9.7. Durations

There is an extensive collection of header keywords that indicate time durations, such as exposure times, but there are many pitfalls and subtleties that make this seemingly simple concept treacherous. Because of their crucial role and common use, keywords are defined below to record exposure and elapsed time.

XPOSURE – [floating-point] The value field of this keyword *shall* contain the value for the effective exposure duration for the data, corrected for dead time and lost time in the units of TIMEUNIT. If the HDU contains multiple time slices, this value *shall* be the total accumulated exposure time over all slices.

TELAPSE – [floating-point] The value field of this keyword *shall* contain the value for the amount of time elapsed, in the units of TIMEUNIT, between the start and the end of the observation or data stream.

Durations *must not* be expressed in ISO-8601 format, but only as actual durations (i.e., numerical values) in the units of the specified time unit.

Good-Time-Interval (GTI) tables are common for exposures with gaps in them, particularly photon-event files, as they make it possible to distinguish time intervals with “no signal detected” from “no data taken.” GTI tables in BINTABLE extensions *must* contain two mandatory columns, START and STOP, and *may* contain one optional column, WEIGHT. The first two define the interval, the third, with a value between 0 and 1, the quality of the interval; *i.e.*, a weight of 0 indicates a *Bad*-Time-Interval. WEIGHT has a default value of 1. Any time interval not covered in the table shall be considered to have a weight of zero.

9.8. Recommended best practices

The following guidelines should be helpful in creating data products with a complete and correct time representation.

– The presence of the informational DATE keyword is *strongly recommended* in all HDUs.
– One or more of the informational keywords DATE-xxxx and/or MJD-xxxx *should* be present in all HDUs whenever a meaningful value can be determined. This also applies, e.g., to catalogs derived from data collected over a well-defined time range.
– The global keyword TIMESYS is *strongly recommended*.
– The global keywords MJDREF or JDREF or DATEREF are *recommended*.
– The remaining informational and global keywords *should* be present whenever applicable.
– All context-specific keywords *shall* be present as needed and required by the context of the data.

9.8.1. Global keywords and overrides

For reference to the keywords that are discussed here, see Table 22. The globally applicable keywords listed in section B of the table serve as default values for the corresponding C* and TC* keywords in that same section, but only when axis and column specifications (including alternate coordinate definitions) use a time scale listed in Table 30 or when the corresponding CTYPE or TTYPE keywords are set to the value 'TIME'. Any alternate coordinate specified in a non-recognized time scale assumes the value of the axis pixels or the column cells, optionally modified by applicable scaling and/or reference value keywords; see also Section 9.2.1.

9.8.2. Restrictions on alternate descriptions

An image will have at most one time axis as identified by having the CTYPE*i* value of TIME or one of the values listed in Table 30. Consequently, as long as the axis is identified through CTYPE*i*, there is no need to have axis number identification on the global time-related keywords. It is expressly prohibited to specify more than one time reference position on this axis for alternate time coordinate frames, since this would give rise to complicated model-dependent non-linear relations between these frames. Hence, time scales TDB and TCB (or ET, to its precision) may be specified in the same image, but cannot be combined with any of the first nine time scales in Table 30; those first nine can be expressed as linear transformations of each other, too, provided the reference position remains unchanged. Time scale LOCAL is by itself, intended for simulations, and should not be mixed with any of the others.

9.8.3. Image time axes

Section 8.2 requires keywords CRVAL*ia* to be numeric and they cannot be expressed in ISO-8601 format. Therefore it is *required* that CRVAL*ia* contain the elapsed time in units of TIMEUNIT or CUNIT*ia*, even if the zero point of time is specified by DATEREF. If the image does not use a matrix for scaling, rotation and shear (Greisen & Calabretta 2002), CDELT*ia* provides the numeric value for the time interval. If the PC form of scaling, rotation and shear (Greisen & Calabretta 2002) is used, CDELT*ia* provides the numeric value for the time interval, and PC*i_j*, where $i = j =$ the index of the time axis (in the typical case of an image cube with axis 3 being time, $i = j = 3$) would take the exact value 1, the default (Greisen & Calabretta 2002). When the CD*i_j* form of mapping is used, CD*i_j* provides the numeric value for the time interval. If one of the axes is time and the matrix form is used, then the treatment of the PC*i_ja* (or CD*i_ja*) matrices involves at least a Minkowsky metric and Lorentz transformations (as contrasted with Euclidean and Galilean).

10. Representations of compressed data

Minimizing data volume is important in may contexts, particularly for publishers of large astronomical data collections. The following sections describe compressed representations of data in FITS images and BINTABLES that preserve metadata and allow for full or partial extraction of the original data as necessary. The resulting FITS file structure is independent of the specific data compression algorithm employed. The implementation details for some compression algorithms that are widely used in astronomy are defined in Sect. 10.4, but other compression techniques could also be supported. See the FITS convention by White et al. (2013) for details of the compression techniques, but beware that the specifications in this Standard *shall* supersede those in the registered convention.

Compression of FITS files can be beneficial for sites that store or distribute large quantities of data; the present section provides a standard framework that addresses such needs. As implementation of compression/decompression codes can be quite complex, not all FITS reading and writing software is necessarily expected to support these capabilities. External utilities are available to compress and uncompress FITS files[14].

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

[14] e.g. `fpack/funpack`, see `https://heasarc.gsfc.nasa.gov/fitsio/fpack/`