. All use of the time scale GMT in FITS files *shall* be taken to have its zero point at midnight, conformant with UT, including dates prior to 1925. For high-precision timing prior to 1972, see Rots et al. (2015), Appendix A.

Some time scales in use are not listed in Table 30 because they are intrinsically unreliable or ill-defined. When used, they

TREFPOS – [string; default: TOPOCENTER]. The value field of this keyword *shall* contain a character string code for the spatial location at which the observation time is valid. The value *should* be one of those given in Table 31. This keyword *shall* apply to time coordinate axes in images as well.

In binary tables different columns *may* represent completely different Time Coordinate Frames. However, each column can have only one time reference position, thus guaranteeing linearity (see Section 9.2.1).

TRPOS*n* – [string; default: TOPOCENTER] The value field of this keyword *shall* contain a character string code for the spatial location at which the observation time is valid. This table keyword *shall* override TREFPOS.

The reference position value *may* be a standard location (such as GEOCENTER or TOPOCENTER) or a point in space defined by specific coordinates. In the latter case one should be aware that a (3-D) spatial coordinate frame needs to be defined that is likely to be different from the frame(s) that the data are associated with. Note that TOPOCENTER is only moderately informative if no observatory location is provided or indicated. The commonly allowed standard values are shown in Table 31. Note that for the gaseous planets the barycenters of their planetary systems, including satellites, are used for obvious reasons. While it is preferable to spell the location names out in full, in order to be consistent with the practice of Greisen et al. (2006) the values are allowed to be truncated to eight characters. Furthermore, in order to allow for alternative spellings, only the first three characters of all these values *shall* be considered significant. The value of the keyword *shall* be case-sensitive.

Table 31: Standard Time Reference Position Values

Value[1]	Meaning
TOPOCENTER	Topocenter: the location from where the observation was made (default)
GEOCENTER	Geocenter
BARYCENTER	Barycenter of the Solar System
RELOCATABLE	Relocatable: to be used for simulation data only
CUSTOM	A position specified by coordinates that is not the observatory location
Less common, but allowed standard values are:	
HELIOCENTER	Heliocenter
GALACTIC	Galactic center
EMBARYCENTER	Earth-Moon barycenter
MERCURY	Center of Mercury
VENUS	Center of Venus
MARS	Center of Mars
JUPITER	Barycenter of the Jupiter system
SATURN	Barycenter of the Saturn system
URANUS	Barycenter of the Uranus system
NEPTUNE	Barycenter of the Neptune system

Notes. (1)Recognized values for TREFPOS, TRPOS*n*; only the first three characters of the values are significant and solar system locations are as specified in the ephemerides.

The reader is cautioned that time scales and reference positions cannot be combined arbitrarily if one wants a clock that runs linearly at TREFPOS. Table 32 provides a summary of compatible combinations. BARYCENTER *should* only be used in conjunction with time scales TDB and TCB and *should* be the only reference position used with these time scales. With proper care GEOCENTER, TOPOCENTER, and EMBARYCENTER are appropriate for the first ten time scales in Table 30. However, relativistic effects introduce a (generally linear) scaling in certain combinations; highly eccentric spacecraft orbits are the exceptions. Problems will arise when using a reference position on another solar system body (including HELIOCENTER). Therefore it is *recommended* to synchronize the local clock with one of the time scales defined on the Earth's surface, TT, TAI, GPS, or UTC (in the last case: beware of leap seconds). This is common practice for spacecraft clocks. Locally, such a clock will not appear to run at a constant rate, because of variations in the gravitational potential and in motions with respect to Earth, but the effects can be calculated and are probably small compared with errors introduced by the alternative: establishing a local time standard.

Table 32: Compatibility of Time Scales and Reference Positions

Reference Position	Time scale[1] TT, TDT TAI, IAT GPS UTC, GMT	TCG	TDB	TCB	LOCAL
TOPOCENTER	t	ls			
GEOCENTER	ls	c			
BARYCENTER			ls	c	
RELOCATABLE					c
Other[2]	re	re			

Notes. (1)Legend (combination is not recommended if no entry); **c**: correct match; reference position coincides with the spatial origin of the space-time coordinates; **t**: correct match on Earth's surface, otherwise usually linear scaling; **ls**: linear relativistic scaling; **re**: non-linear relativistic scaling. (2)All other locations in the solar system.

In order to provide a complete description, TOPOCENTER requires the observatory's coordinates to be specified. There are three options: *(a)* the ITRS Cartesian coordinates defined in Sect. 8.4.1 (OBSGEO-X, OBSGEO-Y, OBSGEO-Z), which are *strongly preferred*; *(b)* a geodetic latitude/longitude/elevation triplet (defined below); or *(c)* a reference to an orbit ephemeris file. A set of geodetic coordinates is recognized:

OBSGEO-B – [floating-point] The value field of this keyword *shall* contain the latitude of the observation in deg, with North positive.

OBSGEO-L – [floating-point] The value field of this keyword *shall* contain the longitude of the observation in deg, with East positive.

OBSGEO-H – [floating-point] The value field of this keyword *shall* contain the altitude of the observation in meters.

An orbital ephemeris file can instead be specified:

OBSORBIT – [string] The value field of this keyword *shall* contain the character-string URI, URL, or the name of an orbit ephemeris file.

Beware that only one set of coordinates is allowed in a given HDU. Cartesian ITRS coordinates are the preferred coordinate

DRAFT

system; however, when using these in an environment requiring nanosecond accuracy, one should take care to distinguish between meters consistent with TCG or with TT. If one uses geodetic coordinates, the geodetic altitude OBSGEO-H is measured with respect to the IAU 1976 ellipsoid which is defined as having a semi-major axis of 6 378 140 m and an inverse flattening of 298.2577.

A non-standard location indicated by CUSTOM *must* be specified in a manner similar to the specification of the observatory location (indicated by TOPOCENTER). One should be careful with the use of the CUSTOM value and not confuse it with TOPOCENTER, as use of the latter imparts additional information on the provenance of the data.

ITRS coordinates (X,Y,Z) may be derived from geodetic coordinates (L,B,H) through:

$$X = (N(B) + H)\cos(L)\cos(B)$$

$$Y = (N(B) + H)\sin(L)\cos(B)$$

$$Z = (N(B)(1 - e^2) + H)\sin(B)$$

where:

$$N(B) = \frac{a}{\sqrt{1 - e^2\sin^2(B)}}$$

$$e^2 = 2f - f^2$$

a is the semi-major axis, and f is the inverse of the inverse flattening. Nanosecond precision in timing requires that OBSGEO-[BLH] be expressed in a geodetic reference frame defined after 1984 in order to be sufficiently accurate.

9.2.4. Time reference direction

If any pathlength corrections have been applied to the time stamps (i.e., if the reference position is not TOPOCENTER for observational data), the reference direction that is used in calculating the pathlength delay *should* be provided in order to maintain a proper analysis trail of the data. However, this is useful only if there is also information available on the location from where the observation was made (the observatory location). The direction will usually be provided in a spatial coordinate frame that is already being used for the spatial metadata, although it is conceivable that multiple spatial frames are involved, e.g., spherical ICRS coordinates for celestial positions, and Cartesian FK5 for spacecraft ephemeris. The time reference direction does not by itself provide sufficient information to perform a fully correct transformation; however, within the context of a specific analysis environment it should suffice.

The uncertainty in the reference direction affects the errors in the time stamps. A typical example is provided by barycentric corrections where the time error is related to the position error:

$$t_{err}(\text{ms}) \leq 2.4 pos_{err}(\text{arcsec})$$

The reference direction is indicated through a reference to specific keywords. These keywords *may* hold the reference direction explicitly or (for data in BINTABLEs) indicate columns holding the coordinates. In event lists where the individual photons are tagged with a spatial position, those coordinates *may* have been used for the reference direction and the reference will point to the columns containing these coordinate values. The time reference direction *shall* be specified by the keyword:

TREFDIR – [string] The value field of this keyword *shall* contain a character string composed of: the name of the keyword containing the longitudinal coordinate, followed by a comma, followed by the name of the keyword containing the latitudinal coordinate. This reference direction *shall* apply to time coordinate axes in images as well.

In binary tables different columns *may* represent completely different Time Coordinate Frames. However, also in that situation the condition holds that each column can have only one Time Reference Direction. Hence, the following keyword may override TREFDIR:

TRDIR*n* – [string] The value field of this keyword *shall* contain a character string consisting of the name of the keyword or column containing the longitudinal coordinate, followed by a comma, followed by the name of the keyword or column containing the latitudinal coordinate. This reference direction *shall* apply to time coordinate axes in images as well.

9.2.5. Solar System Ephemeris

If applicable, the Solar System ephemeris used for calculating pathlength delays *should* be identified. This is particularly pertinent when the time scale is TCB or TDB. The ephemerides that are currently most often used are those from JPL (2014a,b).

The Solar System ephemeris used for the data (if required) *shall* be indicated by the following keyword:

PLEPHEM – [string; default: 'DE405'] The value field of this keyword *shall* contain a character string that *should* represent a recognized designation for the Solar System ephemeris. Recognized designations for JPL Solar System ephemerides that are often used are listed in Table 33.

Table 33: Valid solar system ephemerides

Value	Reference
'DE200'	Standish (1990); considered obsolete, but still in use
'DE405'	Standish (1998); default
'DE421'	Folkner, et al. (2009)
'DE430'	Folkner, et al. (2014)
'DE431'	Folkner, et al. (2014)
'DE432'	Folkner, et al. (2014)

Future ephemerides in this series *shall* be accepted and recognized as they are released. Additional ephemerides designations may be recognized by the IAUFWG upon request.

9.3. Time unit

When recorded as a global keyword, the unit used to express time *shall* be specified by:

TIMEUNIT – [string; default: 's'] The value field of this keyword *shall* contain a character string that specifies the time unit; the value *should* be one of those given in Table 34. This time unit *shall* apply to all time instances and durations that do not have an implied time unit (such as is the case for JD, MJD, ISO-8601, J and B epochs). If this keyword is absent, 's' *shall* be assumed.

DRAFT

10.1. Tiled Image Compression

The following describes the process for compressing n–dimensional FITS images and storing the resulting byte stream in a variable-length column in a FITS binary table, and for preserving the image header keywords in the table header. The general principle is to first divide the n–dimensional image into a rectangular grid of subimages or "tiles." Each tile is then compressed as a block of data, and the resulting compressed byte stream is stored in a row of a variable length column in a FITS binary table (see Section 7.3). By dividing the image into tiles it is possible to extract and uncompress subsections of the image without having to uncompress the whole image. The default tiling pattern treats each row of a 2-dimensional image (or higher dimensional cube) as a tile, such that each tile contains NAXIS1 pixels. This default may not be optimal for some applications or compression algorithms, so any other rectangular tiling pattern may be defined using keywords that are defined below. In the case of relatively small images it may suffice to compress the entire image as a single tile, resulting in an output binary table with a single row. In the case of 3-dimensional data cubes, it may be advantageous to treat each plane of the cube as a separate tile if application software typically needs to access the cube on a plane-by-plane basis.

10.1.1. Required Keywords

In addition to the mandatory keywords for BINTABLE extensions (see Sect. 7.3.1) the following keywords are reserved for use in the header of a FITS binary table extension to describe the structure of a valid compressed FITS image. All are mandatory.

ZIMAGE – [logical; value 'T'] The value field of this keyword *shall* contain the logical value 'T' to indicate that the FITS binary table extension contains a compressed image, and that logically this extension *should* be interpreted as an image rather than a table.

ZCMPTYPE – [string; default: none] The value field of this keyword *shall* contain a character string giving the name of the algorithm that was used to compress the image. Only the values given in Table 36 are permitted; the corresponding algorithms are described in Sect. 10.4. Other algorithms may be added in the future.

ZBITPIX – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the BITPIX keyword in the uncompressed FITS image.

ZNAXIS – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the NAXIS keyword (i.e., the number of axes) in the uncompressed FITS image.

ZNAXIS*n* – [integer; indexed; default: none) The value field of these keywords *shall* contain a positive integer that gives the value of the corresponding NAXIS*n* keywords (i.e., the size of axis *n*) in the uncompressed FITS image.

The comment fields for the BITPIX, NAXIS, and NAXIS*n* keywords in the uncompressed image *should* be copied to the corresponding fields in the ZBITPIX, ZNAXIS, and ZNAXIS*n* keywords.

10.1.2. Other Reserved Keywords

The compressed image tiles *must* be stored in the binary table in the same order that the first pixel in each tile appears in the FITS image; the tile containing the first pixel in the image *must* appear in the first row of the table, and the tile containing the last pixel in the image *must* appear in the last row of the binary table. The following keywords are reserved for use in describing compressed images stored in BINTABLE extensions; they *may* be present in the header, and their values depend upon the type of image compression employed.

ZTILE*n* – [integer; indexed; default: 1 for $n > 1$] The value field of these keywords (where n is a positive integer index that ranges from 1 to ZNAXIS) *shall* contain a positive integer representing the number of pixels along axis n of the compressed tiles. Each tile of pixels *must* be compressed separately and stored in a row of a variable-length vector column in the binary table. The size of each image dimension (given by ZNAXIS*n*) need not be an integer multiple of ZTILE*n*, and if it is not, then the last tile along that dimension of the image will contain fewer image pixels than the other tiles. If the ZTILE*n* keywords are not present then the default "row-by-row" tiling will be assumed, i.e., ZTILE1 = ZNAXIS1, and the value of all the other ZTILE*n* keywords *must* equal 1.

ZNAME*i* – [string; indexed; default: none] The value field of these keywords (where i is a positive integer index starting with 1) *shall* supply the names of up to 999 algorithm-specific parameters that are needed to compress or uncompress the image. The order of the compression parameters *may* be significant, and *may* be defined as part of the description of the specific decompression algorithm.

ZVAL*i* – [string; indexed; default: none] The value field of these keywords (where i is a positive integer index starting with 1) *shall* contain the values of up to 999 algorithm-specific parameters with the same index i. The value of ZVAL*i* may have any valid FITS data type.

ZMASKCMP – [string; default: none] The value field of this keyword *shall* contain the name of the image compression algorithm that was used to compress the optional null-pixel data mask. This keyword may be omitted if no null-pixel data masks appear in the table. See Sect. 10.2.2 for details.

ZQUANTIZ – [string; default: 'NO_DITHER'] The value field of this keyword *shall* contain the name of the algorithm that was used to quantize floating-point image pixels into integer values, which were then passed to the compression algorithm as discussed further in Sect. 10.2. If this keyword is not present, the default is to assume that no dithering was applied during quantization.

ZDITHER0 – [integer; default: none] The value field of this keyword *shall* contain a positive integer (that may range from 1 to 10000 inclusive) that gives the seed value for the random dithering pattern that was used when quantizing the floating-point pixel values. This keyword *may* be absent if no dithering was applied. See Sect. 10.2 for further discussion.

The following keywords are reserved to preserve a verbatim copy of the *value and comment fields* for keywords in the original uncompressed FITS image that were used to describe its structure. These optional keywords, when present, *shall* be used when reconstructing an identical copy of the original FITS HDU

DRAFT

of the uncompressed image. They *should not* appear in the compressed image header unless the corresponding keywords were present in the uncompressed image.

ZSIMPLE – [logical; value 'T'] The value field of this keyword *must* contain the value of the original SIMPLE keyword in the uncompressed image.

ZEXTEND – [string] The value field of this keyword *must* contain the value of the original EXTEND keyword in the uncompressed image.

ZBLOCKED – [logical] The value field of this keyword *must* contain the value of the original BLOCKED keyword in the uncompressed image.

ZTENSION – [string] The value field of this keyword *must* contain the original XTENSION keyword in the uncompressed image.

ZPCOUNT – [integer] The value field of this keyword *must* contain the original PCOUNT keyword in the uncompressed image.

ZGCOUNT – [integer] The value field of this keyword *must* contain the original GCOUNT keyword in the uncompressed image.

ZHECKSUM – [string] The value field of this keyword *must* contain the original CHECKSUM keyword (see Sect. 4.4.2.7) in the uncompressed image.

ZDATASUM – [string] The value field of this keyword *must* contain the original DATASUM keyword (see Sect. 4.4.2.7) in the uncompressed image.

The ZSIMPLE, ZEXTEND, and ZBLOCKED keywords *must not* be used unless the original uncompressed image was contained in the primary array of a FITS file. The ZTENSION, ZPCOUNT, and ZGCOUNT keywords *must not* be used unless the original uncompressed image was contained in an IMAGE extension.

The FITS header of the compressed image *may* contain other keywords. If a FITS primary array or IMAGE extension is compressed using the procedure described here, it is *strongly recommended* that all the keywords (including comment fields) in the header of the original image, except for the mandatory keywords mentioned above, be copied verbatim and in the same order into the header of the binary table extension that contains the compressed image. All these keywords will have the same meaning and interpretation as they did in the original image, even in cases where the keyword is not normally expected to occur in the header of a binary table extension (e.g., the BSCALE and BZERO keywords, or the World Coordinate System keywords such as CTYPE*n*, CRPIX*n* and CRVAL*n*).

10.1.3. Table Columns

Two columns in the FITS binary table are defined below to contain the compressed image tiles; the order of the columns in the table is not significant. One of the table columns describes optional content; but when this column appears it *must* be used as defined in this section. The column names (given by the TTYPE*n* keyword) are reserved; they are shown here in upper case letters, but case is not significant.

COMPRESSED_DATA – [required; variable-length] Each row of this column *must* contain the byte stream that is generated as a result of compressing the corresponding image tile. The data type of the column (as given by the TFORM*n* keyword) *must* be one of '1PB', '1PI', or '1PJ' (or the equivalent '1QB', '1QI', or '1QJ'), depending on whether the compression algorithm generates an output stream of 8-bit bytes, or integers of 16-, or 32-bits.

When using the quantization method to compress floating-point images that is described in Sect. 10.2, it sometimes may not be possible to quantize some of the tiles (e.g., if the range of pixels values is too large or if most of the pixels have the same value and hence the calculated RMS noise level in the tile is close to zero). There also may be other rare cases where the nominal compression algorithm cannot be applied to certain tiles. In these cases, an alternate technique *may* be used in which the raw pixel values are losslessly compressed with the GZIP algorithm.

GZIP_COMPRESSED_DATA [optional; variable-length] If the raw pixel values in an image tile are losslessly compressed with the GZIP algorithm, the resulting byte stream *must* be stored in this column (with a '1PB' or '1QB' variable-length array column format). The corresponding COMPRESSED_DATA column for these tiles *must* contain a null pointer (i.e., the pair of integers that constitute the descriptor for the column *must* both have the value zero: see Sect. 7.3.5).

The compressed data columns described above *may* use either the '1P' or '1Q' variable-length array FITS column format if the size of the heap in the compressed FITS file is < 2.1 GB. If the the heap is larger than 2.1 GB, then the '1Q' format (which uses 64-bit pointers) *must* be used.

When using the optional quantization method described in Sect. 10.2 to compress floating-point images, the following columns are *required*.

ZSCALE – [floating-point; optional] This column *shall* be used to contain linear scale factors that, along with ZZERO, transform the floating-point pixel values in each tile to integers via,

$$I_i = round\left(\frac{F_i - \mathrm{ZZERO}}{\mathrm{ZSCALE}}\right) \tag{12}$$

where I_i and F_i are the integer and (original) floating-point values of the image pixels, respectively and the round function rounds the result to the nearest integer value.

ZZERO – [floating-point; optional] This column *shall* be used to contain zero point offsets that are used to scale the floating-point pixel values in each tile to integers via Eq. 12.

Do not confuse the ZSCALE and ZZERO columns with the BSCALE and BZERO keywords (defined in Sect. 4.4.2) which may be present in integer FITS images. Any such integer images should normally be compressed without any further scaling, and the BSCALE and BZERO keywords *should* be copied verbatim into the header of the binary table containing the compressed image.

Some images contain undefined pixel values; in uncompressed floating-point images these pixels have an IEEE NaN value. However, these pixel values will be altered when using the quantization method described in Sect. 10.2 to compress floating-point images. The value of the undefined pixels *may* be preserved in the following way.

are the mandatory NAXIS1, NAXIS2, PCOUNT, and TFORM*n* keywords, and the optional CHECKSUM, DATASUM (see Sect. 4.4.2.7), and THEAP keywords. These keywords must necessarily describe the contents and structure of the compressed table itself. The original values of these keywords in the uncompressed table *must* be stored in a new set of reserved keywords in the compressed table header. Note that there is no need to preserve a copy of the GCOUNT keyword because the value is always equal to 1 for BINTABLES. The complete set of keywords that have a reserved meaning within a tile-compressed binary table are given below:

ZTABLE – [logical; value: 'T'] The value field of this keyword *shall* be 'T' to indicate that the FITS binary table extension contains a compressed BINTABLE, and that logically this extension *should* be interpreted as a tile-compressed binary table.

ZNAXIS1 – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the NAXIS1 keyword in the original uncompressed FITS table header. This represents the width in bytes of each row in the uncompressed table.

ZNAXIS2 – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the NAXIS2 keyword in the original uncompressed FITS table header. This represents the number of rows in the uncompressed table.

ZPCOUNT – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the PCOUNT keyword in the original uncompressed FITS table header.

ZFORM*n* – [string; indexed; default: none] The value field of these keywords *shall* contain the stringstring values of the corresponding TFORM*n* keywords that defines the data type of column *n* in the original uncompressed FITS table.

ZCTYP*n* – [string; indexed; default: none] The value field of these keywords *shall* contain the character string value mnemonic name of the algorithm that was used to compress column *n* of the table. The only permitted values are given in Sect. 10.3.5, and the corresponding algorithms are described in Sect 10.4.

ZTILELEN – [integer; default: none] The value field of this keyword *shall* contain an integer representing the number of rows of data from the original binary table that are contained in each tile of the compressed table. The number of rows in the last tile may be less than in the previous tiles. Note that if the entire table is compressed as a single tile, then the compressed table will only contains a single row, and the ZTILELEN and ZNAXIS2 keywords will have the same value.

10.3.2. Procedure for Table Compression

The procedure for compressing a FITS binary table consists of the following sequence of steps:

1. Divide table into tiles (optional)
   In order to limit the amount of data that must be managed at one time, large FITS tables *may* be divided into tiles, each containing the same number of rows (except for the last tile which *may* contain fewer rows). Each tile of the table is compressed in order and each is stored in a single row in the output compressed table. There is no fixed upper limit on the allowed tile size, but for practical purposes it is *recommended* that it not exceed 100 MB.
2. Decompose each tile into the component columns
   FITS binary tables are physically stored in row-by-row sequential order, such that the data values for the first row in each column are followed by the values in the second row, and so on (see Sect. 7.3.3). Because adjacent columns in binary tables can contain very non-homogeneous types of data, it can be challenging to efficiently compress the native stream of bytes in the FITS tables. For this reason, the table is first decomposed into its component columns, and then each column of data is compressed separately. This also allows one to choose the most efficient compression algorithm for each column.
3. Compress each column of data
   Each column of data *must* be compressed with one of the lossless compression algorithms described in Sect. 10.4. If the table is divided into tiles, then the same compression algorithm *must* be applied to a given column in every tile. In the case of variable-length array columns (where the data are stored in the table heap: see Sect. 7.3.5), each individual variable length vector *must* be compressed separately.
4. Store the compressed bytes
   The compressed stream of bytes for each column *must* be written into the corresponding column in the output table. The compressed table *must* have exactly the same number and order of columns as the input table, however the data type of the columns in the output table will all have a variable-length byte data type, with TFORM*n* = '1QB'. Each row in the compressed table corresponds to a tile of rows in the uncompressed table.
   In the case of variable-length array columns, the array of descriptors that point to each compressed variable-length array, as well as the array of descriptors from the input uncompressed table, *must* also be compressed and written into the corresponding column in the compressed table. See Sect. 10.3.6 for more details.

10.3.3. Compression Directive Keywords

The following compression-directive keywords, if present in the header of the table to be compressed, are reserved to provide guidance to the compression software on how the table should be compressed. The compression software *should* attempt to obey these directives, but if that is not possible the software may disregard them and use an appropriate alternative. These keywords are optional, but must be used as specified below.

– FZTILELN – [integer] The value field of this keyword *shall* contain an integer that specifies the requested number of table rows in each tile which are to be compressed as a group.

– FZALGOR – [string] The value field of this keyword *shall* contain a character string giving the mnemonic name of the algorithm that is requested to be used by default to compress every column in the table. The permitted values are given in Sect. 10.3.5.

DRAFT

– FZALG*n* – [string; indexed] The value fields of these keywords *shall* contain a character string giving the mnemonic name of the algorithm that is requested to compress column *n* of the table. The current allowed values are the same as for the FZALGOR keyword. The FZALGn keyword takes precedence over FZALGOR in determining which algorithm to use for a particular column if both keywords are present.

10.3.4. Other Reserved Keywords

The following keywords are reserved to store a verbatim copy of the value and comment fields for specific keywords in the original uncompressed BINTABLE. These keywords, if present, *should* be used to reconstruct an identical copy of the uncompressed BINTABLE, and *should not* appear in the compressed table header unless the corresponding keywords were present in the uncompressed BINTABLE.

ZTHEAP – [integer; default: none] The value field of this keyword *shall* contain an integer that gives the value of the THEAP keyword if present in the original uncompressed FITS table header.

ZHECKSUM – [string; default: none] The value field of this keyword *shall* contain a character string that gives the value of the CHECKSUM keyword (see Sect. 4.4.2.7) in the original uncompressed FITS HDU.

ZDATASUM – [string; default: none] The value field of this keyword *shall* contain a character that gives the value of the DATASUM keyword (see Sect. 4.4.2.7) in the original uncompressed FITS HDU.

10.3.5. Supported Compression Algorithms for Tables

The permitted algorithms for compressing BINTABLE columns are RICE_1, GZIP_1, and GZIP_2 (plus NOCOMPRESS), which are lossless and are described in Sect. 10.4. Lossy compression could be allowed in the future once a process is defined to preserve the details of the compression.

10.3.6. Compressing Variable-Length Array Columns

Compression of BINTABLE tiles that contain variable-length array (VLA) columns requires special consideration because the array values in these columns are not stored directly in the table, but are instead stored in a data heap which follows the main table (see Sect. 7.3.5). The VLA column in the original, uncompressed table only contains descriptors, which are composed of two integers that give the size and location of the arrays in the heap. When uncompressing, these descriptor values will be needed to write the uncompressed VLAs back into the same location in the heap as in the original uncompressed table. Thus, the following process *must* be followed, in order, when compressing a VLA column within a tile:

1. For each VLA in the column:
   – Read the array from the input table and compress it using the algorithm specified by ZCTYP for this VLA column.
   – Write the resulting bytestream to the heap of the compressed table.
   – Store (or append) the descriptors to the compressed bytestream (which *must* be 64-bit Q-type) in a temporary array.
2. Append the VLA descriptors from the uncompressed table (which *may* be either Q-type or P-type) to the temporary array of VLA descriptors for the compressed table.
3. Compress the combined array of descriptors using GZIP_1, and write that byte stream into the corresponding VLA column in the output table, so that the compressed array is appended to the heap.

When uncompressing a VLA column, two stages of uncompression *must* be performed in order:

1. Uncompress the combined array of descriptors using the gzip algorithm.
2. For each descriptor to a compressed array:
   – Read the compressed VLA from the compressed table and uncompress it using the algorithm specified by ZCTYP for this VLA column.
   – Write it to the correct location in the uncompressed table.

10.4. Compression Algorithms

Table 36: Valid mnemonic values for the ZCMPTYPE and ZCTYP*n* keywords

Value	Sect.	Compression Type
'RICE_1'	10.4.1	Rice algorithm for integer data
'GZIP_1'	10.4.2	Combination of the LZ77 algorithm and Huffman coding, used in Gnu GZIP
'GZIP_2'	10.4.2	Like 'GZIP_1', but with reshuffled pixel values
'PLIO_1'	10.4.3	IRAF PLIO algorithm for integer data
'HCOMPRESS_1'	10.4.4	H-compress algorithm for 2-D images
'NOCOMPRESS'		The HDU remains uncompressed

The name of the permitted algorithms for compressing FITS HDUs, as recorded in the ZCMPTYPE keyword, are listed in Table 36; if other types are later supported, they *must* be registered with the IAUFWG to reserve the keyword values. Keywords for the parameters of supported compression algorithms have also been reserved, and are described with each algorithm in the subsections below. If alternative compression algorithms require keywords beyond those defined below, they *must* also be registered with the IAUFWG to reserve the associated keyword names.

10.4.1. Rice compression

When ZCMPTYPE = 'RICE_1' the Rice algorithm (Rice et al. 1993) *shall* be used for data (de)compression. When selected, the keywords in Table 37 *should* also appear in the header with one of the values indicated. If these keywords are absent, then their default values *must* be used. The Rice algorithm is lossless, but can only be applied to integer-valued arrays. It offers a significant performance advantage over the other compression techniques (see White et al. 2013).

10.4.2. GZIP compression

When ZCMPTYPE = 'GZIP_1' the gzip algorithm *shall* be used for data (de)compression. There are no algorithm parameters, so the keywords ZNAME*n* and ZVAL*n* *should not* appear in the

DRAFT