The MUSE Data Reduction Software Manual

TANYA URRUTIA
Ole STREICHER
PETER WEILBACHER
CHRISTER SANDIN

September 3, 2015
# Contents

1 Introduction 5
   1.1 Scope of this Document 5
   1.2 Stylistic Conventions 5
   1.3 Abbreviations and Acronyms 5

2 Instrument and Data Design 7
   2.1 The MUSE Instrument 7
   2.2 Data Layout 7

3 MUSE Pipeline Installation 11
   3.1 Installation of Python-CPL 11

4 Pipeline and Recipe Description 13
   4.1 Recipe Description 13

5 Reduction Cookbook - EsoRex 16
   5.1 Basic Reduction 17
      5.1.1 Identification of raw input files 18
      5.1.2 Bias 18
      5.1.3 Dark 19
      5.1.4 Flat and Trace Table 20
      5.1.5 Wavelength Calibration 21
      5.1.6 LSF calculation 22
      5.1.7 Instrument Geometry 23
      5.1.8 Twilight 25
      5.1.9 Basic science processing 26
   5.2 Post-Processing 27
      5.2.1 Standard Star and Flux Calibration 27
      5.2.2 Astrometry 29
      5.2.3 Sky Creation and Subtraction 30
6 Reduction Cookbook - Python-CPL

6.1 Basic Reduction
6.1.1 Identification of raw input files
6.1.2 Bias
6.1.3 Dark
6.1.4 Flat and Trace Table
6.1.5 Wavelength Calibration
6.1.6 LSF calculation
6.1.7 Instrument Geometry
6.1.8 Skyflat
6.1.9 Basic science processing

6.2 Post-Processing
6.2.1 Standard Star and Flux Calibration
6.2.2 Astrometry
6.2.3 Sky Creation and Subtraction
6.2.4 Science Post-Processing and Final Datacube
6.2.5 Combine Exposures

7 Recipe Parameters

7.1 Pre-processing recipes
7.1.1 muse_bias
7.1.2 muse_dark
7.1.3 muse_flat
7.1.4 muse_wavecal
7.1.5 muse_lsf
7.1.6 muse_twilight
7.1.7 muse_geometry
7.1.8 muse_scibasic

7.2 Post-processing recipes
7.2.1 muse_standard
7.2.2 muse_create_sky
7.2.3 muse_astrometry
7.2.4 muse_scipost
7.2.5 muse_exp_align
7.2.6 muse_exp_combine
8 Tips & Troubleshooting

8.1 The output of the logfile .................................................. 89
8.2 Restricting wavelength ranges ........................................... 89
8.3 Debugging options ............................................................. 89
8.4 Tools for debugging and verification ................................... 90
  8.4.1 Verification of the tracing solution ............................... 90
  8.4.2 Verification of the wavelength solution ......................... 92
  8.4.3 Handling of MUSE pixel tables ..................................... 93
  8.4.4 Handling of MUSE bad pixel maps .................................. 95
  8.4.5 Tools to deal with (partial) output cubes ....................... 96
  8.4.6 Other tools ............................................................... 97
8.5 Typical failure cases ....................................................... 97
  8.5.1 Failed tracing ............................................................ 97
8.6 Correcting coordinate offsets ............................................ 98
8.7 Correcting relative fluxes ................................................ 99

A Data Format Description ................................................. 100

A.1 Raw data files ............................................................. 100
  A.1.1 RAW_IMAGE .......................................................... 100
A.2 External calibration files ............................................... 101
  A.2.1 LINE_CATALOG ....................................................... 101
  A.2.2 SKY_LINES ............................................................ 102
  A.2.3 ASTROMETRY_REFERENCE ......................................... 103
  A.2.4 EXTINCT_TABLE ...................................................... 103
  A.2.5 BADPIX_TABLE ....................................................... 104
  A.2.6 STD_FLUX_TABLE ................................................... 104
  A.2.7 FILTER_LIST .......................................................... 105
  A.2.8 TELLURIC_REGIONS ................................................ 105
A.3 Data files created by the pipeline .................................... 106
  A.3.1 MUSE_IMAGE .......................................................... 106
  A.3.2 PIXEL_TABLE .......................................................... 107
  A.3.3 DATA_CUBE ........................................................... 109
  A.3.4 EURO3D_CUBE ........................................................ 110
  A.3.5 TRACE_TABLE .......................................................... 111
  A.3.6 TRACE_SAMPLES ..................................................... 112
  A.3.7 WAVECAL_TABLE ..................................................... 112
  A.3.8 WAVECAL_RESIDUALS ............................................... 113
  A.3.9 LSF_PROFILE .......................................................... 113
A.3.10 GEOMETRY_TABLE .............................................. 114
A.3.11 SPOTS_TABLE ............................................... 115
A.3.12 FLUX_TABLE .................................................. 116
A.3.13 STD_RESPONSE ............................................. 116
A.3.14 STD_TELLURIC .............................................. 117
A.3.15 STD_FLUXES ............................................... 117
A.3.16 AMPL_CONVOLVED ....................................... 118
A.3.17 OFFSET_LIST ............................................... 118
A.4 Other data files ................................................. 119
   A.4.1 OUTPUT_WCS ............................................. 119
Chapter 1

Introduction

1.1 Scope of this Document

This document contains information that lets users reduce raw MUSE data into finished, science-ready data cubes using the MUSE Data Reduction Software (DRS, also called the pipeline) developed at AIP. It is designed as a User Manual for people in the MUSE consortium. Currently, it is not foreseen that scientific end users will work with this manual. However, to learn of the recipes and the processes happening behind the scenes, this document might be of interest to all people working with MUSE data. The final pipeline provided by ESO may contain parts of this cookbook. The overall MUSE data reduction procedure and recipes are described in Chapter 4. It is foreseen that the person performing the reduction will be able to do this with the standard settings following the cookbook instructions referenced in Chapter 5, using EsoRex; this is also the preferred way of ESO to reduce the data. If you are more comfortable working with Python, you might be interested in using the Python-CPL interface described in Chapter 6 after reading about the EsoRex reduction. This package is a module that calls the CPL recipes that are described in Chapter 5, the data then are already in a format that are easily handled by Astropy or your favorite Python recipes. Recipe options and corresponding parameters are described in Chapter 7, to allow the user to configure his/her EsoRex or Python CPL script to his/her liking.

Note that the MUSE data reduction is very resource intensive on your computer architecture, especially on memory. It is not recommended that data be reduced on personal computer, rather a multi-core workstation with at the very least 32 GB RAM is recommended. However, for the end-cube combination a machine with at least 150 GB RAM is needed.

1.2 Stylistic Conventions

muse_bias is a recipe of the MUSE pipeline. MUSE functions within the code will be treated equally.

/home/user> esorex muse_bias bias.sof is a command line prompt command or a part of a code. Arguments within the code will be treated equally. It is assumed that the the reductions are made in the working directory, '/home/user' is merely a placeholder for the purposes of this document.

Note: 'something important' is an author’s note, either indicating that something will likely change or as a reminder of implementation and/or merging of the section with the DRS pipeline design document.

1.3 Abbreviations and Acronyms
<table>
<thead>
<tr>
<th>Short</th>
<th>Long</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO</td>
<td>Adaptive Optics</td>
</tr>
<tr>
<td>ADU</td>
<td>Analogue to Digital Unit</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>CPL</td>
<td>Common Pipeline Library</td>
</tr>
<tr>
<td>DRS</td>
<td>Data Reduction Pipeline</td>
</tr>
<tr>
<td>ESO</td>
<td>European Southern Observatory</td>
</tr>
<tr>
<td>EsoRex</td>
<td>ESO Recipe Execution Tool</td>
</tr>
<tr>
<td>FITS</td>
<td>Flexible Image Transfer System</td>
</tr>
<tr>
<td>FWHM</td>
<td>Full Width Half Maximum</td>
</tr>
<tr>
<td>FOV</td>
<td>field of view</td>
</tr>
<tr>
<td>GCC</td>
<td>GNU Compiler Collection</td>
</tr>
<tr>
<td>IFU</td>
<td>Integral Field Unit</td>
</tr>
<tr>
<td>LSF</td>
<td>Line Spread Function</td>
</tr>
<tr>
<td>MUSE</td>
<td>Multi Unit Spectroscopic Explorer</td>
</tr>
<tr>
<td>NFM</td>
<td>Narrow Field Mode</td>
</tr>
<tr>
<td>pixel</td>
<td>picture element (in an image)</td>
</tr>
<tr>
<td>PSF</td>
<td>Point Spread Function</td>
</tr>
<tr>
<td>spaxel</td>
<td>spatial element (of a datacube)</td>
</tr>
<tr>
<td>QC</td>
<td>Quality Control (parameters)</td>
</tr>
<tr>
<td>WCS</td>
<td>World Coordinate System</td>
</tr>
<tr>
<td>WFM</td>
<td>Wide Field Mode</td>
</tr>
<tr>
<td>voxel</td>
<td>volume element (in a datacube)</td>
</tr>
</tbody>
</table>

Table 1.1: List of Acronyms
Chapter 2

Instrument and Data Design

2.1 The MUSE Instrument

MUSE is an optical wide-field integral field spectrograph that uses the image slicing technique to cover a field of view (FOV) of 1’ x 1’ in wide-field mode (WFM) with a spatial resolution of 0.2×0.2”. The full field is split up into 24 sub-fields (each 2.5’ x 60” in WFM) which are fed into the 24 integral field units (IFUs) of the instrument. Each IFU illuminates a 4k x 4k CCD by separating the incoming light into 48 slices. For each exposure, one FITS file with 24 extensions (one for each MUSE IFU) is written to disk, which is merged in advance by the instrument control software.

In addition to the WFM, a narrow-field mode (NFM) is possible with a FOV of 7.5×7.5’ with a spatial resolution of 0.025×0.025”. The resulting layout is identical at the CCD level; the only difference in the data reduction is the different scale on the sky.

MUSE has a single spectrograph setup that covers the wavelength range 480-930 nm, at a resolution of R=3000. It is modulated only by the possible use of two filters. One is a notch filter that suppresses the wavelength region about the vNaD line at 589 nm; this is used during AO assisted observations. The second filter cuts off light in the blue part of the spectrum. This filter is used by default, but can be removed to gain access to the wavelength range 465-480 nm, at the expense of second-order overlap in the red part.

2.2 Data Layout

Figures 2.2 and 2.3 show the display of the raw image of one CCD, i.e. one of the 24 MUSE IFUs. In Figure 2.2 the presented data show a continuum flat-field exposure at IFU 19, while Figure 2.3 shows a standard star exposure at IFU 2. While one can clearly see the object traces on the slices, most of the field is empty, so that the sky spectrum dominates. Slices are approximately 76.5 pixel wide bands on the detector; the bands are separated by about 6 pixels, and are slightly curved outwards at the edges (where the deviation reaches up to 2 pixels). The bands are offset in wavelength, forming a pattern of three steps, overlaid with a curvature across the CCD, so that the wavelength coverage in each slice is different.

Read-out of the MUSE detectors is organized in a 4-port setup, where four quadrants of equal size are written for each CCD. On each chip, the vertical axis (the columns) is the dispersion direction, with the blue end of the spectrum at the lower edge. The horizontal axis (the rows) is the spatial direction. The vertical/horizontal and column/row denotation will be used in this sense throughout the document when referring to CCD positions.
Figure 2.1: Graphical representations of the splitting and slicing procedures in the MUSE instrument. The example shown is for the wide-field mode; narrow-field mode operates in the same way with a scaled-down field size. Note that the sizes are approximate, real data do not exactly cover a square region on the sky.
Figure 2.2: Flat-field for one CCD (IFU 19), intensity is coded in non-linear gray scale. Wavelength direction is vertical (red at the top).
Figure 2.3: Raw on-sky data, showing one CCD of a 120s standard star exposure in negative linear greyscale. The bright star was located in the left part of the field of view. Dispersion direction is vertical (red at the top). The continuum of the star is seen as vertical stripes and the sky emission lines as horizontal dark stripes.
Chapter 3

MUSE Pipeline Installation

The MUSE pipeline is delivered as one single tarball `muse-kit-1.1.90.tar.bz2`. This tarball contains everything that is needed to install it on a recent Linux system, including the code tarball, a tarball with calibrations, some library dependencies, this manual, and a README file. To install, just unpack the kit into a directory and issue the following command:

```
/home/user/muse-kit-1.1.90> ./install_pipeline
```

and follow the instructions.

In case you want to install only the code, unpack the `muse-1.1.90.tar.bz2` (from inside the kit) and look into the `README` in that tarball for instructions.

3.1 Installation of Python-CPL

If you want to call the MUSE pipeline from Python, you can use the optional Python-CPL package that comes with the source code, see the subdirectory `muse-1.1.90/python/`). Alternatively, the source code may be downloaded from the Python Package Index web page ([https://pypi.python.org/packages/source/p/python-cpl/](https://pypi.python.org/packages/source/p/python-cpl/)) or accessed through its git repository. The latter option is preferred, as it ensures that you will get the latest version of the module. Issue the following command to get the source code from the repository:

```
/home/user> git clone git://github.com/olebole/python-cpl.git
```

This creates a subdirectory named `python-cpl/` with the most recent version of the module. To update to the current version of an existing repository, issue the command `git pull` in the `python-cpl/` directory.

The Python-CPL module has the following prerequisites to work correctly:

- Python 2.6, 2.7, 3.3, or 3.4
- Astropy ([http://www.astropy.org/](http://www.astropy.org/)) or Pyfits ([http://www.pyfits.org](http://www.pyfits.org)). Note that the latter is deprecated.

In the source directory, compile the package with the following command:

```
/home/user/python-cpl> python setup.py install --prefix=PREFIX
```
The `prefix` option denotes an optional installation path of the program; as a default the directory `/usr/local/` is used. If you change it, make sure you add the directory `PREFIX/lib/python2.7/site-packages/` (or `PREFIX/lib64/python2.7/site-packages/` on 64 bit systems) to your environment variable `PYTHONPATH`, where `PREFIX` is the installation path for the package. The Python-CPL module is now installed and you can continue with the cookbook in Section 6!

As an optional step, you can now execute some tests to see if the module was installed correctly. Issue the following commands to initiate the tests:

```
/home/user/python-cpl> cd test/
/home/user/python-cpl/test> python TestRecipe.py
```

The program will then run a variety of tests, which hopefully all pass. The tests may print a memory corruption detection by glibc. This is normal, since the tests also check the behavior of this behavior in the recipe.
Chapter 4

Pipeline and Recipe Description

The main MUSE pipeline is divided into several calibration recipes (handling both instrument-internal and on-sky calibration data), and two main science reduction recipes. Each of these recipes is composed of a few up to many functions implemented in the MUSE data-reduction library. At the moment, the layout into a few high-level recipes is geared towards a user who reduces data mostly automatically using esorex. Given the data volume and data complexity of MUSE, several steps during data reduction can take a long time. It is assumed the automated approach, which is based on a few recipes, will therefore be the most used reduction mode. Some additional recipes exist that allow a more fine-grained step-by-step reduction, but these are not yet well enough tested and are not documented here.

The data processing is split into two parts: 1. the basic reduction including calibrations, which works on the basis of single CCDs and determines and/or removes the signature of each IFU, and 2. a set of recipes that post-process the pre-reduced data into useful scientific output, thereby working on data of all CCDs of one or more exposures. In this part of the process, the pipeline works with pixel tables before combining them to a final Datacube. The two different parts are visually decomposed into the recipes in Figures 4.1 and 4.2.

4.1 Recipe Description

This section will give a basic description of the algorithms used in each recipe. To be done concurrently with the MUSE pipeline paper by P. Weilbacher.
Figure 4.1: Association map for the basic science data reduction. This diagram shows the part of the pipeline that operates on the basis of a single IFU.
Figure 4.2: Association map for the second part of the science data reduction. This part of the pipeline deals with data of all 24 IFUs simultaneously; data in this diagram start with \texttt{PIXTABLE\_type} just as the output data that are shown in the association map in Figure 4.1.
Chapter 5

Reduction Cookbook - EsoRex

EsoRex is a powerful parser that allows you to call a given recipe with a set of frames (sof) as input parameters (see below). Moreover, you can pass values to the different parameters of each recipe via command line options or via a configuration file. Any information on EsoRex that is beyond the scope of this cookbook can be gained from the EsoRex web pages (http://www.eso.org/sci/software/cpl/esorex.html). Make sure that EsoRex is in your executable path; when you install it, you could also set up an alias such as:

    alias esorex=${ESOREX_DIR}/bin/esorex

in your setup file (.tcshrc or similar). To set up your EsoRex configuration file after installation, please issue:

    /home/user> esorex --create-config

Make sure that the created configuration file (~/.esorex/esorex.rc) contains at least the line:

    esorex.caller.recipe-dir=${MUSE_DIR}

so that the recipes are found and executed.

Recipes are usually called with EsoRex as follows:

    /home/user> esorex [esorex-options] [recipe [recipe-options] [sof]]

Notice that after the command itself, all the command-line arguments are grouped according to their function. The EsoRex options come first. The command

    /home/user> esorex --help

lists all the command-line options for the EsoRex application itself. The recipe may optionally be specified. The command

    /home/user> esorex --help recipe

gives you the help screen on any recipe. The relevant recipes and their options for MUSE are listed in Chapter 7.
Any command line options for the recipe itself are specified following the recipe name and a sof file. (Note that it is possible to list several sof files, in which case EsoRex will treat them as if they were appended in a single file.)

A sof (set of files) file contains a list of the input data in plain text format, where each input file is specified with its associated classification and category. The format of each line in the sof file is as follows:

```
full-path-to-file  classification
```

where the different classifications are specified below. You need to create these sof files either manually with your favorite text editor (e.g. emacs or vi), or they need to be created within a script. For the purposes of this document, we assume that the user has already created these sof files and they already contain all the relevant information. An example MUSE sof file might look like this:

```
/home/user/data/raw/MUSE.2013-12-26T01:05:06.233.fits OBJECT
/home/user/data/cal/MASTER_BIAS.fits MASTER_BIAS
/home/user/data/cal/MASTER_FLAT.fits MASTER_FLAT
/home/user/data/cal/WAVECAL_TABLE WAVECAL_TABLE
/home/user/data/cal/geometry_table.fits GEOMETRY_TABLE
```

Screenshots of the results are shown throughout the cookbook. The program DS9 ([http://hea-www.harvard.edu/RD/ds9/site/Home.html](http://hea-www.harvard.edu/RD/ds9/site/Home.html)) was used to this purpose, but you are welcome to use any other FITS viewer, such as Skycat, QFitsView, etc..

The MUSE pipeline generates a lot of files and also requires quite a lot of setup. As such it is helpful to organize files into subdirectories. In the following cookbook, raw files are in their own directory category (e.g. raw/bias/ or raw/std/). Auxiliary files that belong to the reduction (also called "static" calibrations), e.g. the arc-lamp linelist or the Paranal extinction table are shipped with the pipeline and usually get installed in a directory called cal/. Log files are also stored in their own directory named LOGs/, and sof files are moved to the SOFs/ directory after they are used. You will see that the Cookbook scripts move the log files to the LOGs directory, which is why before running them, you should create it or adjust the scripts to your liking.

```
/home/user> mkdir LOGs
/home/user> mkdir SOFs
```

As mentioned in Chapter 4, the data processing is split into two parts: 1. the basic reduction including calibrations, which works on the basis of single CCDs and determines and/or removes the signature of each IFU and 2. a set of recipes to postprocess the pre-reduced data into useful scientific output, thereby working on data of all CCDs of one or more exposures. These parts are decomposed in this cookbook in Sections 5.1 and 5.2 for clarity.

If you should find that something does not work and you have access to the MPDAF websites ([http://urania1.univ-lyon1.fr/mpdaf/wiki/WikiDrs](http://urania1.univ-lyon1.fr/mpdaf/wiki/WikiDrs)), please send in a helpdesk ticket describing the nature of the problem and the version of the pipeline and the manual you are using.

### 5.1 Basic Reduction

The basic reduction sets up all parameters for the subsequent science reductions. Many master files (such as the master dark or the trace table), which will be applied over and over again, are generated during this stage of the reduction. Calibration recipes are executed on the basis of a single CCD, on an IFU per IFU basis.
5.1.1 Identification of raw input files

The name of the raw files are the usual ESO archive format: **MUSE. dateT ime.fits.fz**, where the precision of the time stamps is specified in milliseconds. Date and time stamps are derived from the date and time of the observation (exposure start), which is also stored in the header field **DATE-OBS**, e.g.

```
MUSE.2013-07-11T15:31:00.014.fits.fz
```

If present, the `.fz` extension signifies that a file was compressed using the FITS tiled image compression convention. The headers of such a file can be studied as usual for any other uncompressed FITS file, and in particular can be directly given to the MUSE pipeline without prior decompression.

The primary identification of raw input files is done using the keywords **HIERARCH ESO DPR CATG** and **HIERARCH ESO DPR TYPE** from the FITS header. See section A.1.1 for the list of possible input frames and header keywords. The Python script from section 6.1.1 can be used to sort a given list of input files in the working directory into subdirectories according to their input frame type. Other interesting keywords are **HIERARCH ESO INS MODE** and **HIERARCH ESO DET READ CURNAME**.

In the following tutorial we assume that files are sorted into subdirectories like it is done with this script.

5.1.2 Bias

We combine a set of raw bias frames into one master-bias file that is used throughout the subsequent reduction.

```
/home/user> cat bias01.sof
  raw/bias/MUSE.2014-02-11T20:31:00.123.fits BIAS
  raw/bias/MUSE.2014-02-11T20:32:07.031.fits BIAS
  raw/bias/MUSE.2014-02-11T20:33:12.932.fits BIAS
  raw/bias/MUSE.2014-02-11T20:34:18.689.fits BIAS
  raw/bias/MUSE.2014-02-11T20:35:25.162.fits BIAS
/home/user> esorex muse_bias --nifu=1 bias01.sof
/home/user> mv esorex.log LOGs/bias01.log
/home/user> mv bias01.sof SOFs/bias01.sof
```

This can be repeated for each IFU manually or using a script as shown below. At least three raw bias frames are needed in the sof file for this recipe to work correctly. The final product created with this recipe is **MASTER_BIAS-[xx].fits**, where `[xx]` is the IFU number specified with the `--nifu` option.

```
#!/bin/bash

for ifu in {01..24} ; do
  esorex muse_bias --nifu=${ifu} bias${ifu}.sof 2>&1 | \
    tee LOGs/bias${ifu}.log &
  sleep 5s
done ; wait
```

See section 7.1.1 for a full description of the **muse_bias** recipe.

---

5.1.3 Dark

We combine a set of dark frames into one master-dark file. This procedure also locates bad pixels. Note that since the current of modern CCDs is small, the master-dark frame itself is unlikely to be used further. However, the bad pixel file can be useful for the rest of the reductions.

```
/home/user> cat dark01.sof
MASTER_BIAS-01.fits MASTER_BIAS
```

```
/home/user> esorex muse_dark --nifu=1 dark01.sof
/home/user> mv esorex.log LOGs/dark01.log
```
As above, the commands can be issued for each IFU manually or script it as below. At least 3 raw dark frames are needed in the sof file for the reductions to work. The final product created here is called MASTER_DARK-[xx].fits, again the [xx] representing the IFU number currently being worked on.

```bash
#!/bin/bash
for ifu in {01..24} ; do
    esorex muse_dark --nifu=${ifu} dark${ifu}.sof 2>&1 | \ 
    tee LOGs/dark${ifu}.log &
    sleep 5s
done ; wait
```

See section 7.1.2 for a full description of the `muse_dark` recipe.

### 5.1.4 Flat and Trace Table

We combine a set of raw flat frames into one master-flat file. The procedure also locates and traces the slice locations and dark pixels.

```bash
/home/user> cat flat01.sof
raw/flat/MUSE.2014-02-11T20:34:42.493.fits FLAT
raw/flat/MUSE.2014-02-11T20:34:52.940.fits FLAT
raw/flat/MUSE.2014-02-11T20:35:03.086.fits FLAT
MASTER_DARK-01.fits MASTER_DARK
MASTER_BIAS-01.fits MASTER_BIAS
/home/user> esorex muse_flat --nifu=1 --samples flat01.sof
/home/user> mv esorex.log LOGs/flat01.log
/home/user> mv flat01.sof SOFs/flat01.sof
```

Note that the `--samples` parameter is not necessary, but it is convenient to have the extra TRACE_SAMPLES table, in case one needs to debug tracing failures (see Sect. 8.4.1).

Once again, you can run this manually for each IFU or script the process as shown below. Note that at least three raw flat frames are needed for the recipe `muse_flat` to work.

```bash
#!/bin/bash
for ifu in {01..24} ; do
    esorex muse_flat --nifu=${ifu} --samples flat${ifu}.sof 2>&1 | \ 
    tee LOGs/flat${ifu}.log &
    sleep 5s
done ; wait
```

See section 7.1.3 for a full description of the `muse_flat` recipe.
5.1.5 Wavelength Calibration

In this recipe we reduce a set of arc frames to detect arc emission lines and to determine the wavelength solution for each file. The three available lamps are combined to ensure a smooth wavelength solution across the entire range. Only one raw arc frame is required, but one should have at least one frame for all three lamps for a good solution across the full MUSE wavelength range.

```
/home/user> cat wavecal01.sof
  raw/arc/MUSE.2014-02-11T20:35:15.782.fits ARC
  raw/arc/MUSE.2014-02-11T20:35:28.534.fits ARC
  TRACE_TABLE-01.fits TRACE_TABLE
  MASTER_BIAS-01.fits MASTER_BIAS
```
MASTER_FLAT-01.fitd  MASTER_FLAT
MASTER_DARK-01.fits  MASTER_DARK
  cal/linelist_master.fits  LINE_CATALOG
/home/user>  esorex muse_wavecal  --nifu=1  --residuals wavecal01.sof
/home/user>  mv esorex.log  LOGs/wavecal01.log
/home/user>  mv wavecal01.sof  SOFs/wavecal01.sof

Again, note that the --residuals parameter is not necessary, but it may be convenient to have the extra WAVECAL_RESIDUALS table, in case one wants to verify the wavelength solution (see Section 8.4.2).

As before, the script below does the processing for all IFUs in parallel.

```bash
#!/bin/bash
for ifu in {01..24}  ; do
  esorex muse_wavecal  --nifu=${ifu}  --residuals wavecal${ifu}.sof 2>&1 | \
    tee LOGs/wavecal${ifu}.log &
  sleep 5s
done ; wait
```

See section 7.1.4 for a full description of the muse_wavecal recipe.

### 5.1.6 LSF calculation

If one is planning to subtract the sky from the data later, one needs a representation of the line spread function (LSF). This is computed by the muse_lsf recipe which works by analyzing the arc lines.

```
/home/user>  cat lsf01.sof
raw/arc/MUSE.2014-02-11T20:35:15.782.fits  ARC
raw/arc/MUSE.2014-02-11T20:35:28.534.fits  ARC
TRACE_TABLE-01.fits  TRACE_TABLE
  cal/linelist_master.fits  LINE_CATALOG
MASTER_BIAS-01.fits  MASTER_BIAS
WAVECAL_TABLE-01.fits  WAVECAL_TABLE
/home/user>  esorex muse_lsf  --nifu=1  --save_subtracted lsf01.sof
/home/user>  mv esorex.log  LOGs/lsf01.log
/home/user>  mv lsf01.sof  SOFs/lsf01.sof
```

Here, one should make sure that one has a number of exposures per arc lamp, ideally at least 10, so that the faint wings of the line profiles can be measured with reasonable S/N.

```
#!/bin/bash
for ifu in {01..24}  ; do
  esorex muse_lsf  --nifu=${ifu}  --save_subtracted lsf${ifu}.sof 2>&1  |  \
  tee LOGs/lsf${ifu}.log  &
```
Figure 5.3: Resampled image of the ARC image after wavelength calibration was applied. Note that all lines are now on the same wavelength on uninterrupted horizontal lines.

```bash
    sleep 5s
done ; wait
```

See section 7.1.5 for a full description of the `muse_lsf` recipe.

### 5.1.7 Instrument Geometry

This recipe needs a very long special exposure sequence and care has to be taken to check the data beforehand and afterwards. It should normally be run only by experts.

The instrument geometry contains information on where within the field of view each slice of each IFU is located. It assigns an initial position on the sky for each CCD pixel.
This recipe needs at least the full special exposure sequence as input, as well as master-bias files, the wavelength calibration, trace tables for all IFUs, and a specially prepared line list with only a few bright calibration lines in it. It can make use of extra exposures with different structured content to check its calibration. Master darks and flat-fields can be input, but this is optional and should only be done if the recipe does not otherwise work. Running the recipe does not usually require any parameters as the IFU number.

This recipe does its work in parallel on multiple threads, loading all input data simultaneously, unless the user restricts the number of threads to below 24 (the environment variable OMP_NUM_THREADS should be used for this purpose); only a fraction of the IFU data is loaded at the same time if the number of available threads is less than 24.

```
/home/user> cat geo.sof
raw/geo/MUSE_WFM_WAVE213_0010.fits MASK
raw/geo/MUSE_WFM_WAVE213_0011.fits MASK
raw/geo/MUSE_WFM_WAVE213_0012.fits MASK
raw/geo/MUSE_WFM_WAVE213_0013.fits MASK
raw/geo/MUSE_WFM_WAVE213_0014.fits MASK
raw/geo/MUSE_WFM_WAVE213_0015.fits MASK
[... 70 rows omitted ...]
raw/geo/MUSE_WFM_WAVE213_0086.fits MASK
raw/geo/MUSE_WFM_WAVE213_0087.fits MASK
raw/geo/MUSE_WFM_WAVE213_0088.fits MASK
raw/geo/MUSE_WFM_WAVE213_0089.fits MASK
MASTER_BIAS-01.fits MASTER_BIAS
MASTER_BIAS-02.fits MASTER_BIAS
[... 21 rows omitted ...]
MASTER_BIAS-24.fits MASTER_BIAS
TRACE_TABLE-01.fits TRACE_TABLE
TRACE_TABLE-02.fits TRACE_TABLE
[... 21 rows omitted ...]
TRACE_TABLE-24.fits TRACE_TABLE
WAVECAL_TABLE-01.fits WAVECAL_TABLE
WAVECAL_TABLE-02.fits WAVECAL_TABLE
[... 21 rows omitted ...]
WAVECAL_TABLE-24.fits WAVECAL_TABLE
cal/linelist_master.fits LINE_CATALOG
raw/geo/MUSE_IQE_MASK213_0001.fits MASK_CHECK
```

(The extra switches for EsoRex are there to speed up processing. The checksums are not needed on all intermediate calibrations that this recipe saves, but only on the final table. There they can be set using the muse_product_sign tool.)

The muse_geo_plot tool can be used to create a graphical representation of the resulting GEOMETRY_TABLE in PNG or PDF format (see Fig. 5.4):

```
/home/user> muse_geo_plot -f png -s 4 4 GEOMETRY_TABLE.fits GEOMETRY_TABLE.png
```
The **GEOMETRY_CUBE** output cube as well as any **GEOMETRY_CHECK** output images should be used to visually verify the quality of the calibration.

See section 7.1.7 for a full description of the **muse_geometry** recipe.

### 5.1.8 Twilight

If the observations included twilight sky flat-fields, then this step combines them into a three-dimensional illumination correction. The cube produced here also propagates the integrated flux found in the FITS header to the science data, to even out throughput differences between the IFUs.

At least three input skyflat exposures are required for this recipe to run. Contrary to most other basic processing recipes, this needs input data from all 24 IFUs to produce useful results (but for brevity, not all files are shown):

```
/home/user> cat twilight.sof
    raw/SkyCalib/MUSE.2014-02-12T06:15:22.033.fits SKYFLAT
    raw/SkyCalib/MUSE.2014-02-12T06:18:00.228.fits SKYFLAT
    raw/SkyCalib/MUSE.2014-02-12T06:20:58.877.fits SKYFLAT
    MASTER_BIAS-01.fits MASTER_BIAS
    MASTER_BIAS-02.fits MASTER_BIAS
    [... 21 rows omitted ...]
```
The input VIGNETTING_MASK is optional, but may be used to correct the vignetting that usually affects MUSE data in the lower right corner of the field of view.

The main output for this recipe is TWILIGHT_CUBE.fits, the additional DATACUBE_SKYFLAT.fits is the cube with the resampled data, without any modeling applied.

This recipe should produce reasonable results without parameters, but as always, please refer to section 7.1.6 for a full description of the muse_twilight recipe.

5.1.9 Basic science processing

When all the master calibration files are available, the basic science processing procedure removes the instrumental signature from the data of each of the on-sky exposures and converts them from the FITS image format to a FITS-based pixel table format.

If an illumination flat-field was observed within the ±1 hour around the target observations (or an attached flat-field during the night and similarly close in time), then it’s recommended to pass this exposure as raw input file ILLUM to muse_scibasic, to get per-slice illumination correction:
This is the last recipe that must be called for each IFU separately. The automated script to cycle through all 24 IFUs is shown below. In most cases, the 2D-resampled image is not needed. We switch it off here:

```bash
#!/bin/bash
for ifu in {01..24}; do
    esorex muse_scibasic --nifu=${ifu} --resample=False \ 
    scibasicillum${ifu}.sof
    tee LOGs/scibasicillum${ifu}.log &
    sleep 5s
done ; wait
```

The basic reduction is finished with the creation of the pre-reduced pixel tables (these are the files that are named `PIXTABLE_*`). These pixel tables will now run through (some of) the more complicated post-processing recipes, such as flux calibration and sky subtraction, before creating the final cubes.

For the user to check whether all basic reduction procedures were successful, a reduced image of the CCD (`OBJECT_RED_*`) and a resampled image using tracing and wavelength calibration solutions (`OBJECT_RESAMPLED_*`). Usually, neither of them are needed, so that the user can skip the `--resample` parameter and instead set `--saveimage=false`, to save space.

See section 7.1.8 for a full description of the `muse_scibasic` recipe.

## 5.2 Post-Processing

This section covers the observing-dependent reduction from the pixel tables to the final datacubes. Not every step is mandatory, but the last recipe `muse_scipost` is needed to create the cubes.

The post-processing part of the MUSE Data Reduction works on the pixel tables rather than images, before the actual reconstruction into datacubes.

### 5.2.1 Standard Star and Flux Calibration

Here we create a flux response curve to flux calibrate the science data using exposures of a standard star. We need to remove the instrumental signature not only from the science observations, but also from the standard star observations. The standard star exposures consequently need to be reduced to create pixel tables as other data; this is also performed individually for each IFU:
When the standard star observations are converted into pixel tables, we use these to create a sensitivity function. In the following example, we have taken all 24 pixel tables of one standard star observation:

```
/home/user> cat std.sof
PIXTABLE_STD_0001-01.fits PIXTABLE_STD
PIXTABLE_STD_0001-02.fits PIXTABLE_STD
PIXTABLE_STD_0001-24.fits PIXTABLE_STD
cal/std_flux_table.fits STD_FLUX_TABLE
cal/extinction_paranal.fits EXTINCT_TABLE
```

### This uses the default flux integration using a Moffat profile fit, but one could also choose to use circular flux integration, using `--profile=circle` (see below). Depending on the dataset, circular integration may cause less wiggles in the output response curve.

The file with the tag `STD_FLUX_TABLE` has to contain a reference table for the actual observed standard star. The table that ships with the MUSE pipeline contains usable reference curves for most stars observed with MUSE. In case a new star was observed, selection of the reference table by RA and DEC will fail and the recipe will return an error.

Here is the entire process as a script; in this case we run the standard star recipe twice, once with the default Moffat fit, once with circular integration, to be able to compare the results:

```bash
#!/bin/bash

for i in `seq -w 1 24` ; do
  ( sed "s,XX,$i," scibasic_std.sof > scibasic_std${i}.sof &&
    esorex muse_scibasic --nifu=${i} --saveimage=False \ scibasic_std${i}.sof
    mv esorex.log LOGs/scibasic_std${i}.log
    mv scibasic_std${i}.sof SOFs/scibasic_std${i}.sof
  ) &
  sleep 15s
done ; wait
```

# run flux integration with Moffat profile fits:
### 5.2.2 Astrometry

This recipe normally runs without problems, but for reduction of science data one should use a matched pair of geometry table and astrometric solution. So it's best to use provided input and hence skip this section.

Here, we will create the astrometric calibration file, which is necessary for correct relative transformation from the pixel positions to world coordinates (RA, DEC).

```bash
/home/user> cat scibasic_ast01.sof
raw/ast/MUSE.2014-02-11T21:40:02.300.fits ASTROMETRY
TRACE_TABLE-01.fits TRACE_TABLE
WAVECAL_TABLE-01.fits WAVECAL_TABLE
MASTER_BIAS-01.fits MASTER_BIAS
MASTER_DARK-01.fits MASTER_DARK
cal/geometry_table.fits GEOMETRY_TABLE
MASTER_FLAT-01.fits MASTER_FLAT

/home/user> esorex muse_scibasic --nifu=1 --saveimage=False \ 
    scibasic_ast01.sof

/home/user> mv esorex.log LOGs/scibasic_ast01.log

/home/user> mv scibasic_ast01.sof SOFs/scibasic_ast01.sof
```

(Again, we switched off saving of the pre-reduced object images.)

Once the pixel tables are pre-reduced, we can feed them into the `muse_astrometry` recipe. For this we need an `ASTROMETRY_REFERENCE` table that contains a list of stars in the field observed. A table with the typical reference targets observed with MUSE is shipped with the pipeline. (Optionally, one can input files necessary for flux calibration, but that is usually not necessary and not done here.)

```bash
/home/user> cat astrometry.sof
PIXTABLE_ASTROMETRY_0001-01.fits PIXTABLE_ASTROMETRY
PIXTABLE_ASTROMETRY_0001-02.fits PIXTABLE_ASTROMETRY
PIXTABLE_ASTROMETRY_0001-03.fits PIXTABLE_ASTROMETRY
```
One should now make sure that the computed solution is close to $0''.2$ for both axes.

See section 7.2.3 for a full description of the `muse_astrometry` recipe.

### 5.2.3 Sky Creation and Subtraction

This extra step is necessary, if we reduce data from an object that fills much of the field of view. We then would need to take an extra exposure of an (empty) sky field and create a `SKY_CONTINUUM` and an initial `SKY_LINES` table using the recipe `muse_create_sky`. (Otherwise, the sky subtraction is done directly in the `muse_scipost` recipe.) The `LSF_PROFILE` inputs produced by the `muse_lsf` recipe have to be given as well.

The input pixtable must be either flux calibrated, or the flux calibration has to be specified as calibration files with the tags `STD_RESPONSE`, `EXTINCT_TABLE` and (optionally) `STD_TELLURIC`.
[... 21 LSF_PROFILEs not shown ...]

/home/user> esorex muse_create_sky sky.sof
/home/user> mv esorex.log LOGs/sky.log
/home/user> mv sky.sof SOFs/sky.sof

See section 7.2.2 for a full description of the muse_create_sky recipe.

5.2.4 Science Post-Processing and Final Datacube

When all necessary on-sky calibrations are performed, we can start the post-processing of the science data itself, i.e. the conversion from the pre-reduced pixel tables into the final datacube.

The following example creates a cube for a single full exposure, using flux calibration, model-based sky subtraction, and astrometric calibration, using the maximum pixel fraction for the drizzle algorithm. It additionally creates four images of the field of view, integrated over four filter functions, and saves the images as extensions of the cube FITS file:

/home/user> cat scipost1.sof

PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-02.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-03.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-04.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-05.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-06.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-07.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-08.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-09.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-10.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-11.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-12.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-13.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-14.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-15.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-16.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-17.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-18.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-19.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-20.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-21.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-22.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-23.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
cal/astrometry_wcs.fits ASTROMETRY_WCS
cal/filters.fits FILTER_LIST
cal/sky_lines.fits SKY_LINES
cal/extinction_paranal.fits EXTINCT_TABLE
std/STD_RESPONSE_moffat.fits STD_RESPONSE
cal/LSF_PROFILE-01.fits LSFPROFILE
-cal/LSF_PROFILE-02.fits LSF_PROFILE

31
If more than one exposure is processed separately in this way, the output files have to be renamed to not be overwritten:

```
/home/user> mv -v DATACUBE_FINAL.fits DATACUBE_FINAL_01.fits
/home/user> mv -v PIXTABLE_REduced_0001.fits PIXTABLE_REduced_e01.fits
```

Note: this processing step requires a lot of RAM, approximately 18 GB per exposure are needed to successfully run it.

The next example creates a cube for three exposures, with all on-sky calibrations, now using the default pixel fraction for drizzle. The exposures are automatically aligned using their WCS information, and reconstructed into a single cube. To improve the cosmic ray rejection, this now uses the "median" method with a more stringent sigma rejection level. It also saves the three pixel tables of the different exposures after post-processing but before exposure combination. Not all input files are printed:

```
/home/user> cat scipost3.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-02.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0001-03.fits PIXTABLE_OBJECT
[... 67 PIXTABLE_OBJECTs not shown ...]
PIXTABLE_OBJECT_0003-23.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0003-24.fits PIXTABLE_OBJECT
```
To optimize the cosmic ray rejection when combining more exposures, one generally needs to reduce the sigma level of the rejection. This may need some iteration, best done over a small wavelength range (using the \texttt{--lambdamin} and \texttt{--lambdamax} recipe parameters), to check that lower \texttt{crsigma} values do not cause object data to be rejected. Likewise, when using multiple exposures and the default drizzle algorithm, the \texttt{pixfrac} parameter could be changed to smaller values.

See section 7.2.4 for a full description of the \texttt{muse\_scipost} recipe.

### 5.2.5 Combine Exposures

While the combination of different exposure can be done already in \texttt{muse\_scipost}, it may be easier to let that recipe process each exposure separately and save its post-processed pixel table as intermediate product. This allows the user to check the invididual datacube for proper reduction before starting the last step.

It also facilitates verification or computation of relative exposure offsets (see Sect. 8.6). Briefly, if the exposures are spatially offset (affected by the “wobble”), one can use the \texttt{muse\_exp\_align} recipe to automatically compute offsets using the \texttt{IMAGE\_FOV} images corresponding to each exposure. The offsets then get saved in the \texttt{OFFSET\_LIST} which is an optional input to \texttt{muse\_exp\_combine}. The same table can also be used to provide \texttt{muse\_exp\_combine} with relative exposure flux scales (see Sect. 8.7).

The individual pixel tables of each exposure can then be input into \texttt{muse\_exp\_combine} to create the final combined datacube. As with \texttt{muse\_scipost}, one could try to set the parameters \texttt{crsigma} and \texttt{pixfrac} to smaller values the more overlapping exposures are part of the dataset.

The following example creates the same cube as in the three-exposure example in 5.2.4, but starts from individual pixel tables:

```bash
/home/user> cat expcomb.sof
PIXTABLE\_REDUCED\_e01.fits PIXTABLE\_REDUCED
PIXTABLE\_REDUCED\_e02.fits PIXTABLE\_REDUCED
PIXTABLE\_REDUCED\_e03.fits PIXTABLE\_REDUCED
/home/user> esorex muse\_exp\_combine
```
--filter=white,Johnson_V,Cousins_R,Cousins_I --format=xCube \
expcomb.sof

/home/user> mv esorex.log LOGs/expcomb.log
/home/user> mv expcomb.sof SOFs/expcomb.sof

This needs far fewer inputs, since the reduced individual pixel tables are already fully calibrated. Note: as above, this example needs approximately 55 GB of free RAM to finish!

See section 7.2.6 for a full description of the muse_exp_combine recipe.
Chapter 6

Reduction Cookbook - Python-CPL

The CPL interface is a Python module to access CPL recipes from Python. If you are used to programming in Python and already have your analysis software coded in Python, you should consider calling up the data reduction recipes for MUSE using this handy module. That way you will not have to leave the familiar environment and will have the output data already loaded within the Astropy module.

The Home page for the Python CPL module is https://pypi.python.org/pypi/python-cpl. These pages are much more detailed than this Cookbook, so please consider reading through them first, should you have any questions regarding installation or the running of the module.

The tutorial to fully reduce MUSE data is presented within a Python session. We will usually be working within the Python window (denoted by >>>, which is called by:

```
/home/user> python
Python 2.7 (r27:82500, Aug 07 2010, 16:54:59) [GCC] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```  

Of course, you can also use iPython or just run the commands as a .py script, we will show the commands step by step for illustrative purposes and clarity.

You can check out the results, logfiles or create filelists in another window (the secondary shell window will be denoted here with /home/user> so that there is no confusion with the Python command line. This gives us the flexibility that we don’t lose any variable names or pointers by exiting Python. One of the advantages of Python-CPL is that you can immediately handle the outputs with the Astropy package and do any desired FITS file manipulation with (your) Python scripts without leaving the environment.

Before we begin the reduction process, we need to set everything up, so that the MUSE data-reduction pipeline recipes can be called within Python. We also to set up logging. Within the Python session, first import the necessary module:

```
>>> import cpl
>>> import os
>>> import sys
>>> import astropy
```

These first few commands import the different modules. The relevant one is cpl.

```
>>> cpl.esorex.init()
```
This command loads all CPL settings into Python that are usually loaded with the esorex startup file. It searches in the same directories as esorex to find CPL recipes such as the MUSE pipeline recipes, but also other ESO reduction. Ensure that all the settings that are up to date, otherwise there might be some errors noted. If you do not have esorex installed, you have to explicitly specify the location of the recipes:

```python
>>> cpl.Recipe.path = '/store/01/MUSE/recipes'
```

Once the recipe path is set, you can list all available recipes. The following command shows the name of the recipe and the version numbers:

```python
>>> cpl.Recipe.list()
[('muse_scipost', ['1.0.1', '1.1.90']),
 ('muse_scibasic', ['1.0.1', '1.1.90']),
 ('muse_flat', ['1.0.1', '1.1.90']),
 ('muse_bias', ['1.0.1', '1.1.90']),
 ('muse_dark', ['1.0.1', '1.1.90']),
 ('muse_astrometry', ['1.0.1', '1.1.90']),
 ('muse_wavecal', ['1.0.1', '1.1.90']),
 ('muse_exp_combine', ['1.0.1', '1.1.90']),
 ('muse_standard', ['1.0.1', '1.1.90']),
 ('muse_create_sky', ['1.0.1', '1.1.90'])
]
```

Next, we set up logging, so that we can see if something went wrong during the processing. A basic setup (similar to the style used in esorex) is:

```python
>>> import logging
>>> log = logging.getLogger()
>>> log.setLevel(logging.DEBUG)
>>> ch = logging.FileHandler('cpl_recipe.log')
>>> ch.setLevel(logging.DEBUG)
>>> fr = logging.Formatter('%(created)s [%(levelname)s] %(name)s: %(message)s','%H:%M:%S')
>>> ch.setFormatter(fr)
>>> log.addHandler(ch)
```

Ensure the name of the log (here named: cpl_recipe.log) is replaced with the name of the actual recipe, e.g. cpl_muse_bias.log, or something similar. In this example the logging level is set to the DEBUG, which is the lowest level. Other options are: INFO, WARN, ERROR and OFF.

### 6.1 Basic Reduction

The basic reduction sets up all parameters for the science reductions. Many master files (such as the master dark and the trace table), which are applied over and over again, are generated during this stage of the reductions. The calibration recipes are executed on the basis of single CCDs on an IFU per IFU basis.

#### 6.1.1 Identification of raw input files

The name of the raw files are the usual ESO archive file names: MUSE, date/time.fits.fz, with the precision of the time stamp indicated in milliseconds. Date and time stamp are derived from the date and time of the observation (exposure start), which is also stored in the header field DATE-OBS, for example...
MUSE.2013-07-11T15:31:00.014.fits.fz

The primary identification of raw input files is done using the keywords HIERARCH ESO DPR CATG and HIERARCH ESO DPR TYPE from the FITS header. See section A.1.1 for the list of possible input frames and header keywords. The Python script from section 6.1.1 can be used to sort a given list of input files in the working directory into subdirectories according to their type. Other interesting keywords are HIERARCH ESO INS MODE and HIERARCH ESO DET READ CURNAME.

The following Python script can be used to sort a given list of input files in the working directory into subdirectories according to their input frame type.

```python
import glob, os, pyfits

for fname in glob.glob('*.fits'):
    with pyfits.open(fname) as fits:
        d_type = fits[0].header.get('ESO DPR TYPE')
        d_catg = fits[0].header.get('ESO DPR CATG')

        dir = ('bias' if d_catg == 'CALIB' and d_type == 'BIAS' else
            'dark' if d_catg == 'CALIB' and d_type == 'DARK' else
            'flat' if d_catg == 'CALIB' and d_type == 'FLAT,LAMP' else
            'illum' if d_catg == 'CALIB' and d_type == 'FLAT,LAMP,ILLUM' else
            'ampl' if d_catg == 'TECHNICAL' and d_type == 'FLAT,LAMP,THRUPUT' else
            'arc' if d_catg == 'CALIB' and d_type == 'WAVE' else
            'mask' if d_catg == 'CALIB' and d_type == 'WAVE,MASK' else
            'skyflat' if d_catg == 'CALIB' and d_type == 'FLAT,SKY' else
            'object' if d_catg == 'SCIENCE' and d_type == 'OBJECT' else
            'sky' if d_catg == 'SCIENCE' and d_type == 'SKY' else
            'astrometry' if d_catg == 'CALIB' and d_type == 'ASTROMETRY' else
            'std' if d_catg == 'CALIB' and d_type in ('STD', 'STD,TELLU') else
            None)

        if dir is not None:
            if not os.path.exists(dir):
                os.mkdir(dir)
                os.rename(fname, os.path.join(dir, fname))
            else:
                print('Warning: cannot identify %s' % fname)

    else:
        print('Warning: cannot identify %s' % fname)
```

In the following tutorial, we assume the files are sorted in subdirectories such as it is done with this script.

### 6.1.2 Bias

We now combine the raw bias frames into one master-bias file used throughout the reductions. At least three raw bias frames are needed as input files for this recipe to work correctly. The final product created with this recipe is named MASTER_BIAS-[xx].fits, where [xx] is the IFU number specified with the nifu parameter.

```python
import cpl
```
MUSE Data Reduction Manual
Software Version: 1.1.90
Date: September 3, 2015

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/bias.log'

muse_bias = cpl.Recipe('muse_bias')
muse_bias.output_dir = '. '

for ifu in range(1, 25):
    muse_bias(["raw/bias/MUSE.2014-02-11T20:31:00.123.fits",
    'raw/bias/MUSE.2014-02-11T20:32:07.031.fits',
    'raw/bias/MUSE.2014-02-11T20:34:18.689.fits',
    'raw/bias/MUSE.2014-02-11T20:35:25.162.fits'],
    param = {'nifu': ifu})

See section 7.1.1 for a full description of the muse_bias recipe.

6.1.3 Dark

We now combine the raw dark frames to create one master dark file. This procedure also locates the bad pixels. Since the dark current of modern CCDs is small, the master dark frame itself will likely not be used further. However, the bad pixel file can be used in the rest of the reductions.

At least 3 raw dark frames are needed as input files for the reduction to work. The final product created here is called MASTER_DARK-[xx].fits, again the [xx] represents the current IFU number.

import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/dark.log'

muse_dark = cpl.Recipe('muse_dark')
muse_dark.output_dir = '. '

for ifu in range(1, 25):
    param = {'nifu': ifu},
    calib = {"MASTER_BIAS": 'MASTER_BIAS-%02i.fits' % ifu})

See section 7.1.2 for a full description of the muse_dark recipe.

6.1.4 Flat and Trace Table

In this step we combine the raw flat frames into one master flat file. We also locate and trace the slice locations and locate the dark pixels.
At least three raw flat frames are needed for the recipe `muse_flat` to work.

```python
import cpl
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/flat.log'
muse_flat = cpl.Recipe('muse_flat')
muse_flat.output_dir = '.'
muse_flat.param.samples = True

for ifu in range(1, 25):
    muse_flat(['raw/flat/MUSE.2014-02-11T20:34:42.493.fits',
               'raw/flat/MUSE.2014-02-11T20:34:52.940.fits',
               'raw/flat/MUSE.2014-02-11T20:35:03.086.fits'],
               param = {'nifu': ifu},
               calib = {'MASTER_DARK': 'MASTER_DARK-%02i.fits' % ifu,
                        'MASTER_BIAS': 'MASTER_BIAS-%02i.fits' % ifu})
```

See section 7.1.3 for a full description of the `muse_flat` recipe.

### 6.1.5 Wavelength Calibration

With this recipe we reduce the arc frames to detect arc emission lines and to determine a wavelength solution for each file. The three available lamps are combined to ensure a smooth wavelength solution across the entire range.

Only one raw arc frame is required, but one should aim to have at least one frame per lamp or a frame with all lamps on for complete wavelength coverage.

```python
import cpl
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/wavecal.log'
muse_wavecal = cpl.Recipe('muse_wavecal')
muse_wavecal.output_dir = '.'
muse_wavecal.calib.LINE_CATALOG = 'cal/linelist_master.fits'
muse_wavecal.param.residuals = True

for ifu in range(1, 25):
    muse_wavecal(['raw/arc/MUSE.2014-02-11T20:35:15.782.fits',
                  param = {'nifu': ifu},
                  calib = {'TRACE_TABLE': 'TRACE_TABLE-%02i.fits' % ifu,
                           'MASTER_BIAS': 'MASTER_BIAS-%02i.fits' % ifu,
                           'MASTER_FLAT': 'MASTER_FLAT-%02i.fits' % ifu,
                           'MASTER_DARK': 'MASTER_DARK-%02i.fits' % ifu})
```
See section 7.1.4 for a full description of the muse_wavecal recipe.

### 6.1.6 LSF calculation

If one plans to subtract the sky from the data later, one needs a representation of the line spread function (LSF). This is computed by the muse_lsf recipe, which works by analyzing the arc lines.

Here, one should ensure that one has a number of exposures per arc lamp, ideally at least 10, so that the faint wings of the line profiles can be measured with reasonably high S/N.

```python
import cpl
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/lsf.log'
muse_lsf = cpl.Recipe('muse_lsf')
muse_lsf.output_dir = '.
muse_lsf.calib.LINE_CATALOG = 'cal/linelist_master.fits'
muse_lsf.param.save_subtracted = True

for ifu in range(1, 25):
    muse_lsf(['raw/arc/MUSE.2014-02-11T20:35:15.782.fits',
              param = {'nifu': ifu},
              calib = {'TRACE_TABLE': 'TRACE_TABLE-%02i.fits' % ifu,
                        'MASTER_BIAS': 'MASTER_BIAS-%02i.fits' % ifu,
                        'WAVECAL_TABLE': 'WAVECAL_TABLE-%02i.fits' % ifu})
```

See section 7.1.5 for a full description of the muse_lsf recipe.

### 6.1.7 Instrument Geometry

*This recipe needs a very long special exposure sequence and care has to be taken to check the data beforehand and afterwards. It should normally be used only by experts.*

The instrument geometry provides information on where within the field of view each slice of each IFU is located. Each CCD pixel is assigned an initial position on the sky.

This recipe needs at least the full special exposure sequence, master-bias files, wavelength calibration and trace tables for all IFUs, and a special line list with only a few lines in it. It can make use of extra exposures with different structured content for checking its calibration. Master darks and flat fields can be input, but this is optional and should only be done if it does not otherwise work. Usually the recipe does not require any parameters.

This recipe does its work in parallel, loading all of the input data at the same time, unless the user restricts the number of threads to below 24 (the variable OMP_NUM_THREADS should be used for this purpose).

```python
import cpl
cpl.esorex.init()
```
Figure 6.1: Visual representation of a GEOMETRY_TABLE, produced with the muse_geo_plot tool.

cpl.esorex.log.file = 'LOGs/geometry.log'
muse_geometry = cpl.Recipe('muse_geometry')
muse_geometry.output_dir = '.,'
muse_geometry.calib.MASTER_BIAS = [('calib/MASTER_BIAS-%02i.fits' % ifu) for ifu in range(1, 25)]
muse_geometry.calib.TRACE_TABLE = [('calib/TRACE_TABLE-%02i.fits' % ifu) for ifu in range(1, 25)]
muse_geometry.calib.WAVECAL_TABLE = [('calib/WAVECAL_TABLE-%02i.fits' % ifu) for ifu in range(1, 25)]
muse_geometry.calib.LINE_CATALOG = 'cal/linelist_master.fits'
muse_geometry.calib.MASK_CHECK = mask/MUSE_IQE_MASK213_0001.fits'
muse_geometry.env['OMP_NUM_THREADS'] = '12'
muse_geometry(('[mask/MUSE_WFM_WAVE213_%04i.fits' % i) for i in range(10, 90)])

After it finished, one can use the tool muse_geo_plot to create a graphical representation of the resulting GEOMETRY_TABLE in PNG or PDF format (see Fig. 6.1):

/home/user> muse_geo_plot -f png -s 4 4 GEOMETRY_TABLE.fits GEOMETRY_TABLE.png

One could also look at the GEOMETRY_CUBE output cube as well as any GEOMETRY_CHECK output images to visually verify the quality of the calibration.
See section 7.1.7 for a full description of the `muse_geometry` recipe.

### 6.1.8 Skyflat

If the observations included twilight sky flat-fields, this step combines them into a three-dimensional illumination correction. The cube produced here also propagates the integrated flux found in the FITS header to the science data, to even out throughput differences between the IFUs.

At least three input sky flat-field exposures are required for this recipe to run. Contrary to most other basic processing recipes, this needs input data from all 24 IFUs to produce useful results (for brevity, not all files are shown):

```python
import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/twilight.log'
muse_twilight = cpl.Recipe('muse_twilight')
muse_twilight.output_dir = '.'
muse_twilight.calib.MASTER_BIAS = [('calib/MASTER_BIAS-%02i.fits' % ifu)
for ifu in range(1, 25)]
muse_twilight.calib.MASTER_FLAT = [('calib/MASTER_FLAT-%02i.fits' % ifu)
for ifu in range(1, 25)]
muse_twilight.calib.TRACE_TABLE = [('calib/TRACE_TABLE-%02i.fits' % ifu)
for ifu in range(1, 25)]
muse_twilight.calib.WAVECAL_TABLE = [('calib/WAVECAL_TABLE-%02i.fits' % ifu)
for ifu in range(1, 25)]
muse_twilight.calib.GEOMETRY_TABLE = 'cal/geometry_table.fits'
muse_twilight.calib.VIGNETTING_MASK = 'cal/vignetting_mask.fits'
muse_twilight(['raw/SkyCalib/MUSE.2014-02-12T06:15:22.033.fits SKYFLAT',
'raw/SkyCalib/MUSE.2014-02-12T06:18:00.228.fits SKYFLAT',
...
'raw/SkyCalib/MUSE.2014-02-12T06:20:58.877.fits SKYFLAT'])
```

The input `VIGNETTING_MASK` is optional, but may be used to correct the vignetting that usually affects MUSE data in the lower right corner of the field of view. The main output is `TWILIGHT_CUBE.fits`. The additional `DATA CUBE_SKYFLAT.fits` is the cube with the resampled data, without any modeling applied.

This recipe should produce reasonable results without the specification of additional parameters; please see to section 7.1.6 for a full description of the `muse_twilight` recipe.

### 6.1.9 Basic science processing

This procedure removes the instrumental signature from the data of each of the standard star and science CCD images and converts them from FITS-image to a pixel-table format.

If an illumination flat-field image was observed within one hour around the target observations (or an attached flat-field image during the night and similarly close in time), it’s recommended to pass this exposure as a raw input file to `muse_scibasic (ILLUM)`, to get a per-slice illumination correction:

The automated script to cycle through all 24 IFUs is shown below. The 2D-resampled image is not needed in most cases and it is switched off here:
import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/scibasicillum.log'
muse_scibasic = cpl.Recipe('muse_scibasic')
muse_scibasic.output_dir = '.'
muse_scibasic.calib.GEOMETRY_TABLE = 'cal/geometry_table.fits'
muse_scibasic.param.resample = False

for ifu in range(1, 25):
    muse_scibasic({'ILLUM': ['raw/object/MUSE.2014-02-11T20:59:35.367.fits'],
                   'OBJECT': ['raw/object/MUSE.2014-02-11T20:35:50.720.fits']},
                   param = {'nifu': ifu},
                   calib = {'TRACE_TABLE': 'TRACE_TABLE-%02i.fits' % ifu,
                            'MASTER_BIAS': 'MASTER_BIAS-%02i.fits' % ifu,
                            'WAVECAL_TABLE': 'WAVECAL_TABLE-%02i.fits' % ifu,
                            'MASTER_DARK': 'MASTER_DARK-%02i.fits' % ifu,
                            'MASTER_FLAT': 'MASTER_FLAT-%02i.fits' % ifu})

The basic reduction is now finished and we have created pre-reduced pixel tables (the files named PIXTABLE_*). These pixel tables will now run through (some of) the more complicated post-processing recipes, such as flux calibration or sky subtraction, before creating the final cubes.

For the user to check whether all the basic reduction procedures were successful, a reduced image of the CCD (OBJECT_RED_*) and a resampled image using tracing and wavelength calibration solutions (OBJECT_RESAMPLED_*) were created. Usually, neither of them are needed, skip the --resample parameter and instead set --saveimage=false to save space.

See section 7.1.8 for a full description of the muse_scibasic recipe.

### 6.2 Post-Processing

This section covers the observations-dependent image construction from the pixel tables to the final 3D datacubes. The order of the steps here is not important and not every step is mandatory, but the last muse_scipost recipe is needed to create the cubes.

The Post-Processing part of the MUSE Data Reduction works on the pixel tables rather than the FITS files, before the actual image reconstruction into FITS Datacubes. The recipes in this part construct the images based on observations-dependent conditions.

#### 6.2.1 Standard Star and Flux Calibration

In this step we create a flux response curve for overall flux calibration of the science images using a standard star. As such, we need to remove the instrumental signature not only from the science observations, but also from the standard-star observations. The standard star needs to undergo the basic reduction to pixel tables as described in the “Scibasic” Section (see 6.1.9). As before, this is still performed on a per IFU basis. You can script this process for all IFUs.
import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/scibasic_std.log'

muse_scibasic = cpl.Recipe('muse_scibasic')
muse_scibasic.output_dir = '.
muse_scibasic.calib.GEOMETRY_TABLE = 'cal/geometry_table.fits'
muse_scibasic.param.saveimage = False

for ifu in range(1, 25):
    muse_scibasic('raw/std/MUSE.2014-02-11T20:36:01.300.fits',
        param = {'nifu': ifu},
        calib = {'TRACE_TABLE': 'TRACE_TABLE-%02i.fits' % ifu,
                 'MASTER_BIAS': 'MASTER_BIAS-%02i.fits' % ifu,
                 'WAVECAL_TABLE': 'WAVECAL_TABLE-%02i.fits' % ifu,
                 'MASTER_FLAT': 'MASTER_FLAT-%02i.fits' % ifu})

(Here, we opted to save disk space and not save the OBJECT_RED_* .fits images by passing saveimage=False).

Now that the standard star observations have gone through the basic calibration process and are converted into pixel tables, we can use those for the flux calibration. In the following example, we have taken all 24 pixel tables of one standard-star observation:

import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/std.log'

muse_standard = cpl.Recipe('muse_standard')
muse_standard.output_dir = '.

muse_standard.calib.STD_FLUX_TABLE = 'cal/std_flux_table.fits'
muse_standard.calib.EXTINCT_TABLE = 'cal/extinction_paranal.fits'
muse_standard.param.profile = 'circle'

muse_standard(['PIXTABLE_STD_0001-01.fits', 'PIXTABLE_STD_0001-02.fits',
               'PIXTABLE_STD_0001-24.fits'])

This uses the default flux integration with a Moffat profile fit, but one could also choose to use circular flux integration, using profile=circle (see below). (Depending on the dataset, circular integration may cause less wiggles in the output response curve.)

The file with the tag STD_FLUX_TABLE has to contain a reference table for the actual observed standard star. The table that ships with the MUSE pipeline contains usable reference curves for most stars observed with MUSE. In case a new star was observed, selection of the reference table by RA and DEC will fail and the recipe will return an error.

Here is the entire process as a script; in this case we run the standard star recipe twice, once with the default Moffat fit, once with circular integration, to be able to compare the results:
By default, the recipe selects the brightest star in the field to be the standard star. In some cases, where one of the fainter star(s) in the field is the target object, it may be necessary to use \texttt{--select=distance} to select the correct star to compare to the reference. See section 7.2.1 for a full description of the \texttt{muse_standard} recipe.

6.2.2 Astrometry

For reductions of science data one should use a matched pair of geometry table and astrometric solution. It is recommended to use the provided input and skip this section.

Here, we create the astrometric calibration file, which is necessary for a correct relative transformation from pixel positions to world coordinates (RA, DEC).
Once the pixel tables are pre-reduced, they are fed into the `muse_astrometry` recipe. For this, we need an `ASTROMETRY_REFERENCE` table that contains a list of stars in the observed field. A table with the typical reference targets observed with MUSE is shipped with the pipeline. (Optionally, one can input files necessary for flux calibration, but that is usually unnecessary.)

```
import cpl
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/astrometry.log'
muse_astrometry = cpl.Recipe('muse_astrometry')
muse_astrometry.output_dir = '.

muse_astrometry.calib.ASTROMETRY_REFERENCE = 'cal/astrometry_reference.fits'
muse_astrometry([('PIXTABLE_ASTROMETRY_0001-%02i.fits' % ifu)
for ifu in range(1, 25)]

One should now ensure that the computed solution is close to 0.02 for both axes.
See section 7.2.3 for a full description of the `muse_astrometry` recipe.

6.2.3 Sky Creation and Subtraction

This extra step is necessary if we reduce data from an object that fills much of the field of view. Then we need to take an extra exposure of an (empty) sky field and create a `SKY_CONTINUUM` and an initial `SKY_LINES` table using the recipe `muse_create_sky`; otherwise, the sky subtraction is done directly in the `muse_scipost` recipe.

The input pixtable must be either flux calibrated, or the flux calibration has to be specified as calibration files with the tags `STD_RESPONSE`, `EXTINCT_TABLE` and (optionally) `STD_TELLURIC`. 
import cpl
import os
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/create_sky.log'
muse_create_sky = cpl.Recipe('muse_create_sky')
muse_create_sky.output_dir = 'sky'
muse_create_sky.calib.SKY_LINES = 'cal/sky_lines.fits'
muse_create_sky.calib.STD_RESPONSE = 'moffat/STD_RESPONSE-00.fits'
muse_create_sky.calib.EXTINCT_TABLE = 'cal/extinction_paranal.fits'
muse_create_sky.calib.LSF_PROFILE = ['LSF_PROFILE-%02i.fits' % i for i in range(1, 25)]
muse_create_sky(["PIXTABLE_OBJECT-%02i.fits" % i for i in range(1, 25)])

See section 7.2.2 for a full description of the \texttt{muse\_create\_sky} recipe.

### 6.2.4 Science Post-Processing and Final Datacube

When all necessary on-sky calibrations are created (or none is necessary), we can start the post-processing of the science data themselves, i.e. the conversion from the pre-reduced pixel tables to the final datacube. The following example creates a cube for a single full exposure, using flux calibration, model-based sky subtraction, and astrometric calibration, and the maximum pixel fraction with the drizzle algorithm. It creates four images of the field of view, integrated over four filter functions, and saves the images as extensions of the cube FITS file:

import cpl
import os
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/scipost1.log'
os.mkdir('0001')
muse_scipost = cpl.Recipe('muse_scipost')
muse_scipost.output_dir = '0001'
muse_scipost.calib.LSF_PROFILE = ['cal/LSF_PROFILE-%02i.fits' % ifu for ifu in range(1, 25)]
muse_scipost.calib.ASTROMETRY_WCS = 'cal/astrometrywcs.fits'
muse_scipost.calib.SKY_LINES = 'cal/sky_lines.fits'
muse_scipost.calib.EXTINCT_TABLE = 'cal/extinction_paranal.fits'
muse_scipost.calib.STD_RESPONSE = 'moffat/STD_RESPONSE-00.fits'
muse_scipost.calib.STD_TELLURIC = 'moffat/STD_TELLURIC-00.fits'
muse_scipost.calib.FILTER_LIST = 'cal/filters.fits'
muse_scipost.param.pixfrac = 1.
muse_scipost.param.filter = 'white,Johnson_V,Cousins_R,Cousins_I'
muse_scipost.param.format = 'xCube'
If more than one exposure is processed separately in this way, the output files should be put into separate output directories to not be overwritten.

**Note:** this processing step requires a lot of RAM to successfully run it, approximately 18 GB per exposure.

The next example creates a cube for three exposures, with all on-sky calibrations, now using the default pixel fraction for drizzle. The exposures are automatically aligned using their WCS information, and reconstructed into a single cube. To improve the cosmic ray rejection, this now uses the "median" method with a more stringent sigma rejection level. It also saves the three pixel tables of the different exposures after post-processing but before exposure combination. Not all input files are printed:

```python
import cpl
cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/scipost3.log'
muse_scipost = cpl.Recipe('muse_scipost')
muse_scipost.output_dir = '.'
muse_scipost.calib.LSF_PROFILE = [('cal/LSF_PROFILE-%02i.fits') % ifu for ifu in range(1, 25)]
muse_scipost.calib.ASTROMETRY_WCS = 'cal/astrometry_wcs.fits'
muse_scipost.calib.SKY_LINES = 'cal/sky_lines.fits'
muse_scipost.calib.EXTINCT_TABLE = 'cal/extinction_paranal.fits'
muse_scipost.calib.STD_RESPONSE = 'std/STD_RESPONSE.fits'
muse_scipost.calib.STD_TELLURIC = 'std/STD_TELLURIC.fits'
muse_scipost.calib.FILTER_LIST = 'cal/filters.fits'
muse_scipost.param.filter = 'white,Johnson_V,Cousins_R,Cousins_I'
muse_scipost.param.format = 'xCube'
muse_scipost.param.save = 'cube,individual'
muse_scipost.param.crsigma = 6.0

muse_scipost([('PIXTABLE_OBJECT_0001-%02i.fits' % ifu) for ifu in range(1, 25)]

muse_scipost([('PIXTABLE_OBJECT_0004-%02i.fits' % (exp, ifu)) for exp in range(1, 4) for ifu in range(1, 25)]

**Note:** this example needs approximately 55 GB of free RAM to finish!

To optimize the cosmic ray rejection when combining more exposures, one generally needs to reduce the sigma level of the rejection. This may need some iteration, best done over a small wavelength range (using the `--lambdamin` and `--lambdamax` recipe parameters) to check that lower `crsigma` values do not result in rejected object data. Likewise, when using multiple exposures and the default drizzle algorithm, the `pixfrac` parameter could be changed to smaller values.

See section 7.2.4 for a full description of the `muse_scipost` recipe.

### 6.2.5 Combine Exposures

While the combination of different exposure can be done already in `muse_scipost`, it may be easier to let that recipe process each exposure separately and save its post-processed pixel table as intermediate
product. This allows the user to check the individual datacube for proper reduction before starting the last step.

It also facilitates verification or computation of relative exposure offsets (see Sect. 8.6). Briefly, if the exposures are spatially offset (affected by the “wobble”), one can use the `muse_exp_align` recipe to automatically compute offsets using the `IMAGE_FOV` images corresponding to each exposure. The offsets then get saved in the `OFFSET_LIST` which is an optional input to `muse_exp_combine`. The same table can also be used to provide `muse_exp_combine` with relative exposure flux scales (see Sect. 8.7).

The individual pixel tables of each exposure can then be input into `muse_exp_combine` to create the final combined datacube. As with `muse_scipost`, one could try to set the parameters `crsigma` and `pixfrac` to smaller values if more overlapping exposures are part of the dataset.

The following example creates the same cube as in the three-exposure example in 6.2.4, but starts from individual pixel tables:

```python
import cpl

cpl.esorex.init()
cpl.esorex.log.file = 'LOGs/expcomb.log'
muse_exp_combine = cpl.Recipe('muse_exp_combine')
muse_exp_combine.output_dir = '','

muse_exp_combine.calib.FILTER_LIST = 'cal/filters.fits'
muse_exp_combine.param.filter = 'white,Johnson_V,Cousins_R,Cousins_I'
muse_exp_combine.param.format = 'xCube'

muse_exp_combine(['PIXTABLE_REDUCED_e01.fits',
                   'PIXTABLE_REDUCED_e02.fits', 'PIXTABLE_REDUCED_e03.fits'])
```

This needs far fewer inputs, since the reduced individual pixel tables are already fully calibrated. **Note:** as above, this example needs approximately 55 GB of free RAM to finish!
Chapter 7

Recipe Parameters

In following sections, the documentation of the individual pipeline recipes is given, in terms of input data, recipe parameters, output products, and QC parameters created.

7.1 Pre-processing recipes

7.1.1 muse_bias

Combine several separate bias images into one master bias file.

Description

This recipe combines several separate bias images into one master bias file. The master bias contains the combined pixel values, in adu, of the raw bias exposures, with respect to the image combination method used.

Processing trims the raw data and records the overscan statistics, corrects the data levels using the overscan (if overscan is not "none") and combines the exposures using input parameters. The read-out noise is computed for each quadrant of the raw input images and stored as QC parameter. The variance extension is filled with an initial value accordingly, before image combination. Further QC statistics are computed on the output master bias. Additionally, bad columns are searched for and marked in the data quality extension.

Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS</td>
<td>raw</td>
<td>3</td>
<td>Raw bias (more than 3 frames allowed)</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td></td>
<td>Bad pixel table (optional, usually not used)</td>
</tr>
</tbody>
</table>

Recipe parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovssigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.</td>
</tr>
<tr>
<td>ovssreject</td>
<td>string</td>
<td>dcr</td>
<td>This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovssigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than (</td>
</tr>
<tr>
<td>ovssignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip,</td>
<td>Type of image combination to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>average,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>median,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>minmax,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sigclip</td>
<td></td>
</tr>
<tr>
<td>nlow</td>
<td>int</td>
<td>1</td>
<td>Number of minimum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nhigh</td>
<td>int</td>
<td>1</td>
<td>Number of maximum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nkeep</td>
<td>int</td>
<td>1</td>
<td>Number of pixels to keep with minmax.</td>
</tr>
<tr>
<td>lsigma</td>
<td>double</td>
<td>3</td>
<td>Low sigma for pixel rejection with sigclip.</td>
</tr>
<tr>
<td>hsigma</td>
<td>double</td>
<td>3</td>
<td>High sigma for pixel rejection with sigclip.</td>
</tr>
<tr>
<td>losigbadpix</td>
<td>double</td>
<td>30.</td>
<td>Low sigma to find dark columns in the combined bias.</td>
</tr>
<tr>
<td>hisigbadpix</td>
<td>double</td>
<td>3.</td>
<td>High sigma to find bright columns in the combined bias.</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Merge output products from different IFUs into a common file.</td>
</tr>
</tbody>
</table>

**Product frames**

The following product frames are created by the recipe:
Quality control parameters

The following quality control parameters are available for the muse_bias products:

**ESO.QC.BIAS.INPUTn.NSATURATED**  Number of saturated pixels in raw bias i in input list

**ESO.QC.BIAS.MASTERn.MEDIAN**  Median value of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.MEAN**  Mean value of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.STDEV**  Standard deviation value of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.MIN**  Minimum value of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.MAX**  Maximum value of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.RON**  Read-out noise in quadrant n determined from difference images of each adjacent pair of biases in the input dataset in randomly placed windows

**ESO.QC.BIAS.MASTERn.RONERR**  Read-out noise error in quadrant n determined from difference images of each adjacent pair of biases in the input dataset in randomly placed windows

**ESO.QC.BIAS.MASTERn.SLOPE.X**  Average horizontal slope of master bias in quadrant n

**ESO.QC.BIAS.MASTERn.SLOPE.Y**  Average vertical slope of master bias in quadrant n

**ESO.QC.BIAS.MASTER.NBADPIX**  Bad pixels found as part of the bad column search in the master bias

**ESO.QC.BIAS.MASTER.NSATURATED**  Number of saturated pixels in output data

**ESO.QC.BIAS.LEVELn.MEAN**  Average of the raw median values of all input files in quadrant n

**ESO.QC.BIAS.LEVELn.STDEV**  Standard deviation of the raw median values of all input files in quadrant n

**ESO.QC.BIAS.LEVELn.MEDIAN**  Median of the raw median values of all input files in quadrant n

### 7.1.2 muse_dark

Combine several separate dark images into one master dark file and locate hot pixels.

**Description**

This recipe combines several separate dark images into one master dark file. The master dark contains the combined pixel values of the raw dark exposures, with respect to the image combination method used and normalization time specified.

Processing trims the raw data and records the overscan statistics, subtracts the bias (taking account of the overscan, if --overscan is not "none") from each raw input image, converts them from adu to count, scales them according to their exposure time, and combines them using input parameters. Hot pixels are then identified using image statistics and marked in the data quality extension. The combined image is normalized to the specified exposure time. QC statistics are computed on the output master dark.

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARK</td>
<td>raw</td>
<td>3</td>
<td>Raw dark (more than 3 frames allowed)</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
</tbody>
</table>

Continued on next page
### Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovssigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.</td>
</tr>
<tr>
<td>ovssigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than (</td>
</tr>
<tr>
<td>ovssignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip, average, median, minmax, sigclip</td>
<td>Type of image combination to use.</td>
</tr>
<tr>
<td>nlow</td>
<td>int</td>
<td>1</td>
<td>Number of minimum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nhigh</td>
<td>int</td>
<td>1</td>
<td>Number of maximum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nkeep</td>
<td>int</td>
<td>1</td>
<td>Number of pixels to keep with minmax.</td>
</tr>
<tr>
<td>lsigma</td>
<td>double</td>
<td>3</td>
<td>Low sigma for pixel rejection with sigclip.</td>
</tr>
<tr>
<td>hsigma</td>
<td>double</td>
<td>3</td>
<td>High sigma for pixel rejection with sigclip.</td>
</tr>
<tr>
<td>scale</td>
<td>boolean</td>
<td>true</td>
<td>Scale the individual images to a common exposure time before combining them.</td>
</tr>
</tbody>
</table>

Continued on next page
- continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalize</td>
<td>double</td>
<td>3600.</td>
<td>Normalize the master dark to this exposure time (in seconds). To disable normalization, set this to a negative value.</td>
</tr>
<tr>
<td>hotsigma</td>
<td>double</td>
<td>5</td>
<td>Sigma level, in terms of median deviation above the median dark level, above which a pixel is detected and marked as 'hot'.</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Merge output products from different IFUs into a common file.</td>
</tr>
</tbody>
</table>

Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASTER_DARK</td>
<td>Master dark</td>
</tr>
</tbody>
</table>

Quality control parameters

The following quality control parameters are available for the **muse_dark** products:

- `ESO.QC.DARK.INPUTi.NSATURATED` Number of saturated pixels in raw dark i in input list
- `ESO.QC.DARK.MASTER.NBADPIX` Number of bad pixels determined from master dark
- `ESO.QC.DARK.MASTER.MEDIAN` Median value of the master dark
- `ESO.QC.DARK.MASTER.MEAN` Mean value of the master dark
- `ESO.QC.DARK.MASTER.STDEV` Standard deviation of the master dark
- `ESO.QC.DARK.MASTER.MIN` Minimum value of the master dark
- `ESO.QC.DARK.MASTER.MAX` Maximum value of the master dark
- `ESO.QC.DARK.MASTER.DC` Dark current measured on master dark in randomly placed windows
- `ESO.QC.DARK.MASTER.DCERR` Dark current error measured on master dark in randomly placed windows
- `ESO.QC.DARK.MASTER.NSATURATED` Number of saturated pixels in output data

7.1.3 **muse_flat**

Combine several separate flat images into one master flat file, trace slice locations, and locate dark pixels.

Description

This recipe combines several separate flat-field images into one master flat-field file and traces the location of the slices on the CCD. The master flat contains the combined pixel values of the raw flat exposures, with respect to the image combination method used, normalized to the mean flux. The trace table contains polynomials defining the location of the slices on the CCD.

Processing trims the raw data and records the overscan statistics, subtracts the bias (taking account of the overscan, if --overscan is not "none"), and optionally, the dark from each raw input image, converts them from adu to count, scales them according to their exposure time, and combines the exposures using input parameters.
To trace the position of the slices on the CCD, their edges are located using a threshold method. The edge detection is repeated at given intervals thereby tracing the central position (the mean of both edges) and width of each slit vertically across the CCD. Deviant positions of detections on CCD rows can be detected and excluded before fitting a polynomial to all positions measured for one slice. The polynomial parameters for each slice are saved in the output trace table.

Finally, the area between the now known slice edges is searched for dark (and bright) pixels, using statistics in each row of the master flat.

### Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAT</td>
<td>raw</td>
<td>3</td>
<td>Raw flat (more than 3 frames allowed)</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td></td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td></td>
<td>Bad pixel table (optional, usually not used)</td>
</tr>
</tbody>
</table>

### Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovscsigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant. This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovscreject</td>
<td>string</td>
<td>dcr</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than</td>
</tr>
<tr>
<td>ovscignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>ovscignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>ovscignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
</tbody>
</table>

Continued on next page
- continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip, average, median, minmax, sigclip</td>
<td>Type of combination to use</td>
</tr>
<tr>
<td>nlow</td>
<td>int</td>
<td>1</td>
<td>Number of minimum pixels to reject with minmax</td>
</tr>
<tr>
<td>nhigh</td>
<td>int</td>
<td>1</td>
<td>Number of maximum pixels to reject with minmax</td>
</tr>
<tr>
<td>nkeep</td>
<td>int</td>
<td>1</td>
<td>Number of pixels to keep with minmax</td>
</tr>
<tr>
<td>lsigma</td>
<td>double</td>
<td>3</td>
<td>Low sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>hsigma</td>
<td>double</td>
<td>3</td>
<td>High sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>scale</td>
<td>boolean</td>
<td>true</td>
<td>Scale the individual images to a common exposure time before combining them.</td>
</tr>
<tr>
<td>normalize</td>
<td>boolean</td>
<td>true</td>
<td>Normalize the master flat to the average flux</td>
</tr>
<tr>
<td>trace</td>
<td>boolean</td>
<td>true</td>
<td>Trace the position of the slices on the master flat</td>
</tr>
<tr>
<td>nsum</td>
<td>int</td>
<td>32</td>
<td>Number of lines over which to average when tracing</td>
</tr>
<tr>
<td>order</td>
<td>int</td>
<td>5</td>
<td>Order of polynomial fit to the trace</td>
</tr>
<tr>
<td>edgfrac</td>
<td>double</td>
<td>0.5</td>
<td>Fractional change required to identify edge when tracing</td>
</tr>
<tr>
<td>losigmabadpix</td>
<td>double</td>
<td>5</td>
<td>Low sigma to find dark pixels in the master flat</td>
</tr>
<tr>
<td>hisigmabadpix</td>
<td>double</td>
<td>5</td>
<td>High sigma to find bright pixels in the master flat</td>
</tr>
<tr>
<td>samples</td>
<td>boolean</td>
<td>false</td>
<td>Create a table containing all tracing sample points.</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Merge output products from different IFUs into a common file.</td>
</tr>
</tbody>
</table>

### Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASTER_FLAT</td>
<td>Master flat</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>Trace table</td>
</tr>
<tr>
<td>TRACE_SAMPLES</td>
<td>Table containing all tracing sample points, if --samples=true</td>
</tr>
</tbody>
</table>

### Quality control parameters

The following quality control parameters are available for the `muse_flat` products:

- `ESO.QC.FLAT.INPUTi.MEDIAN` Median value of raw flat i in input list
- `ESO.QC.FLAT.INPUTi.MEAN` Mean value of raw flat i in input list
- `ESO.QC.FLAT.INPUTi.STDEV` Standard deviation of raw flat i in input list
7.1.4 muse_wavecal

Detect arc emission lines and determine the wavelength solution for each slice.

Description

This recipe detects arc emission lines and fits a wavelength solution to each slice of the instrument. The wavelength calibration table contains polynomials defining the wavelength solution of the slices on the CCD.

Processing trims the raw data and records the overscan statistics, subtracts the bias (taking account of the overscan, if --overscan is not "none") and converts them from adu to count. Optionally, the dark can be subtracted and the data can be divided by the flat-field, but this is not recommended. The data is then combined using input parameters, first into separate images for each lamp. If --lampwise is not given
or if --resample is given, these lamp-separate exposures are summed to create a single combined master arc.

To compute the wavelength solution, arc lines are detected at the center of each slice (using threshold detection on a S/N image) and subsequently assigned wavelengths, using pattern matching to identify lines from the input line catalog. Each line is then traced to the edges of the slice, using Gaussian centering in each CCD column. The Gaussians not only yield center, but also centering error, and line properties (e.g. FWHM). Deviant fits are detected using polynomial fits to each arc line (using the xorder parameter) and rejected. If --lampwise is switched on, these analysis and measuring steps are carried out separately on images exposed by the different arc lamps, reducing the amount of blending, that can otherwise influence line identification and Gaussian centering. The final two-dimensional fit uses all positions (of all lamps), their wavelengths, and the given polynomial orders to compute the final wavelength solution for each slice, iteratively rejecting outliers. This final fit can be either unweighted (fitweighting="uniform", for fastest processing) or weighted (other values of fitweighting, for higher accuracy).

Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>raw</td>
<td>1</td>
<td>Raw arc lamp exposures</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td>1</td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>MASTER_FLAT</td>
<td>calib</td>
<td>1</td>
<td>Master flat (optional, usually not used)</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Trace table</td>
</tr>
<tr>
<td>LINE_CATALOG</td>
<td>calib</td>
<td>1</td>
<td>Arc line catalog</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Bad pixel table (optional, usually not used)</td>
</tr>
</tbody>
</table>

Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovssigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.</td>
</tr>
<tr>
<td>ovssreject</td>
<td>string</td>
<td>dcr</td>
<td>This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovssigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sigma rejection level for the iterative polynomial fit (the level comparison is then done afterwards</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with (100 \times \text{stdev}) to guard against incompatible settings). Has no effect for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>\text{overscan}=&quot;offset&quot;.</td>
</tr>
<tr>
<td>ovssigma</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>computing statistics or fits.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip, average,</td>
<td>Type of lampwise image combination to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>median, minmax, sigclip</td>
<td></td>
</tr>
<tr>
<td>lampwise</td>
<td>boolean</td>
<td>true</td>
<td>Identify and measure the arc emission lines on images separately for each lamp setup.</td>
</tr>
<tr>
<td>sigma</td>
<td>double</td>
<td>1.0</td>
<td>Sigma level used to detect arc emission lines above the median background level in the S/N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>image of the central column of each slice</td>
</tr>
<tr>
<td>dres</td>
<td>double</td>
<td>0.05</td>
<td>The allowed range of resolutions for pattern matching (of detected arc lines to line list) in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fractions relative to the expected value</td>
</tr>
<tr>
<td>tolerance</td>
<td>double</td>
<td>0.1</td>
<td>Tolerance for pattern matching (of detected arc lines to line list)</td>
</tr>
<tr>
<td>xorder</td>
<td>int</td>
<td>2</td>
<td>Order of the polynomial for the horizontal curvature within each slice</td>
</tr>
<tr>
<td>yorder</td>
<td>int</td>
<td>6</td>
<td>Order of the polynomial used to fit the dispersion relation</td>
</tr>
<tr>
<td>linesigma</td>
<td>double</td>
<td>-1.0</td>
<td>Sigma level for iterative rejection of deviant fits for each arc line within each slice, a negative</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value means to use the default (2.5).</td>
</tr>
<tr>
<td>residuals</td>
<td>boolean</td>
<td>false</td>
<td>Create a table containing residuals of the fits to the data of all arc lines. This is useful to assess</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the quality of the wavelength solution in detail.</td>
</tr>
<tr>
<td>fitsigma</td>
<td>double</td>
<td>-1.0</td>
<td>Sigma level for iterative rejection of deviant datapoints during the final polynomial wavelength</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>solution within each slice, a negative value means to use the default (3.0).</td>
</tr>
<tr>
<td>fitweighting</td>
<td>string</td>
<td>cersscatter, uniform, cerr, scatter, cersscatter</td>
<td>Type of weighting to use in the final polynomial wavelength solution fit, using centroiding error estimate and/or scatter of each single line as estimates of its accuracy.</td>
</tr>
</tbody>
</table>
- continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resample</td>
<td>boolean</td>
<td>false</td>
<td>Resample the input arc images onto 2D images for a visual check using tracing and wavelength calibration solutions. Note that the image produced will show small wiggles even when the calibration was successful!</td>
</tr>
<tr>
<td>wavemap</td>
<td>boolean</td>
<td>false</td>
<td>Create a wavelength map of the input images</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Merge output products from different IFUs into a common file.</td>
</tr>
</tbody>
</table>

### Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAVECAL_TABLE</td>
<td>Wavelength calibration table</td>
</tr>
<tr>
<td>WAVECAL_RESIDUALS</td>
<td>Fit residuals of all arc lines (if --residuals=true)</td>
</tr>
<tr>
<td>ARC_RED_LAMP</td>
<td>Reduced ARC image, per lamp</td>
</tr>
<tr>
<td>ARC_RED</td>
<td>Reduced, combined master ARC image (if --lampwise=false or --resample=true)</td>
</tr>
<tr>
<td>ARC_RESAMPLED</td>
<td>Resampled arc images (if --resample=true)</td>
</tr>
<tr>
<td>WAVE_MAP</td>
<td>Wavelength map of the input images</td>
</tr>
</tbody>
</table>

### Quality control parameters

The following quality control parameters are available for the `muse_wavecal` products:

- `ESO.QC.WAVECAL.SLICEj.LINES.NDET` Number of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.NID` Number of identified arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.PEAK.MEAN` Mean peak count level above background of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.PEAK.STDEV` Standard deviation of peak count level above background of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.PEAK.MIN` Peak count level above background of the faintest line in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.PEAK.MAX` Peak count level above background of the brightest line in slice j
- `ESO.QC.WAVECAL.SLICEj.LAMPl.LINES.PEAK.MEAN` Mean peak count level of lines of lamp l above background of detected arc lines in slice j. Not produced with --lampwise=FALSE!
- `ESO.QC.WAVECAL.SLICEj.LAMPl.LINES.PEAK.STDEV` Standard deviation of peak count level of lines of lamp l above background of detected arc lines in slice j. Not produced with --lampwise=FALSE!
- `ESO.QC.WAVECAL.SLICEj.LAMPl.LINES.PEAK.MAX` Peak count level above background of the brightest line of lamp l in slice j. Not produced with --lampwise=FALSE!
- `ESO.QC.WAVECAL.SLICEj.LINES.FWHM.MEAN` Mean FWHM of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.FWHM.STDEV` Standard deviation of FWHM of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.FWHM.MIN` Minimum FWHM of detected arc lines in slice j
- `ESO.QC.WAVECAL.SLICEj.LINES.FWHM.MAX` Maximum FWHM of detected arc lines in slice j
Compute the LSF

**Description**

Compute the slice and wavelength dependent LSF from a lines spectrum (ARC lamp).

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARC</td>
<td>raw</td>
<td>1</td>
<td>Raw arc lamp exposures</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td></td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>MASTER_FLAT</td>
<td>calib</td>
<td></td>
<td>Master flat (optional)</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Trace table</td>
</tr>
<tr>
<td>WAVECAL_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Wavelength calibration table</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td></td>
<td>Known bad pixels (optional, usually not used)</td>
</tr>
<tr>
<td>LINE_CATALOG</td>
<td>calib</td>
<td>1</td>
<td>Arc line catalog</td>
</tr>
</tbody>
</table>

**Recipe parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovssigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.</td>
</tr>
<tr>
<td>ovscrreject</td>
<td>string</td>
<td>dcr</td>
<td>This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovssigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than</td>
</tr>
<tr>
<td>ovssignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>save_subtracted</td>
<td>boolean</td>
<td>false</td>
<td>Save the pixel table after the LSF subtraction.</td>
</tr>
<tr>
<td>line_quality</td>
<td>int</td>
<td>3</td>
<td>Minimal quality flag in line catalog for selection</td>
</tr>
<tr>
<td>lsf_range</td>
<td>double</td>
<td>7.5</td>
<td>Wavelength window (half size) around each line to estimate LSF</td>
</tr>
<tr>
<td>lsf_size</td>
<td>int</td>
<td>150</td>
<td>Image size in LSF direction</td>
</tr>
<tr>
<td>lambda_size</td>
<td>int</td>
<td>30</td>
<td>Image size in line wavelength direction</td>
</tr>
<tr>
<td>lsf_regression_window</td>
<td>double</td>
<td>0.7</td>
<td>Size of the regression window in LSF direction Merge output products from different IFUs into a common file.</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Type of lampwise image combination to use.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip, average, median, minmax, sigclip</td>
<td>LSF generation method. Depending on this value, either an interpolated LSF cube is created, or a table with the parameters of a hermitean gaussian.</td>
</tr>
<tr>
<td>method</td>
<td>string</td>
<td>interpolate, hermit</td>
<td></td>
</tr>
</tbody>
</table>

**Product frames**

The following product frames are created by the recipe:

62
<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSF_PROFILE</td>
<td>Slice specific LSF images, stacked into one data cube per IFU.</td>
</tr>
<tr>
<td>PIXTABLE_SUBTRACTED</td>
<td>Subtracted combined pixel table, if --save_subtracted=true. This file contains only the subtracted arc lines that contributed to the LSF calculation. There are additional columns line_lamda and line_flux with the reference wavelength and the estimated line flux of the corresponding arc line.</td>
</tr>
</tbody>
</table>

### Quality control parameters

The following quality control parameters are available for the `muse_lsf` products:

- `ESO.QC.LSF.IFUm.SLICEj.FWHM.MEAN` Mean FWHM of the LSF slice j
- `ESO.QC.LSF.IFUm.SLICEj.FWHM.STDEV` Standard deviation of the LSF in slice j
- `ESO.QC.LSF.IFUm.SLICEj.FWHM.MIN` Minimum FWHM of the LSF in slice j
- `ESO.QC.LSF.IFUm.SLICEj.FWHM.MAX` Maximum FWHM of the LSF in slice j

#### 7.1.6 muse_twilight

Combine several twilight skyflats into one cube, compute correction factors for each IFU, and create a smooth 3D illumination correction.

### Description

Processing first handles each raw input image separately: it trims the raw data and records the overscan statistics, subtracts the bias (taking account of the overscan, if --overscan is not "none"), converts the images from adu to count, subtracts the dark, divides by the flat-field and combines all the exposures using input parameters.

The input calibrations geometry table, trace table, and wavelength calibration table are used to assign 3D coordinates to each CCD-based pixel, thereby creating a pixel table from the master sky-flat. These pixel tables are then cut in wavelength using the --lambdamin and --lambdamax parameters. The integrated flux in each IFU is computed as the sum of the data in the pixel table, and saved in the header, to be used later as estimate for the relative throughput of each IFU.

If an ILLUM exposure was given as input, it is then used to correct the relative illumination between all slices of one IFU. For this, the data of each slice within the pixel table of each IFU is multiplied by the normalized median flux of that slice in the ILLUM exposure.

The pixel tables of all IFUs are then merged, using the integrated fluxes as inverse scaling factors, and a cube is reconstructed from the merged dataset, using given parameters. A white-light image is created from the cube. This skyflat cube is then saved to disk, with the white-light image as one extension.

To construct a smooth 3D illumination correction, the cube is post-processed in the following way: the white-light image is used to create a mask of the illuminated area. From this area, the optional vignetting mask is removed. The smoothing is then computed for each plane of the cube: the illuminated area is smoothed (by a 5x7 median filter), normalized, fit with a 2D polynomial (of given polynomial orders), and normalized again. A smooth white image is then created by collapsing the smooth cube.
If a vignetting mask was given, the corner area given by the mask is used to compute a 2D correction for the vignetted area: the original unsmoothed white-light image is corrected for large scale gradients by dividing it with the smooth white image. The residuals in the corner area then smoothed using input parameters. This smoothed vignetting correction is the multiplied onto each plane of the smooth cube, normalizing each plane again.

This twilight cube is then saved to disk.

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKYFLAT</td>
<td>raw</td>
<td>3</td>
<td>Raw twilight skyflat (more than 3 frames allowed)</td>
</tr>
<tr>
<td>ILLUM</td>
<td>raw</td>
<td></td>
<td>Single optional raw (attached/illumination) flat-field exposure (optional)</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td></td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>MASTER_FLAT</td>
<td>calib</td>
<td>1</td>
<td>Master flat</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Known bad pixels (optional, usually not used)</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Tracing table for all slices</td>
</tr>
<tr>
<td>WAVECAL_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Wavelength calibration table</td>
</tr>
<tr>
<td>GEOMETRY_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Relative positions of the slices in the field of view</td>
</tr>
<tr>
<td>VIGNETTING_MASK</td>
<td>calib</td>
<td></td>
<td>Mask to mark vignetted regions in the MUSE field of view (optional)</td>
</tr>
</tbody>
</table>

**Recipe parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovssigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant. This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovssreject</td>
<td>string</td>
<td>dcr</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovssigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than</td>
</tr>
<tr>
<td>ovssignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>sigclip, average,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>median, minmax, sigclip</td>
<td></td>
</tr>
<tr>
<td>nlow</td>
<td>int</td>
<td>1</td>
<td>Number of minimum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nhhigh</td>
<td>int</td>
<td>1</td>
<td>Number of maximum pixels to reject with minmax.</td>
</tr>
<tr>
<td>nkeep</td>
<td>int</td>
<td>1</td>
<td>Number of pixels to keep with minmax</td>
</tr>
<tr>
<td>lsigma</td>
<td>double</td>
<td>3</td>
<td>Low sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>hsigma</td>
<td>double</td>
<td>3</td>
<td>High sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>scale</td>
<td>boolean</td>
<td>false</td>
<td>Scale the individual images to a common exposure time before combining them.</td>
</tr>
<tr>
<td>resample</td>
<td>string</td>
<td>drizzle, nearest,</td>
<td>The resampling technique to use for the final output cube.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>linear, quadratic, renka, drizzle, lanczos</td>
<td></td>
</tr>
<tr>
<td>crtype</td>
<td>string</td>
<td>median, iraf, mean, median</td>
<td>Type of statistics used for detection of cosmic rays during final resampling. &quot;iraf&quot; uses the variance information, &quot;mean&quot; uses standard (mean/stdev) statistics, &quot;median&quot; uses median and the median median of the absolute median deviation.</td>
</tr>
<tr>
<td>crsigma</td>
<td>double</td>
<td>50.</td>
<td>Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.</td>
</tr>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>5000.</td>
<td>Minimum wavelength for twilight reconstruction.</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>9000.</td>
<td>Maximum wavelength for twilight reconstruction.</td>
</tr>
<tr>
<td>dlambda</td>
<td>double</td>
<td>250.</td>
<td>Sampling for twilight reconstruction, this should result in planes of equal wavelength coverage.</td>
</tr>
</tbody>
</table>

Continued on next page
### Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_CUBE_SKYFLAT</td>
<td>Cube of combined twilight skyflat exposures</td>
</tr>
<tr>
<td>TWILIGHT_CUBE</td>
<td>Smoothed cube of twilight sky</td>
</tr>
</tbody>
</table>

### Quality control parameters

The following quality control parameters are available for the `muse_twilight` products:

- `ESO.QC.TWILIGHTm.INPUTi.MEDIAN` Median value of raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.INPUTi.MEAN` Mean value of raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.INPUTi.STDEV` Standard deviation of raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.INPUTi.MIN` Minimum value of raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.INPUTi.MAX` Maximum value of raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.INPUTi.NSATURATED` Number of saturated pixels in raw exposure i of IFU m in input list
- `ESO.QC.TWILIGHTm.MASTER.MEDIAN` Median value of the combined exposures in IFU m
- `ESO.QC.TWILIGHTm.MASTER.MEAN` Mean value of the combined exposures in IFU m
- `ESO.QC.TWILIGHTm.MASTER.STDEV` Standard deviation of the combined exposures in IFU m
- `ESO.QC.TWILIGHTm.MASTER.MIN` Minimum value of the combined exposures in IFU m
- `ESO.QC.TWILIGHTm.MASTER.MAX` Maximum value of the combined exposures in IFU m
Flux integrated over the whole CCD of the combined exposures of IFU m.

Flux integrated over all slices of IFU m. Computed using the pixel table of the exposure.

**7.1.7 muse_geometry**

Compute relative location of the slices within the field of view and measure the instrumental PSF on the detectors.

**Description**

Processing first works separately on each IFU of the raw input data (in parallel): it trims the raw data and records the overscan statistics, subtracts the bias and converts them from adu to count. Optionally, the dark can be subtracted and the data can be divided by the flat-field. The data of all input mask exposures is then averaged. The averaged image together with the trace mask and wavelength calibration as well as the line catalog are used to detect spots. The detection windows are used to measure the spots on all images of the sequence, the result is saved, with information on the measured PSF, in the spots tables. Then properties of all slices are computed, first separately on each IFU to determine the peak position of the mask for each slice and its angle, subsequently the width and horizontal position. Then, the result of all IFUs is analyzed together to produce a refined horizontal position, applying global shifts to each IFU as needed. The vertical position is then determined using the known slice ordering on the sky; the relative peak positions are put into sequence, taking into account the vertical offsets of the pinholes in the mask. Finally, the geometry table is cleaned up from intermediate debug data and saved. As a last optional step, additional raw input data is reduced using the newly geometry to produce an image of the field of view. If these exposures contain smooth features, they can be used as a visual check of the quality of the geometrical calibration.

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASK</td>
<td>raw</td>
<td>50</td>
<td>The full exposure sequence of raw multi-pinhole mask images (more than 50 frames allowed)</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td>1</td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>MASTER_FLAT</td>
<td>calib</td>
<td>1</td>
<td>Master flat (optional)</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Trace table</td>
</tr>
<tr>
<td>WAVECAL_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Wavelength calibration table</td>
</tr>
<tr>
<td>LINE_CATALOG</td>
<td>calib</td>
<td>1</td>
<td>List of arc lines</td>
</tr>
<tr>
<td>BADPIX_TABLE</td>
<td>calib</td>
<td></td>
<td>Known bad pixels (optional, usually not used)</td>
</tr>
<tr>
<td>MASK_CHECK</td>
<td>calib</td>
<td></td>
<td>Some other optional raw frame, ideally a trace mask exposure or something else with smooth features. (optional, more than one frame allowed)</td>
</tr>
</tbody>
</table>

**Recipe parameters**
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifu1</td>
<td>int</td>
<td>1</td>
<td>First IFU to analyze.</td>
</tr>
<tr>
<td>ifu2</td>
<td>int</td>
<td>24</td>
<td>Last IFU to analyze.</td>
</tr>
<tr>
<td>sigma</td>
<td>double</td>
<td>2.2</td>
<td>Sigma detection level for spot detection, in terms of median deviation above the median.</td>
</tr>
<tr>
<td>centroid</td>
<td>string</td>
<td>gaussian, barycenter, gaussian</td>
<td>Type of centroiding and FWHM determination to use for all spot measurements: simple barycenter method or using a Gaussian fit.</td>
</tr>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>6800.</td>
<td>When passing any MASK_CHECK frames in the input, use this lower wavelength cut before reconstructing the image.</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>7200.</td>
<td>When passing any MASK_CHECK frames in the input, use this upper wavelength cut before reconstructing the image.</td>
</tr>
</tbody>
</table>

### Product Frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASK_REDUCED</td>
<td>Reduced pinhole mask images</td>
</tr>
<tr>
<td>MASK_COMBINED</td>
<td>Combined pinhole mask image</td>
</tr>
<tr>
<td>SPOTS_TABLE</td>
<td>Measurements of all detected spots on all input images.</td>
</tr>
<tr>
<td>GEOMETRY_TABLE</td>
<td>Relative positions of the slices in the field of view.</td>
</tr>
<tr>
<td>GEOMETRY_CUBE</td>
<td>Cube of the field of view to check the geometry calibration. It is restricted to the wavelength range given in the parameters and contains an integrated image (&quot;white&quot;) over this range.</td>
</tr>
<tr>
<td>GEOMETRY_CHECK</td>
<td>Optional field of view image to check the geometry calibration, integrated over the wavelength range given in the parameters.</td>
</tr>
</tbody>
</table>

### Quality Control Parameters

The following quality control parameters are available for the `muse_geometry` products:

- `ESO.QC.GEO.EXP1.FWHM.MEAN` Average FWHM of all bright spots in exposure k.
- `ESO.QC.GEO.EXP1.FWHM.MEDIAN` Median FWHM of all bright spots in exposure k.
- `ESO.QC.GEO.EXP1.FWHM.STDEV` Standard deviation of FWHM of all bright spots in exposure k.
- `ESO.QC.GEO.IFUm.ANGLE` Angle of the mask with respect to the slicer system, computed as median angle of all slices of this IFU for which the measurement could be made.
- `ESO.QC.GEO.IFUm.WLEN1.FLUX.MEAN` Average integrated flux in all spots at reference wavelength l.
- `ESO.QC.GEO.IFUm.WLEN1.FLUX.MEDIAN` Median integrated flux in all spots at reference wavelength l.
- `ESO.QC.GEO.IFUm.WLEN1.FLUX.STDEV` Standard deviation of integrated flux in all spots at reference wavelength l.
- `ESO.QC.GEO.MASK.ANGLE` Angle of the mask with respect to the slicer system, computed as median angle of all slices of all IFUs for which the measurement could be made.
7.1.8 muse_scibasic

Remove the instrumental signature from the data of each CCD and convert them from an image into a pixel table.

Description

Processing handles each raw input image separately: it trims the raw data and records the overscan statistics, subtracts the bias (taking account of the overscan, if --overscan is not "none"), optionally detects cosmic rays (note that by default cosmic ray rejection is handled in the post processing recipes while the data is reformatted into a datacube, so that the default setting is cr="none" here), converts the images from adu to count, subtracts the dark, divides by the flat-field, and (optionally) propagates the integrated flux value from the twilight-sky cube.

[For special cases, users can choose to combine all input files at the image level, so that the pixel table is only created once, for the combined data. This is not recommended for science data, where the combination should take place after correcting for atmospheric effects, before the creation of the final cube.]

The reduced image is then saved (if --saveimage=true).

The input calibrations geometry table, trace table, and wavelength calibration table are used to assign 3D coordinates to each CCD-based pixel, thereby creating a pixel table for each exposure.

If --skylines contains one or more wavelengths for (bright and isolated) sky emission lines, these lines are used to correct the wavelength calibration using an offset.

The data is then cut to a useful wavelength range (if --crop=true).

If an ILLUM exposure was given as input, it is then used to correct the relative illumination between all slices of one IFU. For this, the data of each slice is multiplied by the normalized median flux of that slice in the ILLUM exposure.

As last step, the data is divided by the normalized twilight cube (if given), using the 3D coordinate of each pixel in the pixel table to interpolate the twilight correction onto the data.

This pre-reduced pixel table for each exposure is then saved to disk.

Input frames

At least one raw frame of the category OBJECT, STD, SKY, or ASTROMETRY is required. More raw frames are allowed, but they all must have the same category.

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT</td>
<td>raw</td>
<td></td>
<td>Raw exposure of a science target</td>
</tr>
<tr>
<td>STD</td>
<td>raw</td>
<td></td>
<td>Raw exposure of a standard star field</td>
</tr>
<tr>
<td>SKY</td>
<td>raw</td>
<td></td>
<td>Raw exposure of an (almost) empty sky field</td>
</tr>
<tr>
<td>ASTROMETRY</td>
<td>raw</td>
<td></td>
<td>Raw exposure of an astrometric field</td>
</tr>
<tr>
<td>ILLUM</td>
<td>raw</td>
<td></td>
<td>Single optional raw (attached/illumination) flat-field exposure (optional)</td>
</tr>
<tr>
<td>MASTER_BIAS</td>
<td>calib</td>
<td>1</td>
<td>Master bias</td>
</tr>
<tr>
<td>MASTER_DARK</td>
<td>calib</td>
<td></td>
<td>Master dark (optional, usually not used)</td>
</tr>
<tr>
<td>MASTER_FLAT</td>
<td>calib</td>
<td>1</td>
<td>Master flat</td>
</tr>
<tr>
<td>TRACE_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Trace table</td>
</tr>
</tbody>
</table>

Continued on next page
### Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nifu</td>
<td>int</td>
<td>0, default, other</td>
<td>IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.</td>
</tr>
<tr>
<td>overscan</td>
<td>string</td>
<td>vpoly</td>
<td>If this is &quot;none&quot;, stop when detecting discrepant overscan levels (see ovscsigma), for &quot;offset&quot; it assumes that the mean overscan level represents the real offset in the bias levels of the exposures involved, and adjusts the data accordingly; for &quot;vpoly&quot;, a polynomial is fit to the vertical overscan and subtracted from the whole quadrant. This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovscreject</td>
<td>string</td>
<td>dcr</td>
<td>This influences how values are rejected when computing overscan statistics. Either no rejection at all (&quot;none&quot;), rejection using the DCR algorithm (&quot;dcr&quot;), or rejection using an iterative constant fit (&quot;fit&quot;).</td>
</tr>
<tr>
<td>ovscsigma</td>
<td>double</td>
<td>30.</td>
<td>If the deviation of mean overscan levels between a raw input image and the reference image is higher than</td>
</tr>
<tr>
<td>ovscignore</td>
<td>int</td>
<td>3</td>
<td>The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.</td>
</tr>
<tr>
<td>crop</td>
<td>boolean</td>
<td>true</td>
<td>Automatically crop the output pixel tables in wavelength depending on the expected useful wavelength range of the observation mode used (4750-9350 Angstrom for nominal mode, 4600-9350 Angstrom for extended mode).</td>
</tr>
<tr>
<td>cr</td>
<td>string</td>
<td>none, none, dcr</td>
<td>Type of cosmic ray cleaning to use (for quick-look data processing).</td>
</tr>
<tr>
<td>xbox</td>
<td>int</td>
<td>15</td>
<td>Search box size in x. Only used if cr=dcr.</td>
</tr>
<tr>
<td>ybox</td>
<td>int</td>
<td>40</td>
<td>Search box size in y. Only used if cr=dcr.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Values</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>passes</td>
<td>int</td>
<td>2</td>
<td>Maximum number of cleaning passes. Only used if cr=dcr.</td>
</tr>
<tr>
<td>thres</td>
<td>double</td>
<td>5.8</td>
<td>Threshold for detection gap in factors of standard deviation. Only used if cr=dcr.</td>
</tr>
<tr>
<td>combine</td>
<td>string</td>
<td>none, none, average,</td>
<td>Type of combination to use. Note that in most cases, science exposures cannot easily be combined on the CCD level, so this should usually be kept as &quot;none&quot;! This does not pay attention about the type of input data, and will combine all raw inputs!</td>
</tr>
<tr>
<td>nlow</td>
<td>int</td>
<td>1</td>
<td>Number of minimum pixels to reject with minmax</td>
</tr>
<tr>
<td>nhigh</td>
<td>int</td>
<td>1</td>
<td>Number of maximum pixels to reject with minmax</td>
</tr>
<tr>
<td>nkeep</td>
<td>int</td>
<td>1</td>
<td>Number of pixels to keep with minmax</td>
</tr>
<tr>
<td>lsigma</td>
<td>double</td>
<td>3</td>
<td>Low sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>hsigma</td>
<td>double</td>
<td>3</td>
<td>High sigma for pixel rejection with sigclip</td>
</tr>
<tr>
<td>scale</td>
<td>boolean</td>
<td>true</td>
<td>Scale the individual images to a common exposure time before combining them.</td>
</tr>
<tr>
<td>saveimage</td>
<td>boolean</td>
<td>true</td>
<td>Save the pre-processed CCD-based image of each input exposure before it is transformed into a pixel table.</td>
</tr>
<tr>
<td>skylines</td>
<td>string</td>
<td>5577.339,6300.304</td>
<td>List of wavelengths of sky emission lines (in Angstrom) to use as reference for wavelength offset correction using a Gaussian fit. It can contain multiple (isolated) lines, as comma-separated list, individual shifts are then combined into one weighted average offset. Set to &quot;none&quot; to deactivate.</td>
</tr>
<tr>
<td>skyhalfwidth</td>
<td>double</td>
<td>5.</td>
<td>Half-width of the extraction box (in Angstrom) around each sky emission line.</td>
</tr>
<tr>
<td>skybinsize</td>
<td>double</td>
<td>0.1</td>
<td>Size of the bins (in Angstrom per pixel) for the intermediate spectrum to do the Gaussian fit to each sky emission line.</td>
</tr>
<tr>
<td>resample</td>
<td>boolean</td>
<td>false</td>
<td>Resample the input science data into 2D spectral images using tracing and wavelength calibration solutions for a visual check. Note that the image produced will show small wiggles even when the input calibrations are good and were applied successfully!</td>
</tr>
<tr>
<td>dlambda</td>
<td>double</td>
<td>1.25</td>
<td>Wavelength step (in Angstrom per pixel) to use for resampling.</td>
</tr>
<tr>
<td>merge</td>
<td>boolean</td>
<td>false</td>
<td>Merge output products from different IFUs into a common file.</td>
</tr>
</tbody>
</table>
Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_RED</td>
<td>Pre-processed CCD-based images for OBJECT input (if --saveimage=true)</td>
</tr>
<tr>
<td>OBJECT_RESAMPLED</td>
<td>Resampled 2D image for OBJECT input (if --resample=true)</td>
</tr>
<tr>
<td>PIXTABLE_OBJECT</td>
<td>Output pixel table for OBJECT input</td>
</tr>
<tr>
<td>STD_RED</td>
<td>Pre-processed CCD-based images for STD input (if --saveimage=true)</td>
</tr>
<tr>
<td>STD_RESAMPLED</td>
<td>Resampled 2D image for STD input (if --resample=true)</td>
</tr>
<tr>
<td>PIXTABLE_STD</td>
<td>Output pixel table for STD input</td>
</tr>
<tr>
<td>SKY_RED</td>
<td>Pre-processed CCD-based images for SKY input (if --saveimage=true)</td>
</tr>
<tr>
<td>SKY_RESAMPLED</td>
<td>Resampled 2D image for SKY input (if --resample=true)</td>
</tr>
<tr>
<td>PIXTABLE_SKY</td>
<td>Output pixel table for SKY input</td>
</tr>
<tr>
<td>ASTROMETRY_RED</td>
<td>Pre-processed CCD-based images for ASTROMETRY input (if --saveimage=true)</td>
</tr>
<tr>
<td>ASTROMETRY_RESAMPLED</td>
<td>Resampled 2D image for ASTROMETRY input (if --resample=true)</td>
</tr>
<tr>
<td>PIXTABLE_ASTROMETRY</td>
<td>Output pixel table for ASTROMETRY input</td>
</tr>
</tbody>
</table>

Quality control parameters

The following quality control parameters are available for the muse_scibasic products:

- ESO.QC.SCIBASIC.NSATURATED  Number of saturated pixels in output data
- ESO.QC.SCIBASIC.LAMBDA.SHIFT Shift in wavelength applied to the data using sky emission line(s)

7.2 Post-processing recipes

7.2.1 muse_standard

Create a flux response curve from a standard star exposure.

Description

Merge pixel tables from all IFUs and correct for differential atmospheric refraction.

To derive the flux response curve, integrate the flux of all objects detected within the field of view using the given profile. Select one object as the standard star (either the brightest or the one nearest one, depending on --select) and compare its measured fluxes to tabulated fluxes to derive the sensitivity over wavelength. Postprocess this sensitivity curve to mark wavelength ranges affected by telluric absorption. Interpolate over the telluric regions and derive a telluric correction spectrum for them. The final response curve is then linearly extrapolated to the largest possible MUSE wavelength range and smoothed (with
the method given by --smooth). The derivation of the telluric correction spectrum assumes that the star has a smooth spectrum within the telluric regions.

If there are more than one exposure given in the input data, the derivation of the flux response and telluric corrections are done separately for each exposure. For each exposure, the datacube used for flux integration is saved, together with collapsed images for each given filter.

## Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXTABLE_STD</td>
<td>raw</td>
<td>1</td>
<td>Pixel table of a standard star field</td>
</tr>
<tr>
<td>EXTINCT_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Atmospheric extinction table</td>
</tr>
<tr>
<td>STD_FLUX_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Flux reference table for standard stars (more than one frame allowed)</td>
</tr>
<tr>
<td>TELLURIC_REGIONS</td>
<td>calib</td>
<td></td>
<td>Definition of telluric regions (optional)</td>
</tr>
<tr>
<td>FILTER_LIST</td>
<td>calib</td>
<td></td>
<td>File to be used to create field-of-view images. (optional)</td>
</tr>
</tbody>
</table>

## Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profile</td>
<td>string</td>
<td>moffat, gaussian,</td>
<td>Type of flux integration to use. &quot;gaussian&quot; and &quot;moffat&quot; use 2D profile fitting, circle and square are non-optimal flux integrators.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>moffat, gaussian,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>moffat, circle,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>square</td>
<td></td>
</tr>
<tr>
<td>select</td>
<td>string</td>
<td>flux, flux, distance</td>
<td>How to select the star for flux integration, &quot;flux&quot; uses the brightest star in the field, &quot;distance&quot; uses the detection nearest to the approximate coordinates of the reference source.</td>
</tr>
<tr>
<td>smooth</td>
<td>string</td>
<td>ppoly, none, median, ppoly</td>
<td>How to smooth the response curve before writing it to disk. &quot;none&quot; does not do any kind of smoothing (such a response curve is only useful, if smoothed externally; &quot;median&quot; does a median-filter of 15 Angstrom half-width; &quot;ppoly&quot; fits piecewise cubic polynomials (each one across 2x150 Angstrom width) postprocessed by a sliding average filter of 15 Angstrom half-width.</td>
</tr>
<tr>
<td>lambda_min</td>
<td>double</td>
<td>4000.</td>
<td>Cut off the data below this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambda_max</td>
<td>double</td>
<td>10000.</td>
<td>Cut off the data above this wavelength after loading the pixel table(s).</td>
</tr>
</tbody>
</table>

Continued on next page
Parameter | Type | Values | Description
--- | --- | --- | ---
lambdaref | double | 7000. | Reference wavelength used for correction of differential atmospheric refraction. The R-band (peak wavelength 7000 Angstrom) that is usually used for guiding, is close to the central wavelength of MUSE, so a value of 7000.0 Angstrom should be used if nothing else is known. A value less than zero switches DAR correction off.
darccheck | string | none, none, check, correct | Carry out a check of the theoretical DAR correction using source centroiding. If "correct" it will also apply an empirical correction.
filter | string | white | The filter name(s) to be used for the output field-of-view image. Each name has to correspond to an EXTNAME in an extension of the FILTER_LIST file. If an unsupported filter name is given, creation of the respective image is omitted. If multiple filter names are given, they have to be comma separated.

Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_CUBE_STD</td>
<td>Reduced standard star field exposure</td>
</tr>
<tr>
<td>STD_FLUXES</td>
<td>The integrated flux per wavelength of all detected sources</td>
</tr>
<tr>
<td>STD_RESPONSE</td>
<td>Response curve as derived from standard star(s)</td>
</tr>
<tr>
<td>STD_TELLURIC</td>
<td>Telluric absorption as derived from standard star(s)</td>
</tr>
</tbody>
</table>

Quality control parameters

The following quality control parameters are available for the muse_standard products:

- **ESO.QC.STANDARD.NDET**: Number of detected sources in output cube.
- **ESO.QC.STANDARD.LAMBDA**: Wavelength of plane in combined cube that was used for object detection.
- **ESO.QC.STANDARD.POSk.X**: Position of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.
- **ESO.QC.STANDARD.POSk.Y**: Position of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.
- **ESO.QC.STANDARD.FWHMk.X**: FWHM of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.
- **ESO.QC.STANDARD.FWHMk.Y**: FWHM of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.
- **ESO.QC.STANDARD.FWHM.VALID**: Number of detected sources with valid FWHM in output cube.
- **ESO.QC.STANDARD.FWHM.MEDIAN**: Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected,
Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

### 7.2.2 muse_create_sky

Create night sky model from selected pixels of an exposure of empty sky.

**Description**

This recipe creates the continuum and the atmospheric transition line spectra of the night sky from the data in a pixel table(s) belonging to one exposure of (mostly) empty sky.

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXTABLE_SKY</td>
<td>raw</td>
<td>1</td>
<td>Input pixel table. If the pixel table is not already flux calibrated, the corresponding flux calibration frames should be given as well.</td>
</tr>
<tr>
<td>EXTINCT_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Atmospheric extinction table</td>
</tr>
<tr>
<td>STD_RESPONSE</td>
<td>calib</td>
<td>1</td>
<td>Response curve as derived from standard star(s)</td>
</tr>
<tr>
<td>STD_TELLURIC</td>
<td>calib</td>
<td></td>
<td>Telluric absorption as derived from standard star(s) (optional)</td>
</tr>
<tr>
<td>SKY_LINES</td>
<td>calib</td>
<td>1</td>
<td>List of OH transitions and other sky lines</td>
</tr>
<tr>
<td>SKY_CONTINUUM</td>
<td>calib</td>
<td></td>
<td>Sky continuum to use (optional)</td>
</tr>
<tr>
<td>LSF_PROFILE</td>
<td>calib</td>
<td></td>
<td>Slice specific LSF parameters cubes (optional, more than one frame allowed)</td>
</tr>
<tr>
<td>SKY_MASK</td>
<td>calib</td>
<td></td>
<td>Sky mask to use (optional)</td>
</tr>
</tbody>
</table>

**Recipe parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fraction</td>
<td>double</td>
<td>0.75</td>
<td>Fraction of the image (without the ignored part) to be considered as sky. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 1.</td>
</tr>
<tr>
<td>ignore</td>
<td>double</td>
<td>0.05</td>
<td>Fraction of the image to be ignored. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 0.</td>
</tr>
<tr>
<td>sampling</td>
<td>double</td>
<td>0.31 25</td>
<td>Spectral sampling of the sky spectrum [Angstrom].</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>csampling</td>
<td>double</td>
<td>0.3125</td>
<td>Spectral sampling of the continuum spectrum [Angstrom].</td>
</tr>
<tr>
<td>crsigma</td>
<td>string</td>
<td>15.,0.</td>
<td>Sigma level clipping for cube-based and spectrum-based CR rejection. This has to be a string of two comma-separated floating-point numbers. The first value gives the sigma-level rejection for cube-based CR rejection (using &quot;median&quot;, see muse_scipost), the second value the sigma-level for spectrum-based CR cleaning. Both can be switched off, by passing zero or a negative value; by default, the spectrum-based rejection is switched off.</td>
</tr>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>4000.</td>
<td>Cut off the data below this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>10000.</td>
<td>Cut off the data above this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdaref</td>
<td>double</td>
<td>7000.</td>
<td>Reference wavelength used for correction of differential atmospheric refraction. The R-band (peak wavelength 7000 Angstrom) that is usually used for guiding, is close to the central wavelength of MUSE, so a value of 7000.0 Angstrom should be used if nothing else is known. A value less than zero switches DAR correction off.</td>
</tr>
</tbody>
</table>

### Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKY_MASK</td>
<td>Created sky mask</td>
</tr>
<tr>
<td>IMAGE_FOV</td>
<td>Whitelight image used to create the sky mask</td>
</tr>
<tr>
<td>SKY_SPECTRUM</td>
<td>Sky spectrum within the sky mask</td>
</tr>
<tr>
<td>SKY_LINES</td>
<td>Estimated sky line flux table</td>
</tr>
<tr>
<td>SKY_CONTINUUM</td>
<td>Estimated continuum flux spectrum</td>
</tr>
</tbody>
</table>

### Quality control parameters

The following quality control parameters are available for the **muse_create_sky** products:

- **ESO.QC.SKY.THRESHOLD**: Threshold in the white light considered as sky, used to create this mask
- **ESO.QC.SKY.LINE1.NAME**: Name of the strongest line in group k
- **ESO.QC.SKY.LINE1.AWAV**: Wavelength (air) of the strongest line of group l
- **ESO.QC.SKY.LINE1.FLUX**: Flux of the strongest line of group l
- **ESO.QC.SKY.CONT.FLUX**: Total flux of the continuum
- **ESO.QC.SKY.CONT.MAXDEV**: Maximum (absolute value) of the derivative of the continuum spectrum
7.2.3 muse_astrometry

Compute an astrometric solution.

**Description**

Merge pixel tables from all IFUs, apply correction for differential atmospheric refraction, optionally apply flux calibration and telluric correction (if the necessary input data was given), and resample the data from all exposures into a datacube. Use the cube to detect objects which are then matched to their reference positions from which a two-dimensional WCS solution is computed.

The main output is the ASTROMETRY_WCS file which is a bare FITS header containing the world coordinate solution. The secondary product is DATA_CUBE_ASTROMETRIC, it is not needed for further processing but can be used for verification and debugging. It contains the reconstructed cube and two images created from it in further FITS extensions: a white-light image and the special image created from the central planes of the cube used to detect and centroid the stars (as well as its variance).

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXTABLE_ASTROMETRIC</td>
<td>raw</td>
<td>1</td>
<td>Pixel table of an astrometric field</td>
</tr>
<tr>
<td>ASTROMETRY_REFERENCE</td>
<td>calib</td>
<td>1</td>
<td>Reference table of known objects for astrometry</td>
</tr>
<tr>
<td>EXTINCT_TABLE</td>
<td>calib</td>
<td></td>
<td>Atmospheric extinction table (optional)</td>
</tr>
<tr>
<td>STD_RESPONSE</td>
<td>calib</td>
<td></td>
<td>Response curve as derived from standard star(s) (optional)</td>
</tr>
<tr>
<td>STD_TELLURIC</td>
<td>calib</td>
<td></td>
<td>Telluric absorption as derived from standard star(s) (optional)</td>
</tr>
</tbody>
</table>

**Recipe parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>centroid</td>
<td>string</td>
<td>moffat, gaussian, moffat, box</td>
<td>Centroiding method to use for objects in the field of view. &quot;gaussian&quot; and &quot;moffat&quot; use 2D fits to derive the centroid, &quot;box&quot; is a simple centroid in a square box.</td>
</tr>
<tr>
<td>detsigma</td>
<td>double</td>
<td>3.</td>
<td>Source detection sigma level to use.</td>
</tr>
<tr>
<td>radius</td>
<td>double</td>
<td>5.</td>
<td>Initial radius in pixels for pattern matching identification in the astrometric field.</td>
</tr>
<tr>
<td>faccuracy</td>
<td>double</td>
<td>5.</td>
<td>Factor of initial accuracy relative to mean positional accuracy of the measured positions to use for pattern matching.</td>
</tr>
<tr>
<td>niter</td>
<td>int</td>
<td>2</td>
<td>Number of iterations of the astrometric fit.</td>
</tr>
<tr>
<td>rejsigma</td>
<td>double</td>
<td>3.</td>
<td>Rejection sigma level of the astrometric fit.</td>
</tr>
<tr>
<td>rotcenter</td>
<td>string</td>
<td>-0.01,-1.20</td>
<td>Center of rotation of the instrument, given as two comma-separated floating point values in pixels.</td>
</tr>
</tbody>
</table>

Continued on next page
- continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>4000.</td>
<td>Cut off the data below this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>10000.</td>
<td>Cut off the data above this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdaref</td>
<td>double</td>
<td>7000.</td>
<td>Reference wavelength used for correction of differential atmospheric refraction. The R-band (peak wavelength 7000 Angstrom) that is usually used for guiding, is close to the central wavelength of MUSE, so a value of 7000.0 Angstrom should be used if nothing else is known. A value less than zero switches DAR correction off.</td>
</tr>
<tr>
<td>darccheck</td>
<td>string</td>
<td>none,</td>
<td>Carry out a check of the theoretical DAR correction using source centroiding. If &quot;correct&quot; it will also apply an empirical correction.</td>
</tr>
</tbody>
</table>

Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACUBE_ASTROMETRY</td>
<td>Reduced astrometry field exposure</td>
</tr>
<tr>
<td>ASTROMETRY_WCS</td>
<td>Astrometric solution</td>
</tr>
</tbody>
</table>

Quality control parameters

The following quality control parameters are available for the `muse_astrometry` products:

- `ESO.QC.ASTRO.NDET` Number of detected sources in output cube.
- `ESO.QC.ASTRO.LAMBDAA` Wavelength of plane in combined cube that was used for object detection.
- `ESO.QC.ASTRO.POSk.X` Position of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.ASTRO.POSk.Y` Position of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.ASTRO.FWHMk.X` FWHM of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.ASTRO.FWHMk.Y` FWHM of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.ASTRO.FWHM.VALID` Number of detected sources with valid FWHM in output cube.
- `ESO.QC.ASTRO.FWHM.MEDIAN` Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.
- `ESO.QC.ASTRO.FWHM.MAD` Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.
- `ESO.QC.ASTRO.NSTARS` Number of stars identified for the astrometric solution
- `ESO.QC.ASTRO.SCALE.X` Computed scale in x-direction
ESO.QC.ASTRO.SCALE.Y Computed scale in y-direction
ESO.QC.ASTRO.ANGLE.X Computed angle in x-direction
ESO.QC.ASTRO.ANGLE.Y Computed angle in y-direction
ESO.QC.ASTRO.MEDRES.X Median residuals of astrometric fit in x-direction
ESO.QC.ASTRO.MEDRES.Y Median residuals of astrometric fit in y-direction

7.2.4 muse_scipost

Prepare reduced and combined science products.

Description
Sort input pixel tables into lists of files per exposure, merge pixel tables from all IFUs of each exposure. Correct each exposure for differential atmospheric refraction (unless --lambdaRaf is far outside the MUSE wavelength range). Then the flux calibration is carried out, if a response curve was given in the input; it includes a correction of telluric absorption, if a telluric absorption correction file was given. Then the sky subtraction is carried out (unless --skyMethod="none"), either directly subtracting an input sky continuum and an input sky emission line(s) (for --skyMethod="subtract-model"), or (--skyMethod="model") create a sky spectrum from the darkest fraction (--skyModel_fractions) of the field of view, then fitting and subtracting sky emission lines using an initial estimate of the input sky line(s); then the continuum (residuals after subtracting the sky lines from the sky spectrum) is subtracted as well. If --save contains "skymodel", all sky-related products are saved for each exposure. Afterwards the data is corrected for the radial velocity of the observer (--rVcorr), before the input (or a default) astrometric solution is applied. Now each individual exposure is fully reduced; the pixel tables at this stage can be saved by setting "individual" in --save.

If multiple exposures were given, they are then combined. If --save contains "combined", this final merged pixel table is saved.

Finally (if --save contains "cube"), the data is resampled into a datacube, using all parameters given to the recipe. The extent and orientation of the cube is normally computed from the data itself, but this can be overridden by passing a file with the output world coordinate system (OUTPUT_WCS), for example a MUSE cube. This can also be used to sample the wavelength axis logarithmically (in that file set "CTYPE3='WAVE-LOG'"). As a last step, the computed cube is integrated over all filter functions given (--filter) that are also present in the input filter list table.

Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXTABLE_OBJECT</td>
<td>raw</td>
<td>1</td>
<td>Pixel table of a science object</td>
</tr>
<tr>
<td>EXTINCT_TABLE</td>
<td>calib</td>
<td>1</td>
<td>Atmospheric extinction table</td>
</tr>
<tr>
<td>STD_RESPONSE</td>
<td>calib</td>
<td>1</td>
<td>Response curve as derived from standard star(s)</td>
</tr>
<tr>
<td>STD_TELLURIC</td>
<td>calib</td>
<td></td>
<td>Telluric absorption correction as derived from standard star(s) (optional)</td>
</tr>
<tr>
<td>ASTROMETRY_WCS</td>
<td>calib</td>
<td></td>
<td>Astrometric solution derived from astrometric science frame (optional)</td>
</tr>
<tr>
<td>OFFSET_LIST</td>
<td>calib</td>
<td></td>
<td>List of coordinate offsets (and optional flux scale factors) (optional)</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTER_LIST</td>
<td>calib</td>
<td></td>
<td>File to be used to create field-of-view images. (optional)</td>
</tr>
<tr>
<td>OUTPUT_WCS</td>
<td>calib</td>
<td></td>
<td>WCS to override output cube location / dimensions (optional)</td>
</tr>
<tr>
<td>SKY_LINES</td>
<td>calib</td>
<td></td>
<td>List of OH transitions and other sky lines (optional)</td>
</tr>
<tr>
<td>SKY_CONTINUUM</td>
<td>calib</td>
<td></td>
<td>Sky continuum to use (optional)</td>
</tr>
<tr>
<td>LSF_PROFILE</td>
<td>calib</td>
<td></td>
<td>Slice specific LSF parameters. (optional, more than one frame allowed)</td>
</tr>
<tr>
<td>SKY_MASK</td>
<td>calib</td>
<td></td>
<td>Sky mask to use (optional)</td>
</tr>
</tbody>
</table>

Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>save</td>
<td>string</td>
<td><strong>cube,</strong> <strong>skymodel</strong></td>
<td>Select output product(s) to save. Can contain one or more of &quot;cube&quot;, &quot;skymodel&quot;, &quot;individual&quot;, &quot;positioned&quot;, &quot;combined&quot;, and &quot;stacked&quot;. If several options are given, they have to be comma-separated. (&quot;cube&quot;: output cube and associated images, if this is not given, no final resampling is done at all -- &quot;skymodel&quot;: up to four additional output products about the effectively used sky that was subtracted with the &quot;model&quot; method -- &quot;individual&quot;: fully reduced pixel table for each individual exposure -- &quot;positioned&quot;: fully reduced and positioned pixel table for each individual exposure, the difference to &quot;individual&quot; is that here, the output pixel tables have coordinates in RA and DEC; this is only useful, if both the relative exposure weighting and the final resampling are to be done externally -- &quot;combined&quot;: fully reduced and combined pixel table for the full set of exposures, the difference to &quot;positioned&quot; is that all pixel tables are combined into one, with an added weight column; this is useful, if only the final resampling step is to be done separately -- &quot;stacked&quot;: an additional output file in form of a 2D column-stacked image, i.e. x direction is pseudo-spatial, y direction is wavelength.) The resampling technique to use for the final output cube.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Values</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>dx</td>
<td>double</td>
<td>0.0</td>
<td>Horizontal step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2&quot; for WFM, 0.075&quot; for NFM, 1.0 if data is in pixel units.</td>
</tr>
<tr>
<td>dy</td>
<td>double</td>
<td>0.0</td>
<td>Vertical step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2&quot; for WFM, 0.075&quot; for NFM, 1.0 if data is in pixel units.</td>
</tr>
<tr>
<td>dlambda</td>
<td>double</td>
<td>0.0</td>
<td>Wavelength step size (in Angstrom). Natural instrument sampling is used, if this is 0.0</td>
</tr>
<tr>
<td>c Dtype</td>
<td>string</td>
<td>median, iraf, mean, median</td>
<td>Type of statistics used for detection of cosmic rays during final resampling. &quot;iraf&quot; uses the variance information, &quot;mean&quot; uses standard (mean/stdev) statistics, &quot;median&quot; uses median and the median median of the absolute median deviation.</td>
</tr>
<tr>
<td>crsigma</td>
<td>double</td>
<td>15.0</td>
<td>Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.</td>
</tr>
<tr>
<td>rc</td>
<td>double</td>
<td>1.25</td>
<td>Critical radius for the &quot;renka&quot; resampling method.</td>
</tr>
<tr>
<td>pixfrac</td>
<td>double</td>
<td>0.8</td>
<td>Pixel down-scaling factor for the &quot;drizzle&quot; resampling method.</td>
</tr>
<tr>
<td>ld</td>
<td>int</td>
<td>1</td>
<td>Number of adjacent pixels to take into account during resampling in all three directions (loop distance); this affects all resampling methods except &quot;nearest&quot;.</td>
</tr>
<tr>
<td>format</td>
<td>string</td>
<td>Cube, Cube, Euro3D, xCube, xEuro3D</td>
<td>Type of output file format, &quot;Cube&quot; is a standard FITS cube with NAXIS=3 and multiple extensions (for data and variance). The extended &quot;x&quot; formats include the reconstructed image(s) in FITS image extensions within the same file.</td>
</tr>
<tr>
<td>weight</td>
<td>string</td>
<td>exptime, exptime, fwhm, none</td>
<td>Type of weighting scheme to use when combining multiple exposures. &quot;exptime&quot; just uses the exposure time to weight the exposures, &quot;fwhm&quot; uses the DIMM information in the header as well, &quot;none&quot; preserves an existing weight column in the input pixel tables without changes. The filter name(s) to be used for the output field-of-view image. Each name has to correspond to an EXTNAME in an extension of the FILTER_LIST file. If an unsupported filter name is given, creation of the respective image is omitted. If multiple filter names are given, they have to be comma separated.</td>
</tr>
</tbody>
</table>

Continued on next page
### Parameter Values Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>skymethod</td>
<td>string</td>
<td>model, none, subtract-model, model, simple</td>
<td>The method used to subtract the sky background. &quot;model&quot; should work in all cases, it uses a global sky spectrum model with a local LSF. If &quot;model&quot; is selected, calibration frames for SKY_LINES and LSF_PROFILE must be set; SKY_CONTINUUM and SKY_MASK are optional. If &quot;subtract-model&quot; is selected, precalculated sky lines and continuum are subtracted if specified by SKY_LINES and SKY_CONTINUUM. An LSF_PROFILE is necessary for the two model-based methods. &quot;simple&quot; directly subtracts a sky spectrum created from the data, without regard to LSF variations; it also works on data that was not flux calibrated.</td>
</tr>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>4000.</td>
<td>Cut off the data below this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>10000.</td>
<td>Cut off the data above this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdaref</td>
<td>double</td>
<td>7000.</td>
<td>Reference wavelength used for correction of differential atmospheric refraction. The R-band (peak wavelength 7000 Angstrom) that is usually used for guiding, is close to the central wavelength of MUSE, so a value of 7000.0 Angstrom should be used if nothing else is known. A value less than zero switches DAR correction off.</td>
</tr>
<tr>
<td>darcheck</td>
<td>string</td>
<td>none, none, check, correct</td>
<td>Carry out a check of the theoretical DAR correction using source centroiding. If &quot;correct&quot; it will also apply an empirical correction.</td>
</tr>
<tr>
<td>skymodel_fraction</td>
<td>double</td>
<td>0.75</td>
<td>Fraction of the image (without the ignored part) to be considered as sky. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 1.</td>
</tr>
<tr>
<td>skymodel_ignore</td>
<td>double</td>
<td>0.05</td>
<td>Fraction of the image to be ignored. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 0.</td>
</tr>
<tr>
<td>skymodel_sampling</td>
<td>double</td>
<td>0.3125</td>
<td>Spectral sampling of the sky spectrum [Angstrom].</td>
</tr>
<tr>
<td>skymodel_csampling</td>
<td>double</td>
<td>0.3125</td>
<td>Spectral sampling of the continuum spectrum [Angstrom].</td>
</tr>
</tbody>
</table>
- continued from previous page

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sky_crsigma</td>
<td>string</td>
<td>15.,0.</td>
<td>Sigma level clipping for cube-based and spectrum-based CR rejection when creating the sky spectrum. This has to be a string of two comma-separated floating-point numbers. The first value gives the sigma-level rejection for cube-based CR rejection (using &quot;median&quot;, see muse_scipost), the second value the sigma-level for spectrum-based CR cleaning. Both can be switched off, by passing zero or a negative value; by default, the spectrum-based rejection is switched off.</td>
</tr>
<tr>
<td>rvcorr</td>
<td>string</td>
<td>bary, bary, helio, geo, none</td>
<td>Correct the radial velocity of the telescope with reference to either the barycenter of the Solar System (bary), the center of the Sun (helio), or to the center of the Earth (geo). If false, skip any astrometric calibration, even if one was passed in the input set of files. This causes creation of an output cube with a linear WCS and may result in errors. If you want to use a sensible default, leave this true but do not pass an ASTROMETRY_WCS.</td>
</tr>
<tr>
<td>astrometry</td>
<td>boolean</td>
<td>true</td>
<td>If false, skip any astrometric calibration, even if one was passed in the input set of files. This causes creation of an output cube with a linear WCS and may result in errors. If you want to use a sensible default, leave this true but do not pass an ASTROMETRY_WCS.</td>
</tr>
</tbody>
</table>

**Product frames**

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACUBE_FINAL</td>
<td>Output datacube</td>
</tr>
<tr>
<td>IMAGE_FOV</td>
<td>Field-of-view images corresponding to the &quot;filter&quot; parameter.</td>
</tr>
<tr>
<td>OBJECT_RESAMPLED</td>
<td>Stacked image (if --save contains &quot;stacked&quot;)</td>
</tr>
<tr>
<td>PIXTABLE_REDUCED</td>
<td>Fully reduced pixel tables for each exposure (if --save contains &quot;individual&quot;)</td>
</tr>
<tr>
<td>PIXTABLE_POSITIONED</td>
<td>Fully reduced and positioned pixel table for each individual exposure (if --save contains &quot;positioned&quot;)</td>
</tr>
<tr>
<td>PIXTABLE_COMBINED</td>
<td>Fully reduced and combined pixel table for the full set of exposures (if --save contains &quot;combined&quot;)</td>
</tr>
<tr>
<td>SKY_MASK</td>
<td>Created sky mask (if --skymethod=smodel and --save contains &quot;skymodel&quot;)</td>
</tr>
<tr>
<td>SKY_SPECTRUM</td>
<td>Sky spectrum within the sky mask (if --skymethod=smodel and --save contains &quot;skymodel&quot;)</td>
</tr>
<tr>
<td>SKY_LINES</td>
<td>Estimated sky line flux table (if --skymethod=smodel and --save contains &quot;skymodel&quot;)</td>
</tr>
<tr>
<td>SKY_CONTINUUM</td>
<td>Estimated continuum flux spectrum (if --skymethod=smodel and --save contains &quot;skymodel&quot;)</td>
</tr>
</tbody>
</table>
Quality control parameters

The following quality control parameters are available for the `muse_scipost` products:

- **ESO.QC.SCIPOST.NDET**: Number of detected sources in output cube.
- **ESO.QC.SCIPOST.LAMBDNA**: Wavelength of plane in combined cube that was used for object detection.
- **ESO.QC.SCIPOST.POSk.X**: Position of source k in x-direction in combined frame.
- **ESO.QC.SCIPOST.POSk.Y**: Position of source k in y-direction in combined frame.
- **ESO.QC.SCIPOST.FWHMk.X**: FWHM of source k in x-direction in combined frame.
- **ESO.QC.SCIPOST.FWHMk.Y**: FWHM of source k in y-direction in combined frame.
- **ESO.QC.SCIPOST.FWHM.NVALID**: Number of detected sources with valid FWHM in output cube.
- **ESO.QC.SCIPOST.FWHM.MEDIAN**: Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.
- **ESO.QC.SCIPOST.FWHM.MAD**: Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.
- **ESO.QC.SCIPOST.THRESHOLD**: Threshold in the white light considered as sky, used to create this mask.
- **ESO.QC.SCIPOST.LINE1.NAME**: Name of the strongest line in group 1.
- **ESO.QC.SCIPOST.LINE1.AWAV**: Wavelength (air) of the strongest line of group 1.
- **ESO.QC.SCIPOST.LINE1.FLUX**: Flux of the strongest line of group 1.
- **ESO.QC.SCIPOST.CONT.FLUX**: Total flux of the continuum.
- **ESO.QC.SCIPOST.CONT.MAXDEV**: Maximum (absolute value) of the derivative of the continuum spectrum.

7.2.5 `muse_exp_align`

Create a coordinate offset table to be used to align exposures during exposure combination.

**Description**

Compute the coordinate offset for each input field-of-view image with respect to a reference. The created list of coordinate offsets can then be used in `muse_exp_combine` as the field coordinate offsets to properly align the exposures during their combination.

**Input frames**

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE_FOV</td>
<td>raw</td>
<td>2</td>
<td>Input field-of-view images (more than 2 frames allowed)</td>
</tr>
</tbody>
</table>

**Recipe parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search</td>
<td>string</td>
<td>30.,4.,2.</td>
<td>Search radius (in arcsec) for each iteration of the offset computation.</td>
</tr>
</tbody>
</table>
Product frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET_LIST</td>
<td>List of computed coordinate offsets.</td>
</tr>
</tbody>
</table>

Quality control parameters

The following quality control parameters are available for the `muse_exp_align` products:

- `ESO.QC.EXPALIGN.NDETi` Number of detected sources for input image i
- `ESO.QC.EXPALIGN.NMATCHi` Median number of matches of input image i with other images
- `ESO.QC.EXPALIGN.NMATCH.MIN` Minimum of the median number of matches for all input images
- `ESO.QC.EXPALIGN.NOMATCH` Number of input images that do not have any matches with other images

7.2.6 `muse_exp_combine`

Combine several exposures into one datacube.
Description

Sort reduced pixel tables, one per exposure, by exposure and combine them with applied weights into one final datacube.

Input frames

<table>
<thead>
<tr>
<th>Category</th>
<th>Type</th>
<th>min</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXTABLE_REduced</td>
<td>raw</td>
<td>2</td>
<td>Input pixel tables (more than 2 frames allowed)</td>
</tr>
<tr>
<td>OFFSET_LIST</td>
<td>calib</td>
<td></td>
<td>List of coordinate offsets (and optional flux scale factors) (optional)</td>
</tr>
<tr>
<td>FILTER_LIST</td>
<td>calib</td>
<td></td>
<td>File to be used to create field-of-view images. (optional)</td>
</tr>
<tr>
<td>OUTPUT_WCS</td>
<td>calib</td>
<td></td>
<td>WCS to override output cube location / dimensions (optional)</td>
</tr>
</tbody>
</table>

Recipe parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>save</td>
<td>string</td>
<td>cube, other</td>
<td>Select output product(s) to save. Can contain one or more of &quot;cube&quot; (output cube and associated images; if this is not given, no resampling is done at all) or &quot;combined&quot; (fully reduced and combined pixel table for the full set of exposures; this is useful, if the final resampling step is to be done again separately). If several options are given, they have to be comma-separated.</td>
</tr>
<tr>
<td>resample</td>
<td>string</td>
<td>drizzle, nearest, linear, quadratic, renka, drizzle, lanczos</td>
<td>The resampling technique to use for the final output cube.</td>
</tr>
<tr>
<td>dx</td>
<td>double</td>
<td>0.0</td>
<td>Horizontal step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2&quot; for WFM, 0.075&quot; for NFM, 1.0 if data is in pixel units.</td>
</tr>
<tr>
<td>dy</td>
<td>double</td>
<td>0.0</td>
<td>Vertical step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2&quot; for WFM, 0.075&quot; for NFM, 1.0 if data is in pixel units.</td>
</tr>
<tr>
<td>dlambda</td>
<td>double</td>
<td>0.0</td>
<td>Wavelength step size (in Angstrom). Natural instrument sampling is used, if this is 0.0</td>
</tr>
</tbody>
</table>

Continued on next page
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crtype</td>
<td>string</td>
<td>median, iraf, mean, median</td>
<td>Type of statistics used for detection of cosmic rays during final resampling. &quot;iraf&quot; uses the variance information, &quot;mean&quot; uses standard (mean/stdev) statistics, &quot;median&quot; uses median and the median median of the absolute median deviation.</td>
</tr>
<tr>
<td>crsigma</td>
<td>double</td>
<td>10.</td>
<td>Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.</td>
</tr>
<tr>
<td>rc</td>
<td>double</td>
<td>1.25</td>
<td>Critical radius for the &quot;renka&quot; resampling method.</td>
</tr>
<tr>
<td>pixfrac</td>
<td>double</td>
<td>0.6</td>
<td>Pixel down-scaling factor for the &quot;drizzle&quot; resampling method.</td>
</tr>
<tr>
<td>ld</td>
<td>int</td>
<td>1</td>
<td>Number of adjacent pixels to take into account during resampling in all three directions (loop distance); this affects all resampling methods except &quot;nearest&quot;.</td>
</tr>
<tr>
<td>format</td>
<td>string</td>
<td>Cube, Euro3D, xCube, xEuro3D</td>
<td>Type of output file format, &quot;Cube&quot; is a standard FITS cube with NAXIS=3 and multiple extensions (for data and variance). The extended &quot;x&quot; formats include the reconstructed image(s) in FITS image extensions within the same file.</td>
</tr>
<tr>
<td>weight</td>
<td>string</td>
<td>exptime, fwhm, none</td>
<td>Type of weighting scheme to use when combining multiple exposures. &quot;exptime&quot; just uses the exposure time to weight the exposures, &quot;fwhm&quot; uses the DIMM information in the header as well, &quot;none&quot; preserves an existing weight column in the input pixel tables without changes.</td>
</tr>
<tr>
<td>filter</td>
<td>string</td>
<td>white</td>
<td>The filter name(s) to be used for the output field-of-view image. Each name has to correspond to an EXTNAMES in an extension of the FILTER_LIST file. If an unsupported filter name is given, creation of the respective image is omitted. If multiple filter names are given, they have to be comma separated.</td>
</tr>
<tr>
<td>lambdamin</td>
<td>double</td>
<td>4000.</td>
<td>Cut off the data below this wavelength after loading the pixel table(s).</td>
</tr>
<tr>
<td>lambdamax</td>
<td>double</td>
<td>10000.</td>
<td>Cut off the data above this wavelength after loading the pixel table(s).</td>
</tr>
</tbody>
</table>

### Product Frames

The following product frames are created by the recipe:

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATACUBE_FINAL</td>
<td>Output datacube (if --save contains &quot;cube&quot;)</td>
</tr>
</tbody>
</table>

Continued on next page
--- continued from previous page

<table>
<thead>
<tr>
<th>Default file name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAGE_FOV</td>
<td>Field-of-view images corresponding to the &quot;filter&quot; parameter (if --save contains &quot;cube&quot;).</td>
</tr>
<tr>
<td>PIXTABLE_COMBINED</td>
<td>Combined pixel table (if --save contains &quot;combined&quot;)</td>
</tr>
</tbody>
</table>

**Quality control parameters**

The following quality control parameters are available for the `muse_exp_combine` products:

- `ESO.QC.EXPCOMB.NDET` Number of detected sources in combined cube.
- `ESO.QC.EXPCOMB.LAMBDAX` Wavelength of plane in combined cube that was used for object detection.
- `ESO.QC.EXPCOMB.POSk.X` Position of source k in x-direction in combined cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.EXPCOMB.POSk.Y` Position of source k in y-direction in combined cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.EXPCOMB.FWHMK.X` FWHM of source k in x-direction in combined cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.EXPCOMB.FWHMK.Y` FWHM of source k in y-direction in combined cube. If the FWHM measurement fails, this value will be -1.
- `ESO.QC.EXPCOMB.FWHM.NVALID` Number of detected sources with valid FWHM in combined cube.
- `ESO.QC.EXPCOMB.FWHM.MEDIAN` Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in combined cube. If less than three sources with valid FWHM are detected, this value is zero.
- `ESO.QC.EXPCOMB.FWHM.MAD` Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in combined cube. If less than three sources with valid FWHM are detected, this value is zero.
Chapter 8

Tips & Troubleshooting

8.1 The output of the logfile

The logfile contains a lot of information that is related to the data reduction. Especially, if you encounter a problem, reading the logfile is likely to give you an idea at which point in the process the problem occurred. The logfile displays the following messages preceded by a time-stamp:

[ INFO ] - These lines tell the user what processing the pipeline is doing, at which point, and with which files.
[ DEBUG ] - Here more technical details and information are given (e.g. the number of pixels rejected in a cosmic ray rejection). Usually, one needs to change settings to see them (i.e. use --msg-level=debug or --log-level=debug with esorex). This should be done for all bug reports, but should not be necessary for normal operations.
[ WARNING ] - These messages warn about possible anomalies in the data. They also point out non-standard settings. They do not cause the pipeline to fail, but it is wise to check the data carefully afterwards.
[ ERROR ] - These are lines where a process in the pipeline could not finish properly or where a significant part of the process failed. The error code and the corresponding line in the code is usually printed. If possible, an explanation is given of why the failure occurred.

8.2 Restricting wavelength ranges

All post-processing recipes read pixel tables. When testing such steps of the data reduction, it can be benificial to work on only a subset of the data, like a small wavelength range. All relevant recipes therefore support the --lambdamin and --lambdamax recipe parameters. This causes the code to still read the full pixel tables, but then the pixels with wavelengths not in the given range are discarded. This can be used to speed up processing and constrain memory consumption, e.g. to test parameter ranges that affect the cube reconstruction. Note, however, that they may have unforeseen consequences if e.g. the data are truncated right on a bright sky line.

8.3 Debugging options

Certain environmental variables for testing and debugging were created while developing the pipeline. Many of them might prove useful to you, if your data cannot be reduced with the usual options that are

\[ Since the pixel tables are not and cannot be sorted, reading the full tables is necessary. It causes a temporary peak in memory usage to at least the size of the pixel table. \]
exposed through the recipe interface. Also, you might want to dig deeper into the analysis of what is done to your data during the reduction process.

As these are environment variables and not recipe options, they are set outside of the recipe call. For example, in the bash shell (or your bash script) this would look like this:

```
/home/user> export MUSE_DEBUG_WAVECAL=1
```

In a Python-CPL script, it is set as property of the corresponding recipe, however, not as a parameter, but as an environment variable. For example:

```python
muse_wavecal.env['MUSE_DEBUG_WAVECAL'] = 1
```

You can find a list of the environment variables at the end of the README file of your MUSE installation (usually in `$prefix/share/doc/esopipes/muse-1.1.90`). Remember that these variables are entirely optional. Please proceed with caution when using them, they might generate a large number of files, output that you may be unfamiliar with, or cause unexpected side effects.

### 8.4 Tools for debugging and verification

Usually, the QC parameters as documented in Sect. 7 as well as message printed by should give a good basis to verify that the processing worked as expected. But in some cases, as visual verification is necessary. In other cases, tools are needed to aid debugging of a problem. A few such tools are shipped with the MUSE pipeline and get installed with it (into `$prefix/bin`), they are described in this section.

The visual tools use gnuplot\(^2\) for plotting.\(^3\) All tools mentioned here give a usage hint when called without parameters.

#### 8.4.1 Verification of the tracing solution

When one has doubts about the validity of the tracing solution computed by the `muse_flat` recipe, one can specify the `--samples` parameter so that the extra output product `TRACE_SAMPLES` is written (one file per IFU).

This file contains all tracing samples computed by the recipe, i.e. left and right edge as well as the slice center at many vertical positions. These can be plotted using the tool `muse_trace_plot_samples`. If just using this file, only the central two slices are plotted:

```
muse_trace_plot_samples TRACE_SAMPLES-06.fits
```

If one also passes the number of the slices to show, one can e.g. plot all slices:

```
muse_trace_plot_samples -s1 1 -s2 48 TRACE_SAMPLES-06.fits
```

*Tip:* when the default gnuplot setup is used (with the x11, wxt, or qt "terminals"), one can use the right mouse button on the window that appears to zoom the display to a rectangular region.

When also passing the tracing table on the command line, the tool plots the polynomial solutions for both edges and the center over the crosses that mark the sampling points:


\(^3\)The plots can hence be customized in the same way as other gnuplot-based scripts. One can use e.g. using the file `$HOME/.gnuplot` to set up the preferred terminal type or cause gnuplot to write to a file instead of displaying a window.
Figure 8.1: The graphical window showing the output of the `muse_trace_plot_samples` tool, then plotting slices 12 to 20 in IFU 6, using the trace samples table, the trace table, and the master flat-field image (see text for details).

```
muse_trace_plot_samples -s1 1 -s2 48 TRACE_SAMPLES-06.fits TRACE_TABLE-06.fits
```

Here, one has to be careful to select files that belong to the same IFU! Then one can visually verify that the polynomial solution matches the individual traced points. Finally, one can also use the master flat-field product as background of the plot, so that one can actually check that the tracing points were correctly computed:

```
muse_trace_plot_samples -s1 12 -s2 20 TRACE_SAMPLES-06.fits TRACE_TABLE-06.fits \ MASTER_FLAT-06.fits
```

Plotting this may take a while, so it’s advisable to only use a subset of the slices. The result of this command is shown in Fig. 8.1.

The widths of the slices on the CCD should be around 77 pixels, but their actual widths may slowly vary between top and bottom of the CCD, and between the slices near the edges and in the middle of the CCD. The tool `muse_trace_plot_widths` was written to help assess that there are no sudden jumps in the tracing. When called with a tracing samples table, the samples of all slices are shown, as displayed in Fig. 8.2. A color gradient (from green on the left of the CCD to red on the right) plus different symbols are used to make the slices distinguishable. It is apparent that the slices on the edges of the CCD are the widest (above 78 pix) while those near the center of the CCD are narrow (below 76 pixels).
Figure 8.2: The graphical window showing the output of the `muse_trace_plot_widths` tool, plotting slices 1 to 48 of IFU 6, using the trace samples table (see text for details).

8.4.2 Verification of the wavelength solution

The tool `muse_wave_plot_residuals` can be used to verify the two-dimensional wavelength solution of each slice or of all slices of one IFU. To use it one needs to run the `muse_wavecal` recipe with the `--residuals` option, so that the extra product `WAVECAL_RESIDUALS` is created. Then one can run e.g.

```
muse_wave_plot_residuals WAVECAL_RESIDUALS-10.fits
```

and get a 2D map in CCD coordinates of the residuals of all the computed arc line centers with respect to the final solution. This is displayed in Fig. 8.3. There, one can see regions on the CCD that are not covered by arc lines as white patches, and the points with the strongest blue and red colors give the strongest deviations from the final solution. One can use the same command to change the vertical axis of the plot from CCD pixels to wavelength, using the `-l` parameter:

```
muse_wave_plot_residuals -l WAVECAL_RESIDUALS-10.fits
```

In case one wants to look at only one slice, one can use the `-s` parameter with a slice number; color cuts are adjustable using the `-c` parameter with two numbers, and one can study a different iteration (by default, the final iteration of the fit in each slice is selected), using `-i` and a positive integer.

When looking more in detail into the solution of a single slice, one can use the `muse_wave_plot_column` tool. This needs both the wavelength calibration table and the table with the residuals (make sure to pass the tables of the same run and IFU!). It can be used on the data of a single slice (parameter `-s`) or on a single CCD column (-`c`). It is most useful when displaying the vertical axis as residuals, using `-r`. Fig. 8.4 shows the output of the command

```
muse_wave_plot_column -s 12 -r WAVECAL_TABLE-10.fits WAVECAL_RESIDUALS-10.fits
```
Figure 8.3: The graphical window showing the output of the muse_wave_plot_residuals tool, plotting all slices of IFU 10, using the wavelength calibration residuals table (see text for details).

This is an example of a good calibration with low residuals (the final RMS for the solution in this slice was 0.030 Å). The tool has automatically selected all columns belonging to this slice and colored them according to their horizontal position on the CCD (green is left, red is right), and used different symbols. As one can see, the fainter arc lines (like the NeI line at 5400.6 Å) have typically a much larger spread of residuals than the bright lines (e.g. NeI at 6678.3 Å). With default parameters of muse_wavecal (--fitweighting=cerrscatter) the weak lines are hence weighted much less in the fit of the wavelength solution than bright lines.

8.4.3 Handling of MUSE pixel tables

Since MUSE pixel tables are heavily used as intermediate data products, and they have one special column that is not easy to interpret (the "origin" column), a tool was added to make its content easily readable, muse_pixtable_dump. One should always use the -c parameter to limit the number of rows that are displayed, otherwise it might take very long to complete. One should also give the starting row of the region that one is interested in, using -s:

muse_pixtable_dump -s 100000 -c 10 PIXTABLE_OBJECT_0001-01.fits

This command results in the output:
Figure 8.4: The graphical window showing the output of the muse_wave_plot_column tool, plotting slice 12 of IFU 10, using the wavelength calibration residuals and wavelength calibration tables (see text for details).

Unlike other FITS-table related tools it interprets the "origin" column and all special FITS headers to resolve the originating CCD pixel, slice, IFU, and exposure number of each entry in the table. If the "exposure" column in the output displays zeros, then the pixel table only contains one exposure. The two "CCD" columns in the output give the coordinates on the trimmed image, while the "Raw" columns use the un-trimmed coordinates found in the unprocessed raw data from the instrument.

Another tool that might be useful is muse_pixtable_crop, to extract part of a pixel table into another file. The crop regions can be one of the two spatial axes, or the wavelength axis. The following command cuts the input table simultaneously in the x-direction and in wavelength:

```bash
muse_pixtable_crop -x1 -30 -x2 +30 -l1 5570 -l2 5583 PIXTABLE_OBJECT_0001-12.fits pt1-12_small.fits
```

Since the spatial pixel table columns change depending of the stage of the processing, care must by taken to use values for the correct units. The command therefore echoes the range specified with the units expected for a given pixel table. The command above outputs:

```
MUSE pixel table "PIXTABLE_OBJECT_0001-12.fits" (13485859 rows)
  cropping to lambda = 5570.00..5583.00 Angstrom
  xpos = -3.000e+01..3.000e+01 pix
  ypos = -1.288e+01..4.175e-01 pix
```

The tool muse_pixtable_erase_slice can be used to remove the data of a complete slice of one IFU from a pixel table. When run like this:

```bash
muse_pixtable_erase_slice PIXTABLE_OBJECT_0005-14.fits 14 10
```

The tool muse_pixtable_erase_slice can be used to remove the data of a complete slice of one IFU from a pixel table. When run like this:

```bash
muse_pixtable_erase_slice PIXTABLE_OBJECT_0005-14.fits 14 10
```
it erases slice 10 (numbering on the CCD) from a pixel table of IFU 14 as produced by \texttt{muse\_scibasic}.

The IFU number is always required on the command line, so that when given a pixel table with multiple IFUs in it, the tool knows for which one to erase the slice:

\begin{verbatim}
muse\_pixmap\_erase\_slice \texttt{PIXTABLE\_REDUCED\_0001.fits 14 10} \texttt{PIXTABLE\_REDUCED\_0001\_e1410.fits}
\end{verbatim}

If a pixel table contains even multiple exposures, then it erases the given slice of the given IFU of all exposures.

### 8.4.4 Handling of MUSE bad pixel maps

Three tools exist to create or supplement MUSE bad pixel tables (the \texttt{BADPIX\_TABLE} files), that are optionally read by every recipe that starts from raw data (see Sect. 7.1).

If one wants to start such a bad pixel table, one can start with one of the image-based master calibrations. These contain a \texttt{DQ} extension that is a bad pixel map. While this is automatically used by subsequent recipes, one can transform it into a bad pixel table with the \texttt{muse\_badpix\_from\_dq} tool:

\begin{verbatim}
muse\_badpix\_from\_dq \texttt{MASTER\_FLAT-10.fits} \texttt{BADPIX\_TABLE-10.fits}
\end{verbatim}

This would create a new table containing all bad pixels that were detected by the \texttt{muse\_flat} recipes (including those that were present in all inputs into that run of \texttt{muse\_flat}). Since the tool gets the full FITS file including all headers of the output product, it can set up the correct FITS headers for the bad pixel table. This tool can also be used to merge flagged pixels from the \texttt{DQ} extension into an existing table:

\begin{verbatim}
muse\_badpix\_from\_dq \texttt{-i BADPIX\_TABLE\_in.fits MASTER\_FLAT-10.fits} \texttt{BADPIX\_TABLE\_out.fits}
\end{verbatim}

If one has manually recorded single bad pixels in an ASCII file or measured regions of bad pixels, one can use \texttt{muse\_badpix\_from\_ascii} or \texttt{muse\_badpix\_from\_region}. Here, one needs to specify the IFU that contains the bad pixels to store, since no FITS header with the information is available:

\begin{verbatim}
muse\_badpix\_from\_ascii \texttt{bad\_pixels.ascii} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
muse\_badpix\_from\_region \texttt{[10:12,100:2000]} \texttt{256} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
\end{verbatim}

\texttt{muse\_badpix\_from\_region} requires the region to be in the format \texttt{[x1:x2,y1:y2]} and also needs a Euro3D-like flag value as 2nd argument. The ASCII table has to contain three values per row (x-position, y-position, and flag value). By default, both tools expect the coordinates to be measured on the raw image; if they were determined on trimmed data instead, the \texttt{-t} argument has to be set:

\begin{verbatim}
muse\_badpix\_from\_ascii \texttt{-t bad\_pixels.ascii} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
muse\_badpix\_from\_region \texttt{-t \[10:12,100:2000\]} \texttt{256} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
\end{verbatim}

Again, these tools can be used to supplement the information in existing bad pixel tables; these can be passed in with the \texttt{-i} parameter:

\begin{verbatim}
muse\_badpix\_from\_ascii \texttt{-i BADPIX\_TABLE\_existing.fits} \texttt{bad\_pixels.ascii} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
muse\_badpix\_from\_region \texttt{-i \[10:12,100:2000\]} \texttt{256} \texttt{12} \texttt{BADPIX\_TABLE\_12.fits}
\end{verbatim}
8.4.5 Tools to deal with (partial) output cubes

The tool **muse_cube_combine** is useful, if one has to deal with a large dataset which cannot be fully combined with the **muse_scipost** or **muse_exp_combine** recipes. In that case, one can run **muse_exp_combine** several times, setting the wavelength limits (see Sect. 8.2) such that they overlap in only about 2 wavelength planes in the output cubes. One then has to take care to resample all sub-cubes to the same output grid, defined using the **OUTPUT_WCS** input (see Sect. 7.2.6 and Sect. A). They will then only be filled at the relevant wavelength ranges, the rest will contain NaNs. Then this tool can be used to combine them into a fully populated cube.

As an example, a dataset is too large to fit into memory at once, but does fit, if split into three sections. Then one runs **muse_exp_combine** with the parameters

```bash
esorex muse_exp_combine [...] --lambdamax=6285. ec.sof
mv DATACUBE_FINAL.fits CUBE_blue.fits
esorex muse_exp_combine [...] --lambdamin=6282.5 --lambdamax=7817.5 ec.sof
mv DATACUBE_FINAL.fits CUBE_green.fits
esorex muse_exp_combine [...] --lambdamin=7815. ec.sof
mv DATACUBE_FINAL.fits CUBE_red.fits
```

and the following sof

```bash
PIXTABLE_REDUCED_1.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_2.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_3.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_4.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_5.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_6.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_7.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_8.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_9.fits PIXTABLE_REDUCED
TEST_CUBE_header.fits OUTPUT_WCS
```

The FITS header of the FITS cube should then contain a fully defined world coordinate system in the $CDi_j$ notation that is large enough to include all the data. Then, **muse_cube_combine** can be run in the following way

```bash
muse_cube_combine CUBE_COMBINED.fits CUBE_blue.fits CUBE_green.fits CUBE_red.fits
```

This will automatically analyze the **PRO.REC1.PARAMi.NAME** and **PRO.REC1.PARAMi.VALUE** keywords in the headers of the **CUBE_<color>.fits** pipeline outputs to determine the wavelength range used, throw away the small overlaps (which are needed to guard against edge effects), sort the exposures according to the wavelength the cover, and then copy the relevant wavelength planes (both **DATA** and **STAT** extensions) into the single output cube.

The program **muse_cube_filter** can be used to integrate an existing cube in dispersion direction over a filter function. This is normally done by **muse_scipost** or **muse_exp_combine**, but if a filter was forgotten when running those recipes, or when **muse_cube_combine** was used, additional filter-images can be produced with this tool. Typical usage is

---

\[4\] It can be the header of a cube of one of the exposures as it comes out of **muse_scipost**, but edited by hand to be larger, or positioned differently.
muse_cube_filter -f Johnson_V,Cousins_R,Cousins_I DATACUBE.fits filter_list.fits
to create images in the $V$, $R$, and $I$ filters for the given cube. Only filters in the given filter list file are used.

In the case above, the images are saved as separate files with the filter name before the .fits extension, e.g. DATACUBE_Cousins_R.fits. One can instead select to save the images as new extensions in the input file, using the -x option:

muse_cube_filter -x -f Johnson_V, Cousins_R, Cousins_I DATACUBE.fits filter_list.fits

(Note that no backup of the original file is created!)

8.4.6 Other tools

The tool muse_geo_plot can be used to create a visual representation of a MUSE geometry table. Its operation has already been described briefly in Sect. 5.1.7.

Another tool that may be useful for some special cases is muse_fill_image. This might be used to e.g. create dark frames with a constant dark value. When doing this to feed files into the MUSE pipeline one should use a real output product as starting point. To create "dummy" flat-fields one would use:

```bash
for ifu in {01..24} ; do
    muse_fill_image -d 1. -q 0 -s 0. MASTER_FLAT-\${ifu}.fits DUMMY_FLAT-\${ifu}.fits &
done ; wait
```

This could be used to process technical exposures with the muse_scibasic recipe when real flat-fielding is not desirable but should not be used with scientific data, since this usually results in datacubes with weird artifacts.

8.5 Typical failure cases

In many cases, ERRORs and WARNINGs of the MUSE pipeline alert the user of a problem with the data or the reduction. In the following, a few likely cases and solutions for them are described.

8.5.1 Failed tracing

In some cases, vignetting of the slices on the edge of an IFU is severe enough to cause tracing to fail when running the muse_flat recipe. A typical error message is

```
[ ERROR ] muse_flat: muse_trace: [tid=005] The trace fit in slice 10 of IFU 6 failed
```

likely followed by more warnings and errors. The output TRACE_TABLE then contains invalid elements in the row for that slice. This causes subsequent problems for the wavelength calibration (muse_wavecal) and twilight handling (muse_twilight). Most critically, the science reduction (muse_scibasic) will stop when detecting such a broken trace table.

If this is the case, the most likely fix is to carefully adjust the --edgefrac parameter of the muse_flat recipe, from the default value downwards to e.g. 0.4 or 0.3 (when lowering the value too much, the pipeline might not be able to tell the slices apart any more.) It is likely, that warnings continue to appear for the darkest slices, so it is advisable to run the recipe with the --samples parameter, so that the TRACE_SAMPLES output is created. This can then be used with the command
Figure 8.5: The graphical window showing the output of the muse_trace_plot_samples tool (see text for details).

muse_trace_plot_samples -s1 9 -s2 11 TRACE_SAMPLES-06.fits TRACE_TABLE-06.fits \ MASTER_FLAT-06.fits

(see Fig. 8.5) to visually verify that the trace table contains a good description of the slice location.

8.6 Correcting coordinate offsets

When combining multiple exposures, the pipeline (the recipes muse_scipost and muse_exp_combine) does a good job to automatically recover the relative offsets from the information in the FITS header of each exposure, provided that two conditions are met: 1. the same VLT guide star was used to observe all exposures, 2. the exposures were all taken at the same position angle.

Since MUSE exposures are often observed using a dither pattern that involves 90 deg rotations, condition 2 is often not true, and the user has to provide offset corrections to the pipeline.\footnote{This is due to a slight decentering of the axis of the derotator, leading to a "derotator wobble". Future versions of the pipeline may be able to automatically correct this effect.} This can be done in two ways:

- Edit the FITS headers.
  Before starting muse_scipost or muse_exp_combine to create a combined cube, edit the FITS headers of the input files and replace the RA and DEC headers with corrected values that are measured externally.

- Provide offsets.
  This can currently be done using the columns RA_OFFSET and DEC_OFFSET in the OFFSET_LIST table. Each row in this table has to contain these two values (or zero, if the offset is negligible) and a DATE-OBS value corresponding to the exposure in question (in the DATE_OBS column).
  These offsets can be computed using the muse_exp_align recipe, if the exposures overlap significantly. Otherwise, the table can be edited by other means. Then, each number is the direct
difference of the measured position to the reference position (no \( \cos \delta \! \)!), the values are interpreted in units of degrees:

\[
\begin{align*}
\text{RA\_OFFSET} & = \text{RA}_{\text{measured}} - \text{RA}_{\text{reference}} \\
\text{DEC\_OFFSET} & = \text{DEC}_{\text{measured}} - \text{DEC}_{\text{reference}}
\end{align*}
\]

If the same offset should be applied on single exposures (e.g. for testing), one can input the same OFFSET\_LIST into muse\_scipost.

### 8.7 Correcting relative fluxes

In case an object was observed in non-photometric conditions and exposures need to be adapted relative to each other or to an absolute flux measurement, the column FLUX\_SCALE in the OFFSET\_LIST table can be adapted and set to values different from unity. As for the spatial offsets, both recipes muse\_scipost and muse\_exp\_combine react to this information. The scaling value is again assigned to the exposures using the DATE\_OBS string in the DATE\_OBS column of the table. The data of the exposures are multiplied with the scale factors in the table, the variance (STAT) is treated accordingly.
Appendix A

Data Format Description

The MUSE pipeline uses and produces a number of files in different formats, which are described in this section. For each data format, the structure of the FITS extensions is described, and the tags of all frames are listed that use this format.

A.1 Raw data files

A.1.1 RAW_IMAGE

Description

Raw CCD images taken with the MUSE instrument. Files coming from the instrument usually contain all 24 images from the IFUs in a single file.

FITS extensions

- 2D FITS image (int), may appear 24 times
  Data from one IFU.

Frame tags

- **BIAS**: `ESO.DPR.CATG=='CALIB' & ESO.DPR.TYPE=='BIAS'`
  Raw data taken with zero exposure time and closed shutter.

- **DARK**: `ESO.DPR.CATG=='CALIB' & ESO.DPR.TYPE=='DARK'`
  Raw data taken with positive exposure time and closed shutter.

- **FLAT**: `ESO.DPR.CATG=='CALIB' & ESO.DPR.TYPE=='FLAT,LAMP'`
  Raw exposure of a continuum lamp exposure illuminating the whole field of view.

- **ILLUM**: `ESO.DPR.CATG=='CALIB' & ESO.DPR.TYPE=='FLAT,LAMP,ILLUM'`
  Raw exposure of a continuum lamp exposure illuminating the whole field of view to correct for temperature dependent illumination changes.

- **AMPL**: `ESO.DPR.CATG=='TECHNICAL' & ESO.DPR.TYPE=='FLAT,LAMP,THRUPUT'`
  Raw exposure of a continuum lamp exposure illuminating the whole field of view, with special FITS headers containing pico amplifier measurements.
• ARC: ESO.DPR.CATG==‘CALIB’ & ESO.DPR.TYPE==’WAVE’
  Raw exposure of one or more arc lamps illuminating the whole field of view.

• MASK: ESO.DPR.CATG==‘CALIB’ & ESO.DPR.TYPE==’WAVE,MASK’
  Raw exposure of one or more arc lamps using the multi-pinhole mask for the determination of the relative location of the slices.

• SKYFLAT: ESO.DPR.CATG==‘CALIB’ & ESO.DPR.TYPE==’FLAT,SKY’
  Raw exposure of the twilight sky.

• OBJECT: ESO.DPR.CATG==’SCIENCE’ & ESO.DPR.TYPE==’OBJECT’
  Raw exposure of a science target.

• SKY: ESO.DPR.CATG==’SCIENCE’ & ESO.DPR.TYPE==’SKY’
  Raw exposure of an (almost) empty sky field.

• ASTROMETRY: ESO.DPR.CATG==’CALIB’ & ESO.DPR.TYPE==’ASTROMETRY’
  Raw exposure of an astrometric field.

• STD: ESO.DPR.CATG==’CALIB’ & ESO.DPR.TYPE==’STD’ | ’STD,TELLU’
  Raw exposure of a standard star field.

A.2 External calibration files

A.2.1 LINE_CATALOG

Description

This is a list of arc lines to be used for wavelength calibration. It is a FITS table, with one row for each line, which contains central wavelength of the line in question and a relative strength of the line, if known. The line fluxes may be used in the data reduction software as a first guess to the expected flux, the actual fluxes will be determined using line fitting. Additionally, to identify the lines and associate them with an arc lamp, a column ion (with element and ionization status) and a quality flag are needed. Optionally, a comment column might be useful.

FITS extensions

• FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>float</td>
<td>Wavelength [Å]</td>
</tr>
<tr>
<td>flux</td>
<td>float</td>
<td>Relative flux</td>
</tr>
<tr>
<td>ion</td>
<td>string</td>
<td>Ion from which the line originates</td>
</tr>
<tr>
<td>quality</td>
<td>int</td>
<td>Quality flag (0: undetected line, 1: line used for pattern matching, 2: line that is part of a multiplet, 3: good line, fully used, 5: bright and isolated line, use as FWHM reference)</td>
</tr>
<tr>
<td>comment</td>
<td>string</td>
<td>Optional comment (optional column)</td>
</tr>
</tbody>
</table>
A.2.2 SKY_LINES

Description

This type of file contains one or more binary tables with the relative fluxes on the sky emission lines. If both tables are present, they are merged, so that lines should not appear in both tables.

FITS extensions

- 'LINES': FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Line name</td>
</tr>
<tr>
<td>group</td>
<td>int</td>
<td>Line group id</td>
</tr>
<tr>
<td>lambda</td>
<td>double</td>
<td>Air wavelength [Å]</td>
</tr>
<tr>
<td>flux</td>
<td>double</td>
<td>Line flux ([10^{20}\text{erg/(s cm}^2\text{arcsec}^2]}]</td>
</tr>
<tr>
<td>dq</td>
<td>int</td>
<td>Quality of the entry (&gt;0: don’t use)</td>
</tr>
</tbody>
</table>

- 'OH_TRANSITIONS': FITS table, optional

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Transition name; like &quot;OH 8-3 P1e(22.5) 2&quot;</td>
</tr>
<tr>
<td>lambda</td>
<td>double</td>
<td>Air wavelength [Å]</td>
</tr>
<tr>
<td>v_u</td>
<td>int</td>
<td>Upper transition level</td>
</tr>
<tr>
<td>v_l</td>
<td>int</td>
<td>Lower transition level</td>
</tr>
<tr>
<td>nu</td>
<td>int</td>
<td>Vibrational momentum</td>
</tr>
<tr>
<td>E_u</td>
<td>double</td>
<td>Upper energy ([\text{J}])</td>
</tr>
<tr>
<td>J_u</td>
<td>double</td>
<td>Upper momentum</td>
</tr>
<tr>
<td>A</td>
<td>double</td>
<td>Transition probability</td>
</tr>
</tbody>
</table>

Frame tags

- SKY_LINES: \(\text{ESO.PRO.CATG==’SKY_LINES’}\)
  Catalog of OH transitions and other sky lines,
  \texttt{muse_create\_sky}: Estimated sky line flux table,
  \texttt{muse_scipost}: Estimated sky line flux table (if \(--skymethod=model\) and \(--save\) contains "sky-model")
A.2.3 ASTROMETRY_REFERENCE

Description

This FITS file lists astrometric sources in fields to be observed with MUSE as astrometric calibrators. It is used by the muse_astrometry recipe. One such table exists per field; the tables contain a list of (point) sources. Each row contains information about one object in the field.

The pipeline expects several such tables in multiple binary table extensions of a single FITS file. It then loads the one nearest to the observed sky position, using the RA and DEC keywords present in each FITS extension.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceID</td>
<td>string</td>
<td>Source identification</td>
</tr>
<tr>
<td>RA</td>
<td>double</td>
<td>Right ascension [deg]</td>
</tr>
<tr>
<td>DEC</td>
<td>double</td>
<td>Declination [deg]</td>
</tr>
<tr>
<td>filter</td>
<td>string</td>
<td>Filter name used for column mag</td>
</tr>
<tr>
<td>mag</td>
<td>double</td>
<td>Object (Vega) magnitude [mag]</td>
</tr>
</tbody>
</table>

Frame tags

- ASTROMETRY_REFERENCE: ESO.PRO.CATG=='ASTROMETRY_REFERENCE'
  Catalog of astrometry reference stars

A.2.4 EXTINCT_TABLE

Description

This is a simple binary FITS table with the dependency of the extinction on wavelength.

The wavelengths should cover at least the MUSE wavelength range. The atmospheric extinction values should be applicable for Paranal, ideally for the night of observations.

FITS extensions

- FITS table, may appear more than once

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>Wavelength [Angstrom]</td>
</tr>
<tr>
<td>extinction</td>
<td>double</td>
<td>Extinction [mag/airmass]</td>
</tr>
</tbody>
</table>

Frame tags

- EXTINCT_TABLE: ESO.PRO.CATG=='EXTINCT_TABLE'
Atmospheric extinction table

A.2.5 BADPIX_TABLE

Description
This is a FITS table with 24 extensions. This is used in the low-level recipes working on raw data. Each extension lists known bad pixels of one CCD.

FITS extensions

- FITS table, may appear 24 times

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xpos</td>
<td>int</td>
<td>X position of a bad pixel (on untrimmed raw data)</td>
</tr>
<tr>
<td>ypos</td>
<td>int</td>
<td>Y position of a bad pixel (on untrimmed raw data)</td>
</tr>
<tr>
<td>status</td>
<td>int</td>
<td>32bit bad pixel mask as defined by Euro3D</td>
</tr>
<tr>
<td>value</td>
<td>float</td>
<td>Extra value, e.g. depth for traps [count]</td>
</tr>
</tbody>
</table>

Frame tags

- BADPIX_TABLE: ESO.PRO.CATG=='BADPIX_TABLE'
  This file can be used to list known bad pixels that cannot be found by automated test on dark or flat-field frames.

A.2.6 STD_FLUX_TABLE

Description
This is a binary FITS table with the dependency of the flux on wavelength, and an optional column containing the error of the flux.

The wavelengths should cover at least the MUSE wavelength range.

The pipeline expects several such tables in multiple binary table extensions of a single FITS file. It then loads the one nearest to the observed sky position, using the RA and DEC keywords present in each FITS extension.

FITS extensions

- FITS table, may appear more than once

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>Wavelength [Angstrom]</td>
</tr>
<tr>
<td>flux</td>
<td>double</td>
<td>The standard star flux [erg/s/cm**2/Angstrom]</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fluxerr</td>
<td>double</td>
<td>Error of the standard star flux, optional (optional column) [erg/s/cm²/Angstrom]</td>
</tr>
</tbody>
</table>

### Frame tags

- **STD_FLUX_TABLE**: ESO.PRO.CATG==’STD_FLUX_TABLE’
  
  Reference flux distribution of a standard star. Such a table has to exist for each observed standard star.

### A.2.7 FILTER_LIST

**Description**

This FITS table contains all filter functions that can be used for image reconstruction. Each filter curve is contained within one sub-table.

**FITS extensions**

- **FITS table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>Wavelength [Angstrom]</td>
</tr>
<tr>
<td>throughput</td>
<td>double</td>
<td>Filter throughput (in fractions of 1)</td>
</tr>
</tbody>
</table>

**Frame tags**

- **FILTER_LIST**: ESO.PRO.CATG==’FILTER_LIST’
  
  File to be used to create field-of-view images.

### A.2.8 TELLURIC_REGIONS

**Description**

This FITS table regions of telluric absorption lines. It can be used to override the internal telluric bands used in the muse_standard recipe.

**FITS extensions**

- **FITS table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmin</td>
<td>double</td>
<td>Lower limit of the telluric region [Angstrom]</td>
</tr>
<tr>
<td>lmax</td>
<td>double</td>
<td>Upper limit of the telluric region [Angstrom]</td>
</tr>
</tbody>
</table>

Continued on next page
Frame tags

- **TELLURIC_REGIONS**: ESO.PRO.CATG=='TELLURIC_REGIONS'
  File to be used to override the internal telluric bands.

A.3 Data files created by the pipeline

A.3.1 MUSE_IMAGE

Description

A reduced CCD image of one IFU accompanied with quality and statistics information. These files follow the ESO specification for FITS files with data, bad pixel maps, and variance. The units of the extensions are given by the standard BUNIT keyword.

If the DQ extension is missing, the bad pixel status is then encoded as NaN values in the data and variance extensions.

FITS extensions

- **'DATA'**: 2D FITS image (float)
  Data values

- **'DQ'**: 2D FITS image (int), optional
  Euro3D data quality. This information is used to propagate information about bad pixels found e. g. in the processing of dark and flat-field exposures.

- **'STAT'**: 2D FITS image (float)
  Data variance

Frame tags

- **MASTER_BIAS**:
  `muse_bias`: Master bias

- **MASTER_DARK**:
  `muse_dark`: Master dark

- **MASTER_FLAT**:
  `muse_flat`: Master flat

- **ARC_RED_LAMP**:
  `muse_wavecal`: Reduced ARC image, per lamp

- **ARC_RED**:
  `muse_wavecal`: Reduced, combined master ARC image (if --lampwise=false or --resample=true)
• **MASK_REDUCE**D:
  muse\_geometry: Reduced pinhole mask images

• **MASK COMBINE**D:
  muse\_geometry: Combined pinhole mask image

• **IMAGE FOV**:
  muse\_create\_sky: Whitelight image used to create the sky mask,
  muse\_exp\_combine: Field-of-view images corresponding to the "filter" parameter,
  muse\_scipost\_make\_cube: Field-of-view images corresponding to the "filter" parameter.

• **MASTER AMPL**:
  muse\_ampl: Combined master AMPL image, written if --savemaster=true

• **OBJECT RED**:
  muse\_scibasic: Pre-processed CCD-based images for OBJECT input (if --saveimage=true)

• **OBJECT RESAMPLED**:
  muse\_scibasic: Resampled 2D image for OBJECT input (if --resample=true),
  muse\_scipost\_make\_cube: Stacked image (if --save contains "stacked")

• **STD RED**:
  muse\_scibasic: Pre-processed CCD-based images for STD input (if --saveimage=true)

• **STD RESAMPLED**:
  muse\_scibasic: Resampled 2D image for STD input (if --resample=true)

• **SKY RED**:
  muse\_scibasic: Pre-processed CCD-based images for SKY input (if --saveimage=true)

• **SKY RESAMPLED**:
  muse\_scibasic: Resampled 2D image for SKY input (if --resample=true)

• **ASTROMETRY RED**:
  muse\_scibasic: Pre-processed CCD-based images for ASTROMETRY input (if --saveimage=true)

• **ASTROMETRY RESAMPLED**:
  muse\_scibasic: Resampled 2D image for ASTROMETRY input (if --resample=true)

• **REDUCED RESAMPLED**:
  muse\_scibasic: Resampled 2D image (if --resample=true)

### A.3.2 PIXEL TABLE

**Description**

In the reduction approach of the MUSE pipeline, data need to be kept un-resampled until the very last step. The pixel tables used for this purpose can be saved at each intermediate reduction step and hence contain lists of pixels together with output coordinates and values.
By default, they are saved as multi-extension FITS images, where each extension corresponds to one table column. The name of the column is saved in the EXTNAME keyword, the unit in the standard BUNIT keyword. The units evolve through the processing.

In case CUNITs are available in the primary FITS header, they are used to track a spatial WCS for the construction of the reconstructed datacube, and are not to be used to interpret the data in the pixel table.

Formerly, pixel tables were written as binary FITS tables, and the MUSE pipeline can still read them for backward compatibility. In that format, the standard FITS table keywords in the table extension header are used to track column names (TTYPEi) and units (TUNITi). Reading and writing the binary table format is typically slower than the image format.

**FITS extensions**

- 'xpos': 1D FITS image (float)
  x position of a pixel within the field of view [pix, rad, deg]

- 'ypos': 1D FITS image (float)
  y position of a pixel within the field of view [pix, rad, deg]

- 'lambda': 1D FITS image (float)
  Wavelength assigned to the pixel [Angstrom]

- 'data': 1D FITS image (float)
  Data value [count, 10**(-20)*erg/s/cm**2/Angstrom]

- 'dq': 1D FITS image (int)
  32bit bad pixel status as defined by the Euro3D specification

- 'stat': 1D FITS image (float)
  The data variance [count**2, (10**(-20)*erg/s/cm**2/Angstrom)**2]

- 'origin': 1D FITS image (int)
  Encoded value of IFU and slice number, as well as x and y position in the raw (trimmed) data

- 'weight': 1D FITS image (float)
  The optional relative weight of this pixel

**Frame tags**

- **PIXTABLE_SUBTRACTED:**
  - `muse_lsf`: Subtracted combined pixel table, if --save_subtracted=true. This file contains only the subtracted arc lines that contributed to the LSF calculation. There are additional columns line_lambda and line_flux with the reference wavelength and the estimated line flux of the corresponding arc line.

- **PIXTABLE_OBJECT:**
  - `muse_scibasic`: Output pixel table for OBJECT input,
  - `muse_scipost_correct_dar`: DAR corrected pixel table,
  - `muse_scipost_calibrate_flux`: Flux calibrated pixel table,
  - `muse_scipost_apply_astrometry`: Pixel table with astrometric calibration

- **PIXTABLE_STD:**
  - `muse_scibasic`: Output pixel table for STD input
• **PIXTABLE\_SKY**:
  muse\_scibasic: Output pixel table for SKY input

• **PIXTABLE\_ASTROMETRY**:
  muse\_scibasic: Output pixel table for ASTROMETRY input

• **PIXTABLE\_REDUCED**:
  muse\_scibasic: Output pixel table,
  muse\_scipost: Fully reduced pixel tables for each exposure (if --save contains "individual"),
  muse\_scipost\_subtract\_sky: Output pixel table(s) for sky subtraction,
  muse\_scipost\_correct\_rv: RV corrected pixel table

• **PIXTABLE\_POSITIONED**:
  muse\_scipost: Fully reduced and positioned pixel table for each individual exposure (if --save contains "positioned")

• **PIXTABLE\_COMBINED**:
  muse\_scipost: Fully reduced and combined pixel table for the full set of exposures (if --save contains "combined"),
  muse\_exp\_combine: Combined pixel table (if --save contains "combined"),
  muse\_scipost\_combine\_pixtables: Combined pixel table

A.3.3 DATA CUBE

Description

Three FITS NAXIS=3 cubes in two extensions for data values and variance. A bad pixel is represented by a NAN value in the data and variance extensions. Such datacubes follow the ESO specification for FITS files with data, bad pixel maps, and variance. The units of the extensions are given by the standard BUNIT keyword.

They can have two-dimensional image extensions, of the same type as IMAGE\_FOV. For these, the EXTNAME will be called the same as the filter function that was used to create it. (Depending on recipe parameters, additional filtername\_STAT extensions may be present to represent the variance of the images. These images then follow the ESO specification.)

FITS extensions

• `\textquoteleft\textquoteleft DATA\textquoteright\textquoteright`: 3D FITS image (float)
  Data values

• `\textquoteleft\textquoteleft STAT\textquoteright\textquoteright`: 3D FITS image (float)
  Data variance

• 2D FITS image (float), optional, may appear more than once
  Data values of a filtered image

• 2D FITS image (float), optional, may appear more than once
  Data variance of a filtered image
Frame tags

- **GEOMETRY_CUBE**:
  - `muse_geometry`: Cube of the field of view to check the geometry calibration. It is restricted to the wavelength range given in the parameters and contains an integrated image ("white") over this range.

- **DATACUBE_STD**:
  - `muse_standard`: Reduced standard star field exposure

- **DATACUBE_Astrometry**:
  - `muse_astrometry`: Reduced astrometry field exposure

- **DATACUBE_FINAL**:
  - `muse_scipost`: Output datacube,
  - `muse_exp_combine`: Output datacube (if --save contains "cube"),
  - `muse_scipost_make_cube`: Output datacube

### A.3.4 EURO3DCUBE

**Description**

Euro3D format. See Format Definition Document, Kissler-Patig et al., Issue 1.2, May 2003, for a description.

Contrary to the examples in the Euro3D specs we use floats instead of doubles for the entries in the group table. This is because the E3D tool is otherwise not able to read the values correctly.

This data format may be written alternatively to the common DATACUBE format, if the parameter "format" is set to "Euro3D" or "xEuro3D".

**FITS extensions**

- **E3D_DATA**: FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEC_ID</td>
<td>int</td>
<td>Spectrum identifier</td>
</tr>
<tr>
<td>SELECTED</td>
<td>int</td>
<td>Selection flag</td>
</tr>
<tr>
<td>NSPAX</td>
<td>int</td>
<td>Number of instrument spaxels composing the spectrum</td>
</tr>
<tr>
<td>SPEC_LEN</td>
<td>int</td>
<td>Useful number of spectral elements [pixel]</td>
</tr>
<tr>
<td>SPEC_STA</td>
<td>int</td>
<td>Starting wavelength of spectrum [pixel]</td>
</tr>
<tr>
<td>XPOS</td>
<td>double</td>
<td>Horizontal position [pix]</td>
</tr>
<tr>
<td>YPOS</td>
<td>double</td>
<td>Vertical position [pix]</td>
</tr>
<tr>
<td>GROUP_N</td>
<td>int</td>
<td>Group number</td>
</tr>
<tr>
<td>SPAX_ID</td>
<td>string</td>
<td>Spaxel identifier</td>
</tr>
<tr>
<td>DATA_SPE</td>
<td>float array</td>
<td>Data a spectrum</td>
</tr>
<tr>
<td>QUAL_SPE</td>
<td>int array</td>
<td>Data quality spectrum</td>
</tr>
<tr>
<td>STAT_SPE</td>
<td>float array</td>
<td>Associated statistical error spectrum</td>
</tr>
</tbody>
</table>

- **E3D_GRP**: FITS table
<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP_N</td>
<td>int</td>
<td>Group number</td>
</tr>
<tr>
<td>G_SHAPE</td>
<td>string</td>
<td>Spaxel shape keyword</td>
</tr>
<tr>
<td>G_SIZE1</td>
<td>float</td>
<td>Horizontal size per spaxel [arcsec]</td>
</tr>
<tr>
<td>G_ANGLE</td>
<td>float</td>
<td>Angle of spaxel on the sky [deg]</td>
</tr>
<tr>
<td>G_SIZE2</td>
<td>float</td>
<td>Vertical size per spaxel [arcsec]</td>
</tr>
<tr>
<td>G_POSWAV</td>
<td>float</td>
<td>Wavelength for which the WCS is valid [Angstrom]</td>
</tr>
<tr>
<td>G_AIRMAS</td>
<td>float</td>
<td>Airmass</td>
</tr>
<tr>
<td>G_PARANG</td>
<td>float</td>
<td>Parallactic angle [deg]</td>
</tr>
<tr>
<td>G_PRESSU</td>
<td>float</td>
<td>Pressure [hPa]</td>
</tr>
<tr>
<td>G_TEMPER</td>
<td>float</td>
<td>Temperature [K]</td>
</tr>
<tr>
<td>G_HUMID</td>
<td>float</td>
<td>Humidity</td>
</tr>
</tbody>
</table>

Frame tags

- **DATACUBE_FINAL**:
  - `muse_scipost`: Output datacube,
  - `muse_exp_combine`: Output datacube (if `--save` contains "cube"),
  - `muse_scipost_make_cube`: Output datacube

A.3.5 TRACE_TABLE

Description

This file gives the trace solution for each slice in the form of a polynomial. It is a FITS table with 48 rows, one for each slice.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SliceNo</td>
<td>int</td>
<td>Slice number</td>
</tr>
<tr>
<td>Width</td>
<td>float</td>
<td>Average slice width</td>
</tr>
<tr>
<td>tc0_ij</td>
<td>double</td>
<td>polynomial coefficients for the central trace solution</td>
</tr>
<tr>
<td>MSE0</td>
<td>double</td>
<td>mean squared error of fit (central solution)</td>
</tr>
<tr>
<td>tc1_ij</td>
<td>double</td>
<td>polynomial coefficients for the left-edge trace solution</td>
</tr>
<tr>
<td>MSE1</td>
<td>double</td>
<td>mean squared error of fit (left-edge solution)</td>
</tr>
<tr>
<td>tc2_ij</td>
<td>double</td>
<td>polynomial coefficients for the right-edge trace solution</td>
</tr>
<tr>
<td>MSE2</td>
<td>double</td>
<td>mean squared error of fit (right-edge solution)</td>
</tr>
</tbody>
</table>
Frame tags

- TRACE_TABLE:
  - muse_flat: Trace table

A.3.6 TRACE_SAMPLES

Description

This is an optional FITS table, output on request by the muse_flat recipe. It can be used to verify the quality of the tracing, i.e. find out how accurate the pipeline was able to determine the location and boundary of the slices on the CCD.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice</td>
<td>int</td>
<td>Slice number</td>
</tr>
<tr>
<td>y</td>
<td>float</td>
<td>y position on the CCD [pix]</td>
</tr>
<tr>
<td>mid</td>
<td>float</td>
<td>Midpoint of the slice at this y position [pix]</td>
</tr>
<tr>
<td>left</td>
<td>float</td>
<td>Left edge of the slice at this y position [pix]</td>
</tr>
<tr>
<td>right</td>
<td>float</td>
<td>Right edge of the slice at this y position [pix]</td>
</tr>
</tbody>
</table>

Frame tags

- TRACE_SAMPLES:
  - muse_flat: Table containing all tracing sample points, if --samples=true

A.3.7 WAVECAL_TABLE

Description

This file gives the dispersion solution for each slice in one IFU. It is a FITS table with 48 rows, one for each slice.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SliceNo</td>
<td>int</td>
<td>Slice number</td>
</tr>
<tr>
<td>wlcIJ</td>
<td>double</td>
<td>Polynomial coefficients for the wavelength solution</td>
</tr>
<tr>
<td>MSE</td>
<td>double</td>
<td>Mean squared error of fit</td>
</tr>
</tbody>
</table>
Frame tags

- **WAVECAL_TABLE:**
  - `muse_wavecal`: Wavelength calibration table

### A.3.8 WAVECAL_RESIDUALS

**Description**

This is an optional FITS table, output on request by the `muse_wavecal` recipe. It can be used to verify the quality of the wavelength solution.

**FITS extensions**

- **FITS table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice</td>
<td>int</td>
<td>Slice number</td>
</tr>
<tr>
<td>iteration</td>
<td>int</td>
<td>Iteration</td>
</tr>
<tr>
<td>x</td>
<td>int</td>
<td>x position on the CCD [pix]</td>
</tr>
<tr>
<td>y</td>
<td>int</td>
<td>y position on the CCD [pix]</td>
</tr>
<tr>
<td>lambda</td>
<td>float</td>
<td>Wavelength [Angstrom]</td>
</tr>
<tr>
<td>residual</td>
<td>double</td>
<td>Residual at this point [Angstrom]</td>
</tr>
<tr>
<td>rejlimit</td>
<td>double</td>
<td>Rejection limit for this iteration [Angstrom]</td>
</tr>
</tbody>
</table>

Frame tags

- **WAVECAL_RESIDUALS:**
  - `muse_wavecal`: Fit residuals of all arc lines (if --residuals=true)

### A.3.9 LSF PROFILE

**Description**

This file contains the line spread function for all slices of one IFU.

It may come in two formats: one contains a datacube with a 2D image description of the LSF per slice; the other is a table with parameters of a Gauss-Hermite parametrization.

The pipeline automatically detects the format and continues processing accordingly.

**FITS extensions**

- **3D FITS image (float)**
  Data cube with the slice number (1...48) on the z axis, the line wavelength on the y axis, and the pixel wavelength on the x axis. The coordinate transformation between pixels and wavelength on x and y axes is done via the WCS header entries. The y wavelength range usually contains the full MUSE wavelength.
**FITS table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifu</td>
<td>int</td>
<td>IFU number</td>
</tr>
<tr>
<td>slice</td>
<td>int</td>
<td>Slice number within the IFU</td>
</tr>
<tr>
<td>sensitivity</td>
<td>double array</td>
<td>Detector sensitivity, relative to the reference</td>
</tr>
<tr>
<td>offset</td>
<td>double</td>
<td>Wavelength calibration offset</td>
</tr>
<tr>
<td>refraction</td>
<td>double</td>
<td>Relative refraction index</td>
</tr>
<tr>
<td>slit_width</td>
<td>double</td>
<td>Slit width</td>
</tr>
<tr>
<td>bin_width</td>
<td>double</td>
<td>Bin width</td>
</tr>
<tr>
<td>lsf_width</td>
<td>double array</td>
<td>LSF gauss-hermitean width</td>
</tr>
<tr>
<td>hermit3</td>
<td>double array</td>
<td>3th order hermitean coefficient</td>
</tr>
<tr>
<td>hermit4</td>
<td>double array</td>
<td>4th order hermitean coefficient</td>
</tr>
<tr>
<td>hermit5</td>
<td>double array</td>
<td>5th order hermitean coefficient</td>
</tr>
<tr>
<td>hermit6</td>
<td>double array</td>
<td>6th order hermitean coefficient</td>
</tr>
</tbody>
</table>

**Frame tags**

- **LSF_PROFILE**:
  - `muse_lsf`: Slice specific LSF images, stacked into one data cube per IFU.

**A.3.10 GEOMETRY_TABLE**

**Description**

This file provides the relative location of each slice in the MUSE field of view. It contains one table of $24 \times 48 = 1152$ rows, one for each slice.

Other columns (e.g. columns containing errors estimates of the slice properties, xerr, yerr, ...) may be present in this table but are ignored by the MUSE pipeline.

**FITS extensions**

- **FITS table**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubField</td>
<td>int</td>
<td>sub-field (IFU / channel) number</td>
</tr>
<tr>
<td>SliceCCD</td>
<td>int</td>
<td>Slice number on the CCD, counted from left to right</td>
</tr>
<tr>
<td>SliceSky</td>
<td>int</td>
<td>Slice number on the sky</td>
</tr>
<tr>
<td>x</td>
<td>double</td>
<td>x position within field of view [pix]</td>
</tr>
<tr>
<td>y</td>
<td>double</td>
<td>y position within field of view [pix]</td>
</tr>
<tr>
<td>angle</td>
<td>double</td>
<td>Rotation angle of slice [deg]</td>
</tr>
<tr>
<td>width</td>
<td>double</td>
<td>Width of slice within field of view [pix]</td>
</tr>
</tbody>
</table>

**Frame tags**

- **GEOMETRY_TABLE**
muse_geometry: Relative positions of the slices in the field of view

A.3.11 SPOTS_TABLE

Description

This file lists all detections and properties of all spots (the image of a pinhole at one arc line) during geometrical calibration. It is thought to be used for debugging of the geometrical calibration.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>string</td>
<td>(Raw) filename from which this measurement originates</td>
</tr>
<tr>
<td>image</td>
<td>int</td>
<td>Number of the image in the series</td>
</tr>
<tr>
<td>PosenC2</td>
<td>int</td>
<td>X position of the mask in encoder steps</td>
</tr>
<tr>
<td>Pospos2</td>
<td>double</td>
<td>X position of the mask [mm]</td>
</tr>
<tr>
<td>PosenC3</td>
<td>int</td>
<td>Y position of the mask in encoder steps</td>
</tr>
<tr>
<td>Pospos3</td>
<td>double</td>
<td>Y position of the mask [mm]</td>
</tr>
<tr>
<td>PosenC4</td>
<td>int</td>
<td>Z position of the mask in encoder steps</td>
</tr>
<tr>
<td>Pospos4</td>
<td>double</td>
<td>Z position of the mask [mm]</td>
</tr>
<tr>
<td>Vpos</td>
<td>double</td>
<td>Real vertical position of the mask [mm]</td>
</tr>
<tr>
<td>ScaleFOV</td>
<td>double</td>
<td>Focus scale in VLT focal plane (from the FITS header) [arcsec/mm]</td>
</tr>
<tr>
<td>SubField</td>
<td>int</td>
<td>Sub-field number</td>
</tr>
<tr>
<td>SliceCCD</td>
<td>int</td>
<td>Slice number as counted on the CCD</td>
</tr>
<tr>
<td>Lambda</td>
<td>double</td>
<td>Wavelength [Å]</td>
</tr>
<tr>
<td>SpotNo</td>
<td>int</td>
<td>Number of this spot within the slice (1 is left, 2 is the central one, 3 is right within the slice)</td>
</tr>
<tr>
<td>Xc</td>
<td>double</td>
<td>x center of this spot on the CCD [pix]</td>
</tr>
<tr>
<td>Yc</td>
<td>double</td>
<td>y center of this spot on the CCD [pix]</td>
</tr>
<tr>
<td>XFWHM</td>
<td>double</td>
<td>FWHM in x-direction on the CCD [pix]</td>
</tr>
<tr>
<td>YFWHM</td>
<td>double</td>
<td>FWHM in y-direction on the CCD [pix]</td>
</tr>
<tr>
<td>Flux</td>
<td>double</td>
<td>Flux of the spot as integrated on the CCD image</td>
</tr>
<tr>
<td>BG</td>
<td>double</td>
<td>Background level around the spot</td>
</tr>
<tr>
<td>Dxcen</td>
<td>double</td>
<td>distance to center of slice at vertical position yc (positive: right of center) [pix]</td>
</tr>
<tr>
<td>Twidth</td>
<td>double</td>
<td>trace width of the slice at the vertical CCD position of the spot [pix]</td>
</tr>
</tbody>
</table>

Frame tags

- SPOTS_TABLE:
  muse_geometry: Measurements of all detected spots on all input images.
A.3.12 FLUX_TABLE

Description

This is a simple binary FITS table with the dependency of the flux on wavelength.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>Wavelength [Å]</td>
</tr>
<tr>
<td>flux</td>
<td>double</td>
<td>Flux [erg/(s cm² arcsec²)]</td>
</tr>
<tr>
<td>fluxerr</td>
<td>double</td>
<td>Error of the flux (optional column) [erg/(s cm² arcsec²)]</td>
</tr>
</tbody>
</table>

Frame tags

- **SKY_SPECTRUM**:  
  - *muse_create_sky*: Sky spectrum within the sky mask,  
  - *muse_scipost*: Sky spectrum within the sky mask (if --skymethod=model and --save contains "skymodel")

- **SKY_CONTINUUM**:  
  - *muse_create_sky*: Estimated continuum flux spectrum,  
  - *muse_scipost*: Estimated continuum flux spectrum (if --skymethod=model and --save contains "skymodel")

A.3.13 STD_RESPONSE

Description

MUSE flux response table.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>wavelength [Å]</td>
</tr>
<tr>
<td>response</td>
<td>double</td>
<td>instrument response derived from standard star [2.5 log10((count/s/Å) / (erg/s/cm²*2/Å))]</td>
</tr>
<tr>
<td>resperr</td>
<td>double</td>
<td>instrument response error derived from standard star [2.5 log10((count/s/Å) / (erg/s/cm²*2/Å))]</td>
</tr>
</tbody>
</table>
Frame tags

- **STD_RESPONSE**: 
  muse_standard: Response curve as derived from standard star(s)

A.3.14 STD_TELLURIC

Description

MUSE telluric correction table.

FITS extensions

- FITS table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>double</td>
<td>wavelength [Angstrom]</td>
</tr>
<tr>
<td>ftelluric</td>
<td>double</td>
<td>the telluric correction factor, normalized to an air-mass of 1</td>
</tr>
<tr>
<td>ftellerr</td>
<td>double</td>
<td>the error of the telluric correction factor</td>
</tr>
</tbody>
</table>

Frame tags

- **STD_TELLURIC**: 
  muse_standard: Telluric absorption as derived from standard star(s)

A.3.15 STD_FLUXES

Description

2D Image containing measurements of flux integration of all stars detected in a standard star field. This is mainly thought to be used for debugging. The image contains a spectral axis (axis 1) with corresponding WCS information. Axis 2 is the arbitrary numbering of stars detected in the field. Several ESO.DRS.MUSE.* keywords in the output header contain information regarding each source (x and y position [pix] in the corresponding data cube, approximate celestial position [deg], and integrated flux); their numbering corresponds to the axis 2 coordinate. The unit of the integrated flux is given by the standard BUNIT keyword.

FITS extensions

- **'DATA'**: 2D FITS image (float) 
  Integrated fluxes per wavelength bin

- **'DQ'**: 2D FITS image (int), optional 
  Corresponding Euro3D data quality per wavelength bin

- **'STAT'**: 2D FITS image (float) 
  Corresponding data variance per wavelength bin
Frame tags

- STD_FLUXES:
  - muse_standard: The integrated flux per wavelength of all detected sources

A.3.16 AMPL_CONVOLVED

Description

This FITS image contains two extensions, PHOTONS and ENERGY, showing filter-convolved values of the convolved flat-fields.

FITS extensions

- 'PHOTONS': 2D FITS image (float)
  - Photon counts [ph]
- 'ENERGY': 2D FITS image (float)
  - Per-pixel energy [J]

Frame tags

- AMPL_CONVOLVED:
  - muse;amp: Combined and convolved master AMPL image

A.3.17 OFFSET_LIST

Description

Coordinate offsets suitable for being used with muse_exp_combine to properly align a set exposures to a reference position during the creation of a combined data cube.

The offset corrections in the RA_OFFSET and DEC_OFFSET columns are the direct difference of the measured position to the reference position, without cos(DEC):

RA_OFFSET = RA(measured) - RA(reference)
DEC_OFFSET = DEC(measured) - DEC(reference)

This table optionally also contains a FLUX_SCALE column that is then used to correct relative scaling of exposures in a sequence, e.g. to correct observations taken in non-photometric conditions. When created by muse_exp_align, the table does contain the FLUX_SCALE column. It is then filled with invalid values (NANs), so that the pipeline knows to ignore these values. The user can fill these values by hand with any FITS editor.

FITS extensions

- FITS table
<table>
<thead>
<tr>
<th>Column name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE_OBS</td>
<td>string</td>
<td>Date and time at the start of the exposure</td>
</tr>
<tr>
<td>MJD_OBS</td>
<td>double</td>
<td>MJD at the start of the exposure (optional column)</td>
</tr>
<tr>
<td>RA_OFFSET</td>
<td>double</td>
<td>Right ascension offset in degrees [deg]</td>
</tr>
<tr>
<td>DEC_OFFSET</td>
<td>double</td>
<td>Declination offset in degrees [deg]</td>
</tr>
<tr>
<td>FLUX_SCALE</td>
<td>double</td>
<td>(Relative) flux scaling of the given exposure (optional column)</td>
</tr>
</tbody>
</table>

**Frame tags**

- **OFFSET_LIST**
  - **muse_exp_align**: List of computed coordinate offsets.

**A.4 Other data files**

**A.4.1 OUTPUT_WCS**

**Description**

Normally, the MUSE pipeline automatically adapts the output cube dimensions and sky location depending on the data. This type of file can be used to override this automatism. The first valid FITS extension with a FITS WCS (so with either NAXIS or WCSAXES) is used set override parameters for the output cube.

There are, however, several restrictions:

* The axes have to be in the order RA (1), DEC (2), wavelength (3)
* Only support gnomonic projection spatially (TAN), and linear or log air wavelength sampling are supported.
* A tilted 3rd axis is rejected.
* Only the primary WCS description is evaluated.
* The WCS transformation matrix has to be in CDi_j form (PCI_j is not accepted).
* Floating point WCS parameters without dots in the FITS are not recognized.

Output cubes (FITS NAXIS=3) written by the MUSE pipeline can be used as OUTPUT_WCS.