A *Multi Unit Spectroscopic Explorer for ESO Very Large Telescope*

MUSE Python Data Analysis Framework

Interface Control Document for source file

Reference	
Issue	0.4
Date	30-06-2016

Prepared	*Date*	*Signature*
Laure Piqueras	24-04-2015	
Verified	*Date*	*Signature*
Approved & Released	*Date*	*Signature*

ESO	Göttingen	Leiden	Lyon	Potsdam	Toulouse	Zurich

Change Record			
Issue	Date	Section affected	Reason / Initiation / Documents / Remarks
0.1	24/04/2015	All	Creation
	28/04/2015	2.2.2 2.3.2	Rename image extension Magnitude is unitless
	07/05/2015	All	JR comments
0.2	13/05/2015	2.2.3	Several cubes
0.3	29/06/2015	All	TAB extensions
 0.4	28/07/2015 30/07/2016	2.3.3 All	Redshift error becomes a redshift interval Update according to mpdaf v 2.0

Table of Contents

1 Introduction

1.1 Documents

1.1.1 Applicable Documents

AD1		

1.1.2 Reference Documents

RD1	Definition of the Flexible Image Transport System (FITS)	NOST 100-2.0, 29/03/1999
RD2	Definition of the Flexible Image Transport System (FITS) – Hanisch et al.	2001, A&A, 376, 359
RD3	The FITS image extension - Ponz et al.	1994, A&A Supp., 105, 53
RD4	Binary table extension to FITS – Cotton et al.	1995, A&A Supp., 113, 159
RD5	Representations of World Coordinates in FITS – Greisen et al.	2002, A&A, 395, 1061

1.2 Abbreviations and Acronyms

1.3 Scope

A lot of tools are currently been developed to detect sources in MUSE data cubes. In order to easily exchange information about detected sources, we need to define a format for storing source file.

The Flexible Image Transport System (FITS) is widely used in the astronomical community and has been defined in RD1 and RD2. Basic FITS extensions like images and tables are defined in RD3 and RD4, respectively.

A FITS file format is used to store the information about a detected source. This document describes the interface definitions for this source FITS file.

2 Source FITS file

Source FITS file contains the usual primary header with an empty data array and optional extensions giving further information about the source.

Generic information are stored in the primary header of the file, FITS image extension may be used to store spectra, small images and sub data cubes and FITS binary table extensions may be used to store the information relative to line profiles, magnitudes and redshift values. The overall structure of the file is described in Table 1.

We distinguish between ten different types of extensions:

1. The LINES extension, which is unique (maximum one per file) but not mandatory. This FITS table extension is used to describe the parameters of spectral lines.
2. The Z extension, which is unique but not mandatory. This is a table extension that contains redshift values.
3. The MAG extension, which is unique but not mandatory. This a table extension that contains magnitude values.
4. The SPE_xxx_DATA extension(s), which are not mandatory and not unique (there can be multiple SPE_xxx_DATA extension per file). These extensions are used to describe spectra of the source. The "xxx" is used to distinguish the different spectra.
5. The SPE_xxx_STAT extension(s), which are not mandatory and not unique. The extension SPE_xxx_STAT may contain the variance of the spectrum that is stored in the SPE_xxx_DATA extension.
6. The IMA_xxx_DATA extension(s), which are not mandatory and not unique (there can be multiple IMA_xxx_DATA extension per file). These extensions may contain images of the source. The "xxx" is used to distinguish the different images.
7. The IMA_xxx_STAT extension(s), which are not mandatory and not unique. The extension IMA_xxx_STAT may contain the variance of the image that is stored in the IMA_xxx_DATA extension.
8. The CUB_xxx_DATA extension(s), which are not mandatory and not unique. These FITS image extensions may contain small data cubes.
9. The CUB_xxx_STAT extension(s), which are not mandatory and not unique. The extension CUB_xxx_STAT may contain the variance of the cube that is stored in the CUB_xxx_DATA extension.
10. The TAB_xxx extension(s), which are not mandatory and not unique. These FITS table extension are used to store possible information.

.fits		Source file
FITS extension	Description	Section
0	Primary header with an empty primary data array.	
LINES	Non-mandatory unique binary table extension used to describe the lines profiles	2.3.1
Z	Non-mandatory unique binary table extension used to store redshift values	2.3.3
MAG	Non-mandatory unique binary table extension used to store magnitude values	2.3.2
SPE_xxx_DATA	Not mandatory and not unique 1D image extension used to describe a spectra of the source.	2.2.1
SPE_xxx_STAT	Not mandatory and not unique 1D image extension that contains the variance of the spectrum stored in the SPE_xxx_DATA extension	2.2.1
IMA_xxx_DATA	Not mandatory and not unique 2D image extension that contains an image of the source.	2.2.2
IMA_xxx_STAT	Not mandatory and not unique 2D image extension that contains the variance of the image stored in the IMA_xxx_DATA extension	2.2.2
CUB_xxx_DATA	Not mandatory and not unique 3D image extension that contains a small data cube.	2.2.3
CUB_xxx_STAT	Not mandatory and not unique 3D image extension that contains the variance of the cube stored in the CUBE_DATA extension.	2.2.3
TAB_xxx	Not mandatory and not unique binary table extension	

Table 1: Overall structure of a source file

2.1 FITS primary header

The primary header of a source file contains, in addition to keywords that are mandatory in the FITS definition, keywords that gives information about the source (position, origin, confidence ...). In Table 2 we listed the main keywords present in the primary header of the source FITS file. This list is not restrictive, i.e. other keywords can be present / added in the primary header (as long as they comply with the FITS standards, RD1 and RD2).

Keyword	Type	Value	Description	Status
SIMPLE	boolean	T	data conform to FITS standard	M
BITPIX	integer	8	bits per data value	M
NAXIS	integer	0	number of data axes	M
EXTEND	boolean	T	file has extension(s)	M
DATE	string	'yyyy-mm-ddThh:mm:ss'	file creation date	R
AUTHOR	string		institution or individual responsible for creating the file	R
ID	integer		ID of the source	S
RA	float		Right ascension in degrees	S
DEC	float		Declination in degrees	S
ORIGIN	string		Name of the software used to detect the source	S
ORIGIN_V	string	'vx.x'	Version of the software used to detect the source	S
CUBE	string		Name of the MUSE data cube	S
CUBE_V	float		Cube version	O
DPROBA	float		Detection probability	O
CONFI	integer		Expert confidence index	O
COMxxx	string		User comment the comment of this keyword must contains the author and the date of the comment	O
HISTxxx	string		History	
END				M

Table 2: List of keywords that can be present in the primary header of the source FITS files. The following status characters are used: M for mandatory FITS keywords; R for reserved FITS keywords; S for mandatory source fits keywords, O for optional.

2.2 FITS image extensions

FITS image extensions will be used when it is necessary to store 1D, 2D or 3D data arrays. The way the data will be stored is defined in RD3.

The FITS image extensions also carry the concept of the “world coordinates”. These coordinates are defined using a set of keywords in the header of the image (RD5).

2.2.1 FITS image extensions used to store spectra

The source may be described by several spectra.

A spectrum is a 1D data array containing flux values and wavelength solution linear with the array index. Each spectrum is stored in a 1D image extension labelled 'SPE_xxx_DATA', associated with world coordinates containing the wavelength information. 'xxx' is the name of the spectrum (as a string). In Table 3 we listed the usualy names used to distingish the spectrum extensions of source files. This list is not restrictive, other spectra can be added.

Optionally, a variance data array can be attached to the spectrum xxx. Then, it is saved in a 1D image extension labelled 'SPE-xxx-STAT'.

The overall structure of the two images extensions used to store spectrum in the source FITS file is described in Table 4.

Spectrum name	
Name	Description
TOT	Global spectrum
CONT	Continuous spectrum

Table 3: List of usual names used to distingish different spectra in the source files

.fits	Spectrum extensions of the source file
FITS extension	Description

SPE_xxx_DATA	This 1D image extension is not mandatory and not unique (there can be multiple SPE_xxx_DATA extension per file). These extensions are used to store flux values of the a spectrum.
	
	A typical header would look like that (in this example, we have assumed that the spectrum contains 100 values and the first flux value corresponds to a wavelength of 4000 Angstroms, and the next values are spaced by 1.25 Angstroms):
	
	`XTENSION= 'IMAGE   '           / image extension`
	`EXTNAME = 'SPE_TEST_DATA'      / extension name`
	`COMMENT = 'test'               / descriptive comment`
	`BITPIX  =                  -32 / array data type`
	`NAXIS   =                    1 / number of array dimensions`
	`NAXIS1  =                  100`
	`PCOUNT  =                    0 / number of parameters`
	`GCOUNT  =                    1 / number of groups`
	`CRVAL1  =               4800.0 / Start in world coordinate`
	`CRPIX1  =                  1.0 / Start in pixel`
	`CDELT1  =                 1.25 / Step in world coordinate`
	`CTYPE1  = 'LINEAR'             / world coordinate type`
	`CUNIT1  = 'Angstrom'           / world coordinate units`
	`BUNIT`
	`END`

SPE_xxx_STAT	This 1D image extension is not mandatory and not unique (there can be multiple SPE_xxx_STAT extension per file). These extensions are used to store variance values of the a spectrum.
	
	A typical header would look like that:
	
	`XTENSION= 'IMAGE   '           / image extension`
	`EXTNAME = 'SPE_TEST_STAT'      / extension name`
	`COMMENT = 'test'               / descriptive comment`
	`BITPIX  =                  -32 / array data type`
	`NAXIS   =                    1 / number of array dimensions`
	`NAXIS1  =                  100`
	`PCOUNT  =                    0 / number of parameters`
	`GCOUNT  =                    1 / number of groups`
	`CRVAL1  =               4800.0 / Start in world coordinate`
	`CRPIX1  =                  1.0 / Start in pixel`
	`CDELT1  =                 1.25 / Step in world coordinate`
	`CTYPE1  = 'LINEAR'             / world coordinate type`
	`CUNIT1  = 'Angstrom'           / world coordinate units`
	`BUNIT`
	`END`

Table 4: Overall structure of the two 1D image extensions used to store a spectrum

2.2.2 FITS image extensions used to store images

A source file may contain several images.

Each image is stored in a 2D image extension labelled 'IMA_xxx_DATA', associated with world coordinates containing spatial information. 'xxx' is a string used to distingish the different images. Table 5 lists the usualy names used but this list is not restrictive.

Optionally, a variance data array can be attached to the image xxx. Then, it is saved in a 2D image extension labelled 'IMA-xxx-STAT'.

The overall structure of the two images extensions used to store image in the source FITS file is described in Table 6.

Image name	
Name	Description
MUSE_WHITE	White image computed from MUSE data cube.
MASK	Array of booleans that determines if the pixel is in the source or not
HST_xxx	Image from Hubble (xxx is the filter name, HST_F606 for example)
NB_xxx	Narrow-band image (HST_LYALPHA for example)

Table 5: List of usual names used to distingish different images in the source files

.fits	image extensions of the source file
FITS extension	Description

IMA_xxx_DATA	This 2D image extension is not mandatory and not unique (there can be multiple IMA_xxx_DATA extension per file). These extensions are used to store small images.
	
	A typical header would look like that (in this example, the image is composed by 27x27 pixels of 0.2 arcsec, centered on (-60:33:49.0809,22:32:57.8485) and rotated by -0.2 degrees)
	
	`XTENSION= 'IMAGE    '      / image extension`
	`EXTNAME = 'IMA_TEST_DATA' / extension name`
	`COMMENT = 'test'           / descriptive comment`
	`BITPIX  =             -32 / array data type`
	`NAXIS   =               2 / number of array dimensions`
	`NAXIS1  =              27`
	`NAXIS2  =              27`
	`PCOUNT  =               0 / number of parameters`
	`GCOUNT  =               1 / number of groups`
	`CTYPE1  = 'RA---TAN '      / Right ascension, gnomonic projection`
	`CUNIT1  = 'deg      '      / Units of coordinate increment and value`
	`CRVAL1  =       338.24103 / [deg] Coordinate value at reference point`
	`CRPIX1  =            14.0 / Pixel coordinate at reference point`
	`CD1_1   =    -5.549201E-05 / partial of first axis coordinate w.r.t. x`
	`CD1_2   =     -1.89661E-07 / partial of first axis coordinate w.r.t. y`
	`CTYPE2  = 'DEC--TAN '      / Declination, gnomonic projection`
	`CUNIT2  = 'deg      '      / Units of coordinate increment and value`
	`CRVAL2  =       -60.56363 / [deg] Coordinate value at reference point`
	`CRPIX2  =            14.0 / Pixel coordinate at reference point`
	`CD2_1   =     -1.89703E-07 / partial of 2nd axis coordinate w.r.t. x`
	`CD1_2   =      5.55042E-05 / partial of 2nd axis coordinate w.r.t. y`
	`BUNIT`
	`END`

IMA_xxx_STAT	This 2D image extension is not mandatory and not unique (there can be multiple IMA_xxx_STAT extension per file). These extensions are used to store variance values of the an image. Except the EXTNAME keyword, its header would be the same that the IMA_xxx_DATA header. `BUNIT`

Table 6: Overall structure of the two 2D FITS image extensions used to store an image

2.2.3 FITS image extension used to store small data cubes

A source file may contain small data cubes.

Each cube is stored in a 3D image extension labelled 'CUB_xxx_DATA', associated with world coordinates containing spatial and spectral information. 'xxx' is a string used to distingish the different cubes. Table 7 lists the usualy names used but this list is not restrictive.

Optionally, a variance data array can be attached to the cube xxx. Then, it is saved in a 2D image extension labelled 'CUB-xxx-STAT'.

The overall structure of the two images extensions is described in Table 8.

Cube name	
Name	Description
SRC	Cube that contains only the source (the potential neighboring sources has been removed)
MUSE	Sub-cube computed from MUSE data cube.
MASK	Array of booleans that determines if the pixel is in the source or not

Table 7: List of usual names used to distingish different cubes in the source files

.fits	cube extensions of the source file
FITS extension	Description

CUB_TEST_DATA	This 3D image extension is not mandatory. It is used to store a small data cube.
	
	A typical header would look like that (in this example, the cube is composed by 27x27 pixels of 0.2 arcsec, centered on (-60:33:49.0809,22:32:57.8485) and rotated by -0.2 degrees)
	
	`XTENSION= 'IMAGE    '      / image extension`
	`EXTNAME = 'CUB_TEST_DATA' / extension name`
	`COMMENT = 'test'           / descriptive comment`
	`BITPIX  =              -32 / array data type`
	`NAXIS   =                3 / number of array dimensions`
	`NAXIS1  =               27`
	`NAXIS2  =               27`
	`NAXIS3 =               100`
	`PCOUNT  =                0 / number of parameters`
	`GCOUNT  =                1 / number of groups`
	`CTYPE1  = 'RA---TAN '      / Right ascension, gnomonic projection`
	`CUNIT1  = 'deg      '      / Units of coordinate increment and value`
	`CRVAL1   =         338.24103 /  [deg] Coordinate value at reference point`
	`CRPIX1  =             14.0 / Pixel coordinate at reference point`
	`CD1_1    =   -5.549201E-05 / partial of first axis coordinate w.r.t. x`
	`CD1_2    =     -1.89661E-07 / partial of first axis coordinate w.r.t. y`
	`CTYPE2  = 'DEC--TAN '      / Declination, gnomonic projection`
	`CUNIT2  = 'deg      '      / Units of coordinate increment and value`
	`CRVAL2   =         -60.56363 /  [deg] Coordinate value at reference point`
	`CRPIX2  =             14.0 / Pixel coordinate at reference point`
	`CD2_1   =    -1.89703E-07 / partial of 2nd axis coordinate w.r.t. x`
	`CD1_2   =     5.55042E-05 / partial of 2nd axis coordinate w.r.t. y`
	`CRVAL3  =                  4800.0 / Start in world coordinate`
	`CRPIX3  =                     1.0 / Start in pixel`
	`CDELT3  =                    1.25 / Step in world coordinate`
	`CTYPE3  = 'LINEAR'                / world coordinate type`
	`CUNIT3  = 'Angstrom'              / world coordinate units`
	`BUNIT`
	`END`

CUB_TEST_STAT	This 3D image extension is not mandatory. It is used to store variance values of the cube. Except the EXTNAME keyword, its header would be the same that the CUB_TEST_DATA header. `BUNIT`

Table 8: Overall structure of the two 2D FITS image extensions used to store a small data cube

2.3 FITS binary table extensions

FITS table extensions will be used when it is necessary to store the information relative to line profiles, magnitudes and redshift values.

There are two kinds of table in the FITS standard: binary tables and ASCII tables. We will use binary tables because they are more economical in storage and faster in data access and manipulation. A detailed definition of the format of the FITS table extension can be found in RD4.

2.3.1 FITS table extension used to store lines

Lines profiles are stored in the Source file via a binary table labeled 'LINES'. This extension is unique (maximum one per file) but not mandatory.

The Table 8 lists the quantities that are used in the source file to characterize the profile of each line. Each quantity corresponds to a column of the LINES table. The overall structure of the LINES extensions is described in Table 9.

Columns of the LINES table		
Name	Description	Unit
LBDA_OBS	Observed wavelength	Angstroms
LBDA_OBS_ERR	Observed wavelength error	Angstroms
FWHM_OBS	Observed Full Width at Half Maximum	Angstroms
FWHM_OBS_ERR	Observed FWHM error	Angstroms
FLUX	Integrated flux	10^{-20} erg/s/cm^2
FLUX_ERR	Integrated flux error	10^{-20} erg/s/cm^2
LINE	Line name	unitless
LINE_LBDA	Vacuum wavelength	Angstroms

Table 9: Characterization of a line profile

.fits	LINES extension
FITS extension	Description
[LINES]	This non mandatory but unique extension is a binary table containing description of the spectral lines of the object. It will contain height columns: one column *LBDA_OBS* giving the observed wavelength (float, in Angstroms); one column *LBDA_OBS_ERR* giving the observed wavelength error (float, in Angstroms); one column *FWHM_OBS* giving the observed FWHM (float, in Angstroms); one column *FWHM_OBS_ERR* giving the observed FWHM error (float, in Angstroms); one column *FLUX* giving the flux(float, in 10^{-20} erg/s/cm^2); one column *FLUX_ERR* giving the flux error (float, in 10^{-20} erg/s/cm^2); one column *LINE* giving the line name (string); one column LINE_*LBDA* giving the vacuum wavelength (float, in Angstroms) A typical header would look like that (in this example, we have assumed a number of rows of 3):

```
XTENSION= 'BINTABLE'              / extension type
EXTNAME = 'LINES    '             / extension name
BITPIX  =                       8 / number of bits per pixel
NAXIS   =                       2 / number of axes
NAXIS1  =                      12 / number of 8 bit bytes in each row
NAXIS2  =                       3 / number of rows (example)
PCOUNT  =                       0 / number of parameters per group
GCOUNT  =                       1 / number of groups
TFIELDS =                       8 / number of columns in the table
TFORM001= '1E          '          / data type for column 1 (float)
TFORM002= '1E          '          / data type for column 2 (float)
TFORM003= '1E          '          / data type for column 3 (float)
TFORM004= '1E          '          / data type for column 4 (float)
TFORM005= '1E          '          / data type for column 5 (float)
TFORM006= '1E          '          / data type for column 6 (float)
TFORM007= '1E          '          / data type for column 7 (float)
TFORM008= '1E          '          / data type for column 8 (float)
TFORM009= '1E          '          / data type for column 9 (float)
TFORM010= '1E          '          / data type for column 10 (float)
TFORM011= '20A          '         / data type for column 11 (string)
TTYPE001= 'LBDA_OBS    '          / label of column 1
TTYPE002= 'LBDA_OBS_ERR '         / label of column 2
TTYPE003= 'FWHM_OBS    '          / label of column 3
TTYPE004= 'FWHM_OBS_ERR '         / label of column 4
TTYPE005= 'FLUX        '          / label of column 5
TTYPE006= 'FLUX_ERR    '          / label of column 6
TTYPE007= 'LINE        '          / label of column 7
TTYPE008= 'LINE_LBDA   '          / label of column 8
TUNIT001= 'ANGSTROMS '            / unit of column 1
TUNIT002= 'ANGSTROMS '            / unit of column 2
TUNIT003= 'ANGSTROMS '            / unit of column 3
TUNIT004= 'ANGSTROMS '            / unit of column 4
TUNIT005= '10**(-20)*erg/s/cm**2' / unit of column 5
TUNIT006= '10**(-20)*erg/s/cm**2' / unit of column 6
TUNIT007= 'UNITLESS'              / unit of column 7
TUNIT008= 'ANGSTROMS '            / unit of column 8
END
```

Table 10: Overall structure of the FITS binary table extension used to store line profiles

2.3.2 FITS table extension used to store magnitudes

Magnitudes are stored in the Source file via a binary table labeled 'MAG'. This extension is unique (maximum one per file) but not mandatory.

This table contains three columns, the first gives the name of the wavelength band, the secong gives the magnitude value and the third gives the estimation of error (see Table 10). The overall structure of the MAG extensions is described in Table 11.

Columns of the MAG table		
Name	Description	Unit
BAND	Filter name	unitless
MAG	Magnitude value	unitless
MAG_ERR	Magnitude error	unitless

Table 11: Columns of magnitude table

.fits	MAG extension
FITS extension	Description
[MAG]	This non mandatory but unique extension is a binary table containing magnitudes of the source. It will contain three columns: one column *BAND* giving the name of the filter (string); one column *MAG* giving the magnitude value (float, in 10^{-20} erg/s/cm^2/A); one column *MAG_ERR* giving the magnitude error (float, in 10^{-20} erg/s/cm^2/A). A typical header would look like that (in this example, we have assumed a number of rows of 5):

```
XTENSION= 'BINTABLE'                / extension type
EXTNAME = 'MAG     '                / extension name
BITPIX  =                         8 / number of bits per pixel
NAXIS   =                         2 / number of axes
NAXIS1  =                        12 /number of 8 bytes in each row
NAXIS2  =                         5 / number of rows (example)
PCOUNT  =                         0 / number of parameters per group
GCOUNT  =                         1 / number of groups
TFIELDS =                         3 / number of columns in the table
TFORM001= '20A          '           / data type for column 1 (string)
TFORM002= '1E           '           / data type for column 2 (float)
TFORM003= '1E           '           / data type for column 3 (float)
TTYPE001= 'BAND         '           / label of column 1
TTYPE002= 'MAG          '           / label of column 2
TTYPE003= 'MAG_ERR      '           / label of column 3
TUNIT001= 'UNITLESS   '             / unit of column 1
TUNIT002= 'UNITLESS   '             / unit of column 2
TUNIT003= 'UNITLESS   '             / unit of column 3
END
```

Table 12: Overall structure or MAG extension

2.3.3 FITS table extension used to store redshift values

Redshift values are stored in the Source file via a binary table labeled 'Z'. This extension is unique (maximum one per file) but not mandatory.

This table contains three columns, the first gives a description of the redshift, the second gives the value and the two last give the redshift interval. The overall structure of the Z extensions is described in Table 14.

Columns of the Z table		
Name	Description	Unit
Z_DESC	Redshift description	unitless
Z	Redshift value	unitless
Z_MIN	Lower bound of estimated redshift	unitless
Z_MAX	Upper bound of estimated redshift	unitless

Table 13: Columns of redshift table

Redshift names	
Z_DESC	Desciption
PHOTO	Photometric redshift
CORR	Cross-correlation redshift
EMI	Emission-line redshift
ABS	Absorption-line redshit
FINAL	Final redshift

Table 14: List of usual names used to distingish different redshifts

.fits	Z extension
FITS extension	Description
[Z]	This non mandatory but unique extension is a binary table containing redshift values of the source. It will contain three columns: one column *Z_DESC* giving the name of the redshift (string); one column Z giving the redshift value (float); one column *Z_MIN* giving the lower bound (float); One column Z_MAX giving the upper bound (float). A typical header would look like that (in this example, we have assumed a number of rows of 5):

```
XTENSION= 'BINTABLE'                  / extension type
EXTNAME = 'Z       '                  / extension name
BITPIX  =                           8 / number of bits per pixel
NAXIS   =                           2 / number of axes
NAXIS1  =                          12 /number of 8 bytes in each row
NAXIS2  =                           5 / number of rows (example)
PCOUNT  =                           0 / number of parameters per group
GCOUNT  =                           1 / number of groups
TFIELDS =                           3 / number of columns in the table
TFORM001= '20A         '              / data type for column 1 (string)
TFORM002= '1E          '              / data type for column 2 (float)
TFORM003= '1E          '              / data type for column 3 (float)
TFORM004= '1E          '              / data type for column 4 (float)
TTYPE001= 'Z_DESC      '              / label of column 1
TTYPE002= 'Z           '              / label of column 2
TTYPE003= 'Z_MIN       '              / label of column 3
TTYPE004= 'Z_MAX       '              / label of column 4
TUNIT001= 'UNITLESS    '              / unit of column 1
TUNIT002= 'UNITLESS    '              / unit of column 2
TUNIT003= 'UNITLESS    '              / unit of column 3
TUNIT004= 'UNITLESS    '              / unit of column 4
END
```

Table 15: Overall structure or Z extension

2.4 Constraints on the data

1. All wavelengths must be in Angstrom and strictly positive
2. All fluxes must be in 10^{-20} erg/s/cm^2
3. All coordinates in degrees