MissFITS

*v*2.5

User's guide

C. MARMO

E. BERTIN

Institut d'Astrophysique de Paris

Contents

1 What is MissFITS?

MissFITS is a program that performs basic maintenance and packaging tasks on FITS files:

- add, edit and remove FITS header keywords;
- split or join Multi-Extension-FITS (MEF) files;
- pack or unpack data-cubes;
- create/check/update FITS checksums, using R. Seaman's protocol[1];
- compress or decompress FITS data (*not yet available*).

2 License

3 Installing the software

3.1 Software and hardware requirements

MissFITS has been developed on Linux PCs. The software is run in (ANSI) text-mode from a shell. A window system is therefore unnecessary with present versions.

3.2 Obtaining MissFITS

The easiest way to obtain MissFITS is to download it from the official website[2], or alternatively from the current official anonymous FTP site at `ftp://ftp.iap.fr/pub/from_users/bertin/missfits/`. The latest versions of the program are available as standard `.tar.gz` source archives, including the documentation, and RPM binary packages for various architectures. For production, it is strongly advised to install the RPM packages if you are running Linux on an x86 or x86-64 system with RPM-support.

3.3 Installation

To install, you must first uncompress and unarchive the archive:

```
gzip -dc missfits-x.x.tar.gz | tar xvf -
```

[1] see `http://www.adass.org/adass/proceedings/adass94/seamanr.html`
[2] `http://astromatic.net/software/missfits`

A new directory called `missfits-x.x` should now appear at the current position on your disk. You should then just enter the directory and follow the instructions in the file called "`INSTALL`".

The software is also available as a precompiled RPM for Linux systems with an x86 or x86-64 architectures. The simplest way to install an RPM package is to log in as root and use the following command

```
rpm -U missfits-x.x-dist.arch.rpm
```

4 Overview of the software

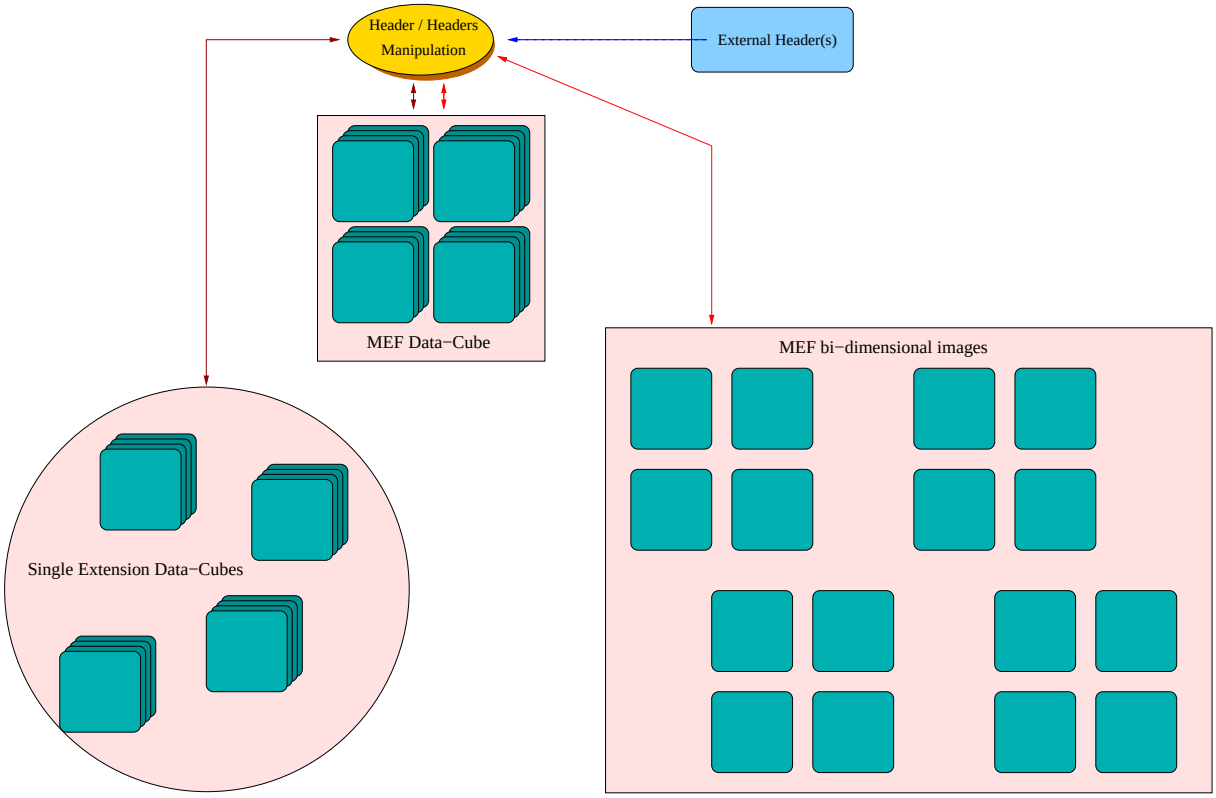

Figure 1: Global layout of MissFITS.

5 Using MissFITS

MissFITS is run from the shell with the following syntax:

% `missfits -c` *configuration-file* `[` *-Parameter1 Value1* `] [` *-Parameter2 Value2* `]` ...

The part enclosed within brackets is optional. Any "*-Parameter Value*" statement in the command-line overrides the corresponding definition in the configuration-file or any default value (see below).

5.1 The Configuration file

Each time MissFITS runs, it looks for a configuration file. If no configuration file is specified in the command-line, it is assumed to be called "`default.missfits`" and to reside in the current directory. If no configuration file is found, MissFITS will use its own internal default configuration.

5.1.1 Creating a configuration file

MISSFITS can generate an ASCII dump of its internal default configuration, using the "-d" option. By redirecting the standard output of MISSFITS to a file, one creates a configuration file that can easily be modified afterward:

```
% missfits -d >default.missfits
```

5.1.2 Format of the configuration file

The format is ASCII. There must be only one parameter per line, following the form:

Config-parameter *Value(s)*

Extra spaces or linefeeds are ignored. Comments must begin with a "#" and end with a linefeed. Values can be of different types: strings (can be enclosed between double quotes), floats, integers, keywords or boolean (`Y/y` or `N/n`). Some parameters accept zero or several values, which must then be separated by commas. Environment variables, written as `$HOME` or `${HOME}` are expanded.

5.1.3 Parameter list

Here is a list of all the parameters known to MISSFITS. Please refer to §5.2 for a detailed description of their meaning. Some "advanced" parameters (indicated with an asterisk) are also listed. They must be used with caution, and may be rescoped or removed without notice in future versions.

Parameter	default	type	Description
REMOVE_KEYWORD		*string(s)*	Keyword(s) to be removed from the header.
REPLACE_KEYWORD		*string(s)*	Keyword(s) to be replaced in the header. – Syntax: OLD_KEY1:NEW_KEY1, OLD_KEY2:NEW_KEY2,...
SLICE_KEYWORD		*string(s)*	Keyword(s) to be sliced.
SLICEKEY_FORMAT	%02d	*string*	Format of keyword(s) referring to each slice.
DISPLAY_KEYWORD	OBJECT	*string(s)*	Keyword(s) to be displayed while processing the files.
HEADER_SUFFIX	.head	*string(s)*	Filename extension for headers to add.
FIX_WFI*	N	*boolean*	Fix WFI camera image-headers?
NEXTENSIONS_MIN	0	*integer(s)*	Minimum number of extensions (warns if less are found).
OUTFILE_TYPE	SAME	*keyword(s)*	Format of output file.
		SAME	– same as input file,
		MULTI	– MEF starting from single extension images,
		SPLIT	– single extension starting from MEF file,
		SLICE	– bi-dimensional FITS files starting from data cubes,
		CUBE	– data cubes starting from bi-dimensional images,
		DIR	– directory containing extensions or slices.
SPLIT_SUFFIX	.%02d.fits	*string*	Suffix expected for split FITS files.
SPLIT_START*	1	*integer*	Suffix number that corresponds to the first extension.
SLICE_SUFFIX	.%02d.fits	*string*	Suffix expected for sliced FITS files.

SLICE_START*	1	*integer*	Suffix number that corresponds to the first slice.
PROCESS_TYPE	NONE	*keyword*	Operations on FITS data (not yet available).
		NONE	– same as input file,
		COMPRESS	– compression,
		UNCOMPRESS	– uncompression.
CHECKSUM_TYPE	NONE	*keyword*	Checksum operations.
		NONE	– no CHECKSUM operation,
		COMPUTE	– compute CHECKSUM,
		VERIFY	– verify CHECKSUM,
		UPDATE	– update CHECKSUM.
SAVE_TYPE	BACKUP	*keyword*	Behaviour towards output filename.
		NONE	– no saving operation,
		BACKUP	– if the output image already exists it is backupped,
		NEW	– the output image is saved as name<NEW_SUFFIX>.fits,
		REPLACE	– input image(s) is/are replaced (dangerous),
NEW_SUFFIX	.miss	*string*	suffix to add in SAVE_TYPE NEW mode.
VERBOSE_TYPE	NORMAL	*keyword*	How much MissFITS comments its operations:
		QUIET	– run silently,
		NORMAL	– display warnings and limited information concerning the work in progress,
		FULL	– display more complete information.
WRITE_XML	Y	*boolean*	Write XML file (Y/N)?
XML_NAME	missfits.xml	*string*	Filename for XML output.
XSL_URL*	missfits.xsl	*string*	location of XSLT style-sheet.

5.2 Detailed description of the configuration parameters

5.2.1 Working on FITS keywords

REMOVE_KEYWORD - Keyword(s) to remove from header in the output file.

REPLACE_KEYWORD - Keyword(s) to replace in the output file. The syntax is OLD_KEY1:NEW_KEY1, OLD_KEY2:NEW_KEY2,... Remember that keywords must be 8 characters or less.

SLICE_KEYWORD - Keyword(s) referring to single slice in data cubes headers are sometimes written as KEY_slice_index. Slicing data cubes, you will maybe translate these keywords in KEY. This parameter contains the root name you want write in the output file (KEY). If you are building a cube starting from bi-dimensional FITS files the selected keywords are written in the header adding the slice index.

SLICEKEY_FORMAT - The indexing format to be used for keywords for each slice in the data cubes.

DISPLAY_KEYWORD - Keyword(s) to be displayed while processing the files.

HEADER_SUFFIX - Suffix of the file containing additional external headers (as in [1] or [2]). In MissFITS an external header is read and appended to the output image header.

FIX_WFI - should be set to Y to fix old WFI camera headers.

5.2.2 Handling FITS files

NEXTENSIONS_MIN - Minimum number of extensions in the input file. A warning is printed if less than NEXTENSIONS_MIN extensions are found among input files. This parameter can be useful to identify missing extensions in files.

OUTFILE_TYPE - Output file type (more details in subsection 5.3). SAME: the output file type is the same as the input one, useful when you are interested only in header manipulation. MULTI: input files have a single FITS extension and you want join them in a MEF file; in the command line FITS files are inputted with their root name without any suffix. SPLIT: in this case the input file is a MEF FITS file which you want to to split into separate extensions. Header manipulation is made on all output images and the external header root is taken from the input image. SLICE: input files are data cubes and you want to slice them in bidimensional images. CUBE: input files bidimensional images which are used to construct data cubes DIR: inputs files are MEFs or data cutes the slices or extensions are written in a directory.

SPLIT_SUFFIX - Suffix describing single extensions images, in input if OUTPUT_FILE is MULTI, or in output if OUTPUT_FILE is SPLIT. This parameter must be set to NONE if OUTPUT_FILE is set to DIR and the input file is a cube you want to unstack. File suffix indices start at the value specified by the SPLIT_START configuration parameter (1 by default).

SLICE_SUFFIX - Suffix describing bidimensional images obtained from an input data cube (if OUTPUT_TYPE is SLICE), or the input files used to build a data cube (if OUTPUT_TYPE is CUBE). File suffix indices start at the value specified by the SLICE_START configuration parameter (1 by default). You will need to set SLICE_SUFFIX to NONE if OUTPUT_FILE is set to DIR and if the input file is a MEF that you want to split as individual files.

PROCESS_TYPE - ** not yet available **.

CHECKSUM_TYPE - To compute, verify or update the CHECKSUM keyword.

SAVE_TYPE - Define how files have to be saved (more details in subsection 5.3).

NEW_SUFFIX - Suffix for file name when SAVE_TYPE is set to NEW.

5.3 Examples of configuration

SAVE_TYPE NONE no file is saved in output.

1. OUTFILE_TYPE SAME, useful if you want just modify the header.

    a) SAVE_TYPE BACKUP: image.fits (if it exists) is renamed image.fits.back and the output file is saved as image.fits

b) SAVE_TYPE NEW: the output file is saved as image<NEW_SUFFIX>.fits

c) SAVE_TYPE REPLACE: image.fits is overwritten by the output file

2. OUTFILE_TYPE SPLIT, the input image is, in general, a multi extension file and you want to split it in single extensions.

   a) SAVE_TYPE BACKUP: single extensions are saved as image<SPLIT_SUFFIX>, if a file named image<SPLIT_SUFFIX> already exists it is renamed image<SPLIT_SUFFIX>.back

   b) SAVE_TYPE NEW: single extensions are saved as image<SPLIT_SUFFIX><NEW_SUFFIX>.fits

   c) SAVE_TYPE REPLACE: single extensions are saved as image<SPLIT_SUFFIX> and image.fits is removed

3. OUTFILE_TYPE MULTI: input images are image<SPLIT_SUFFIX> (in the command line you need only to specify the root image) and you want to build a MEF file from them. The single extensions can also be placed in a directory named image, in this case you need to specify the directory name in the command line.

   a) SAVE_TYPE BACKUP: the MEF output file is saved as image.fits, if a file named image.fits already exists it is renamed image.fits.back

   b) SAVE_TYPE NEW: output image is a MEF file named image<NEW_SUFFIX>.fits

   c) SAVE_TYPE REPLACE: MEF file is saved as image.fits and image<SPLIT_SUFFIX> are removed

4. OUTFILE_TYPE SLICE, the input image is, in general, a data-cube file and you want to unpack it in single slices.

   a) SAVE_TYPE BACKUP: single slices are saved as image<SLICE_SUFFIX>, if a file named image<SLICE_SUFFIX> already exists it is renamed image<SLICE_SUFFIX>.back

   b) SAVE_TYPE NEW: bi-dimensional images are saved as image<SLICE_SUFFIX><NEW_SUFFIX>.fits

   c) SAVE_TYPE REPLACE: single slices are saved as image<SLICE_SUFFIX> and image.fits is removed

5. OUTFILE_TYPE CUBE: input images are image<SLICE_SUFFIX> (in the command line you need only to specify the root image) and you want to build a data-cube file from them. The single slices can also be placed in a directory named image, in this case you need to specify the directory name in the command line.

   a) SAVE_TYPE BACKUP: the data-cube output file is saved as image.fits, if a file named image.fits already exists it is renamed image.fits.back

   b) SAVE_TYPE NEW: output image is a data-cube file named image<NEW_SUFFIX>.fits

   c) SAVE_TYPE REPLACE: data-cube file is saved as image.fits and image<SLICE_SUFFIX> are removed

6. OUTFILE_TYPE DIR: input image is image.fits and you want to split or slice it in a directory named image; if your file is a MEF and you want to split it, you need to specify the option SLICE_SUFFIX NONE; if your file is a cube and you want to unpack it you need to specify the option SPLIT_SUFFIX NONE.

   a) SAVE_TYPE BACKUP: single extensions or slices are saved as image<SPLIT_SUFFIX> or image<SLICE_SUFFIX>, if a file named in the same way already exists it is renamed as .back

b) SAVE_TYPE NEW: output images are named image<SLICE_SUFFIX><NEW_SUFFIX>.fits or image<SPLIT_SUFFIX><NEW_SUFFIX>.fits

c) SAVE_TYPE REPLACE: the original file is deleted.

For example, if you want to unpack a FITS MEF datacube 843573o.fits, with four extensions, in a directory, adding to each slice raw astrometric keywords written in the .head file 843573o.head, replacing the EXPTIME keyword with TIME_OBS. In the image header the keyword ZP_0*n* specifies the photometric zero-point of the n^{th} slice in the datacube and you want a unique ZP keyword in the sliced image headers. In addition you want to compute and write the CHECKSUM keyword in the header. The configuration file will be:

```
#------------------------------ FITS keywords --------------------------------
REMOVE_KEYWORD                          # Remove a FITS keyword from the headers
REPLACE_KEYWORD     EXPTIME:TIME_OBS    # Replace a FITS keyword with another
                                        # Syntax: OLD_KEY1:NEW_KEY1,
                                        #         OLD_KEY2:NEW_KEY2,...
SLICE_KEYWORD       ZP                  # Replace the keyword
                                        # SLICE_KEYWORD+SLICEKEY_FORMAT
                                        # with SLICE_KEYWORD for every slice
                                        # or viceversa building cubes
SLICEKEY_FORMAT         _%02d           # format of slice referring keywords
DISPLAY_KEYWORD         OBJECT          # Display the following keywords while
                                        # processing the files

HEADER_SUFFIX           .head           # Filename extension for add. headers

#------------------------------ FITS properties ------------------------------

NEXTENSIONS_MIN         4               # Minimum number of extensions (warns
                                        # if less are found)
OUTFILE_TYPE            DIR             # Basic or Multi-FITS output:
                                        # "SAME", "MULTI", "SPLIT",
                                        # "SLICE", "CUBE" or "DIR"
SPLIT_SUFFIX            NONE            # Suffix expected for split FITS files
SLICE_SUFFIX            .%02d.fits      # Suffix expected for sliced FITS files

#------------------------------- FITS data  ----------------------------------

PROCESS_TYPE            NONE            # Operations on FITS data:
                                        # "NONE", "COMPRESS" or "UNCOMPRESS"

CHECKSUM_TYPE           COMPUTE         # Checksum operations:
                                        # "NONE", "COMPUTE", "VERIFY" or
                                        # "UPDATE"
#---------------------------- Output filename  -------------------------------

SAVE_TYPE               NEW             # Behaviour towards output filename:
                                        # "NONE", "BACKUP", "NEW" or "REPLACE"

NEW_SUFFIX              .miss           # suffix to add in SAVE_TYPE NEW mode
```

```
#------------------------------ Miscellaneous ---------------------------------

VERBOSE_TYPE           NORMAL          # "QUIET","NORMAL" or "FULL"
WRITE_XML              Y               # Write XML file (Y/N)?
XML_NAME               missfits.xml    # Filename for XML output
XSL_URL                file:///usr/local/share/missfits/missfits.xsl
```

References

[1] Bertin E., SWarp, User's manual, 1999-2003, IAP

[2] Bertin E., SCAMP, User's manual, 2002-2010, IAP