EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

MUSE Data Reduction Cookbook

VLT-MAN-ESO-14670-6186

Issue 4

Date 2014-11-21

Prepared:	MUSE Pipeline Team	2014-07-08	
	Name	Date	Signature
Approved:	J.Vernet, R.Bacon		
	Name	Date	Signature
Released:	J.Vernet		
	Name	Date	Signature

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1	2014-05-30	All	First version
2	2014-07-01	1.6, 3.3	References to *Reflex* documentation added
		7.3	New pixel table handling tool added
3	2014-07-08	4, 8.2	Added failed tracing recovery procedure for SV data
4	2014-11-21	6.6.3, 6.6.4	Correct typing error in environment variable name

This page was intentionally left blank

Contents

1 Introduction **11**

1.1 Scope 11

1.2 Acknowledgements 11

1.3 Stylistic Conventions 11

1.4 Notational Conventions 11

1.5 Reference Documents 12

1.6 Abbreviations and Acronyms 13

2 Overview **14**

2.1 The MUSE Instrument 14

2.2 The MUSE Data Reduction Pipeline 14

3 Installation **18**

3.1 System Requirements – Please Read Carefully! 18

3.2 Installing the Software 19

3.3 Toolchain Support 19

3.4 Hints on Running the MUSE Pipeline Recipes 20

3.5 Hints on Using 3rd-Party Tools 20

4 Known Issues **22**

5 Data Description **23**

5.1 Raw Data 23

5.2 Static Calibration Data 23

5.3 Pipeline Products 25

5.3.1 MUSE Images 25

5.3.2 MUSE Pixel Tables 25

5.3.3 MUSE Data Cubes 25

5.3.4 Combined Product Data 26

6 Data Reduction Cookbook **27**

6.1 Getting Started with EsoRex . . . 27
6.2 Data Organization . . . 28
6.2.1 Useful Header Keywords . . . 28
6.2.2 Data Classification and Association . . . 29
6.3 Basic Reduction . . . 31
6.3.1 Bias . . . 31
6.3.2 Dark . . . 32
6.3.3 Flat Field . . . 34
6.3.4 Wavelength Calibration . . . 34
6.3.5 Skyflat . . . 36
6.3.6 Instrument Geometry . . . 38
6.4 Observation Pre-processing . . . 38
6.5 Observation Post-Processing . . . 39
6.5.1 Flux Calibration . . . 39
6.5.2 Sky Creation . . . 41
6.5.3 Astrometry . . . 42
6.5.4 Science Observations . . . 43
6.6 Combining Exposures . . . 44
6.6.1 Correcting Coordinate Offsets . . . 44
6.6.2 Limiting Wavelength Ranges . . . 45
6.6.3 Combining Exposures using muse_scipost . . . 45
6.6.4 Combining Exposures using muse_exp_combine . . . 46

7 Tips & Tricks **49**
7.1 Restricting wavelength ranges . . . 49
7.2 Verification Tools . . . 49
7.2.1 Verification of the tracing solution . . . 49
7.2.2 Verification of the wavelength solution . . . 51
7.3 Miscellaneous Tools . . . 53
7.3.1 Handling of MUSE pixel tables . . . 53
7.3.2 Handling of MUSE bad pixel maps . . . 54

8 Troubleshooting **56**

8.1 Typical Problems 56

8.2 Failed Tracing 56

8.3 The Logfile 57

8.4 Debugging Options 58

9 Recipe Reference **59**

9.1 muse_bias 59

9.1.1 Description 59

9.1.2 Input frames 59

9.1.3 Recipe parameters 59

9.1.4 Product frames 60

9.1.5 Quality control parameters 60

9.2 muse_dark 62

9.2.1 Description 62

9.2.2 Input frames 62

9.2.3 Recipe parameters 62

9.2.4 Product frames 63

9.2.5 Quality control parameters 63

9.3 muse_flat 65

9.3.1 Description 65

9.3.2 Input frames 65

9.3.3 Recipe parameters 65

9.3.4 Product frames 67

9.3.5 Quality control parameters 67

9.4 muse_wavecal 69

9.4.1 Description 69

9.4.2 Input frames 69

9.4.3 Recipe parameters 69

9.4.4 Product frames 71

9.4.5 Quality control parameters 72

9.5 muse_skyflat . . . 73
9.5.1 Description . . . 73
9.5.2 Input frames . . . 73
9.5.3 Recipe parameters . . . 73
9.5.4 Product frames . . . 74
9.5.5 Quality control parameters . . . 75
9.6 muse_geometry . . . 76
9.6.1 Description . . . 76
9.6.2 Input frames . . . 76
9.6.3 Recipe parameters . . . 76
9.6.4 Product frames . . . 77
9.6.5 Quality control parameters . . . 77
9.7 muse_scibasic . . . 79
9.7.1 Description . . . 79
9.7.2 Input frames . . . 79
9.7.3 Recipe parameters . . . 79
9.7.4 Product frames . . . 81
9.7.5 Quality control parameters . . . 82
9.8 muse_standard . . . 83
9.8.1 Description . . . 83
9.8.2 Input frames . . . 83
9.8.3 Recipe parameters . . . 83
9.8.4 Product frames . . . 84
9.8.5 Quality control parameters . . . 84
9.9 muse_create_sky . . . 85
9.9.1 Description . . . 85
9.9.2 Input frames . . . 85
9.9.3 Recipe parameters . . . 85
9.9.4 Product frames . . . 86
9.9.5 Quality control parameters . . . 86

9.10 muse_astrometry . . . 87
9.10.1 Description . . . 87
9.10.2 Input frames . . . 87
9.10.3 Recipe parameters . . . 87
9.10.4 Product frames . . . 88
9.10.5 Quality control parameters . . . 88
9.11 muse_scipost . . . 89
9.11.1 Description . . . 89
9.11.2 Input frames . . . 89
9.11.3 Recipe parameters . . . 89
9.11.4 Product frames . . . 92
9.11.5 Quality control parameters . . . 92
9.12 muse_exp_combine . . . 94
9.12.1 Description . . . 94
9.12.2 Input frames . . . 94
9.12.3 Recipe parameters . . . 94
9.12.4 Product frames . . . 95
9.12.5 Quality control parameters . . . 96

A Data Formats **97**
A.1 Raw Data Files . . . 97
A.1.1 RAW_IMAGE . . . 97
A.2 Static Calibration Files . . . 99
A.2.1 LINE_CATALOG . . . 99
A.2.2 SKY_LINES . . . 99
A.2.3 ASTROMETRY_REFERENCE . . . 100
A.2.4 EXTINCT_TABLE . . . 101
A.2.5 BADPIX_TABLE . . . 101
A.2.6 STD_FLUX_TABLE . . . 102
A.2.7 FILTER_LIST . . . 102
A.2.8 TELLURIC_REGIONS . . . 103

A.3 Recipe Product Files . . . 104
A.3.1 MUSE_IMAGE . . . 104
A.3.2 PIXEL_TABLE . . . 105
A.3.3 DATACUBE . . . 107
A.3.4 EURO3DCUBE . . . 108
A.3.5 TRACE_TABLE . . . 109
A.3.6 TRACE_SAMPLES . . . 110
A.3.7 WAVECAL_TABLE . . . 110
A.3.8 WAVECAL_RESIDUALS . . . 111
A.3.9 LSF_TABLE . . . 111
A.3.10 GEOMETRY_TABLE . . . 112
A.3.11 SPOTS_TABLE . . . 113
A.3.12 FLUX_TABLE . . . 114
A.3.13 STD_RESPONSE . . . 115
A.3.14 STD_TELLURIC . . . 115

B Benchmarks **117**
B.1 The Reference System . . . 117
B.2 Benchmark results . . . 117

C Performance Tools **118**
C.1 Using `taskset` . . . 118
C.2 Using the Likwid Lightweight Performance Tools . . . 118

D Useful links **120**

1 Introduction

1.1 Scope

This document is a quick start guide to the processing of MUSE observations using the MUSE Instrument Pipeline Recipes (also known as the MUSE Data Reduction Software or MUSE DRS). While it is not as detailed as the full MUSE Pipeline Manual when it comes to how individual processing recipes work, it should guide the users through the installation of the software, and bring them quickly to the point where they can use the software tools to create science ready datacubes from raw MUSE data sets.

The processing of MUSE data is extremly demanding with respect to the computing environment. Therefore, the document describes the system requirements, the installation and the setup of the environment in some detail in order to avoid the most common pitfalls, and it is strongly recommended that the users reads these sections before they start working on MUSE data.

This manual refers to the MUSE Data Reduction Software version 0.18.5. This is a pre-release version made available to support the MUSE Science Verification. This current release has a few known limitations, which are listed in Section 4.

1.2 Acknowledgements

This cookbook is to a large extent based on the MUSE Data Reduction Software Manual, prepared by T. Urrutia, O. Streicher and P. Weilbacher, for use within the MUSE consortium. We also would like to thank J. Richard, P. Weilbacher and B. Husemann for providing valuable comments.

1.3 Stylistic Conventions

Throughout this document the following stylistic conventions are used:

bold	in text sections for commands and other user input which has to be typed as shown
italics	in the text and example sections for parts of the user input which have to be replaced with real contents
`teletype`	in the text for FITS keywords, program names, file paths, and terminal output, and as the general style for examples, commands, code, etc

In example sections expected user input is indicated by a leading shell prompt.

In the text **bold** and *italics* may also be used to highlight words.

1.4 Notational Conventions

Hierarchical FITS keyword names, appearing in the document, are given using the dot–notation to improve readability. This means, that the prefix “HIERARCH ESO” is left out, and the spaces separating the keyword name constituents in the actual FITS header are replaced by a single dot.

1.5 Reference Documents

[RD1]	MUSE Pipeline Manual	TBD
[RD2]	MUSE User Manual	VLT–MAN–ESO–14670–1477
[RD3]	MUSE Calibration Plan	VLT–MAN–ESO–14670–0500
[RD4]	Gasgano User's Manual	VLT–PRO–ESO-19000–1932
[RD5]	FITS format description for pipeline products with data, error and data quality information	VLT–MAN–ESO–19500–5667
[RD6]	The Euro3D Data Format, Kissler-Patig, et al., Issue 1.2, May 2003	
[RD7]	3D Visualization Tool Manual	VLT–MAN–ESO–19500–5651
[RD8]	Reflex MUSE Tutorial	VLT–MAN–ESO–19540–6195
[RD9]	Reflex User Manual	VLT–MAN–ESO–19000–5037

1.6 Abbreviations and Acronyms

AO	Adaptive Optics
CCD	Charge Coupled Device
CPL	Common Pipeline Library
CPU	Central Processing Unit
DFS	Data Flow System
DRS	Data Reduction System
ESO	European Southern Observatory
EsoRex	ESO Recipe Execution Tool
FITS	Flexible Image Transport System
FOV	Field of View
GCC	GNU Compiler Collection
GUI	Graphical User Interface
HDU	Header Data Unit
IFU	Integral Field Unit
MUSE	Multi Unit Spectroscopic Explorer
NFM	Narrow Field Mode
OpenMP	Open Multi-Processing
PAF	VLT parameter file format
pixel	picture element (of a raster image)
PSF	Point Spread Function
QC	Quality Control
SGS	Slow Guilding System
SOF	Set Of Frames
SV	Science Verification
spaxel	spatial element (of a data cube)
TBC	To be confirmed
TBD	To be defined
VLT	Very Large Telescope
voxel	volume element (of a data cube)
WCS	World Coordinate System
WFM	Wide Field Mode

2 Overview

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 $1' \times 1'$ in wide-field mode (WFM) with a sampling of $0.2'' \times 0.2''$ spaxels. The full field is split up into 24 sub-fields (each $2.5'' \times 60''$ in WFM) which are fed into one of 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. This is illustrated in Figure 2.1.

On the CCD, the slices appear as approximately 76.5 pixel wide vertical bands, separated by gaps of about 6 pixel. At the edges of the detector the slices are slightly curved outwards (up to 2 pixel). The slices are offset vertically, i.e. along the wavelength axis, forming a repetitive, horizontal pattern of three steps. This pattern is affected by curvature across the CCD, which results in a different wavelength coverage for each slice.

For performance reasons the MUSE detectors are always read in 4-port mode. All four quadrants have equal size, and are visible in the raw data images separated horizontally and vertically by pre-scan and over-scan regions. On the detectors, and the raw images the dispersion axis is oriented along the pixel columns (vertical axis) with the red end of the spectrum at the top, and the blue end at the bottom. The pixel rows (horizontal axis) correspond to the spatial axis.

2.2 The MUSE Data Reduction Pipeline

The MUSE pipeline is basically divided into two stages. The first stage consists of the six basic calibration recipes and a preprocessing recipe (basic science reduction) which work on the data of individual CCDs to determine and/or remove the signature of each IFU. The recipes of the second stage, another three calibration recipes and the final science recipe, use the pre-processed data from the first stage and transform it into physical quantities which can be used for science. These second stage recipes combine the data from all IFUs of one or more exposures into the final data cube. The two stages of the reduction process are illustrated in Figure 2.2 and Figure 2.3.

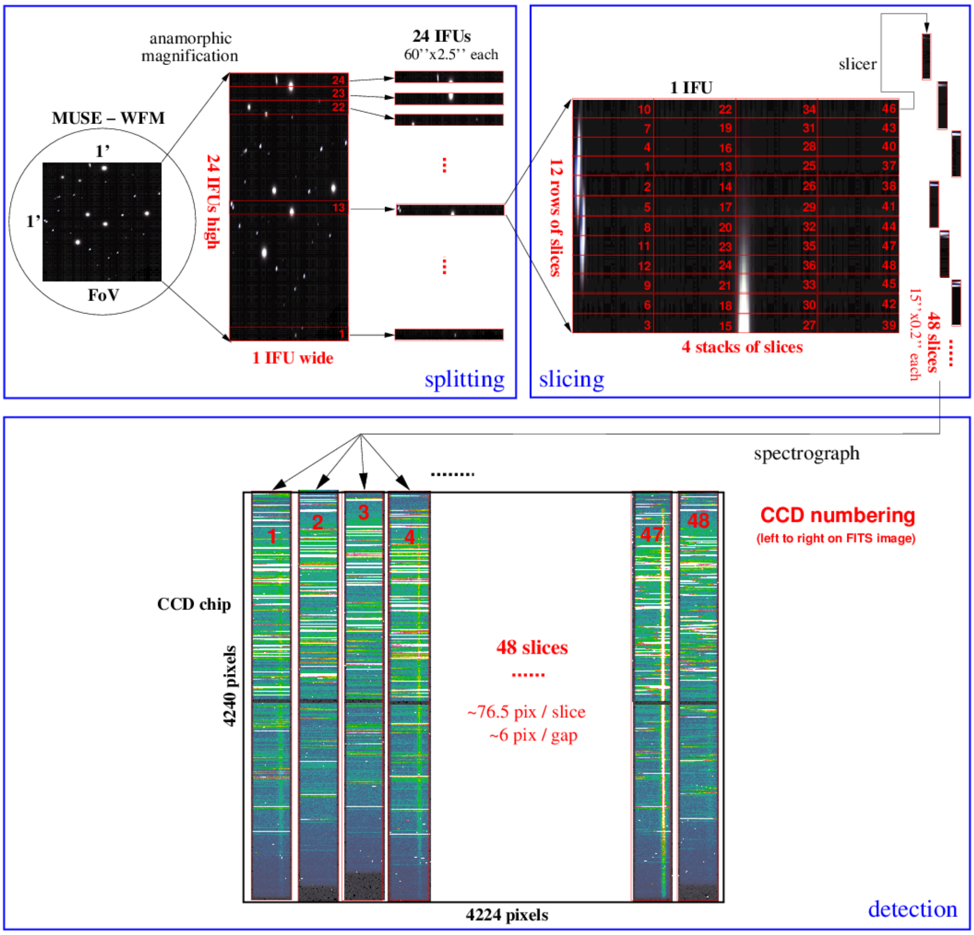

Figure 2.1: Graphical representations of the splitting and slicing procedures in the MUSE instrument. The example shown is for wide-field mode; narrow-field mode operates in the same way with a scaled-down field size. Note that the sizes given are approximate, the real data does not exactly cover a square region on the sky.

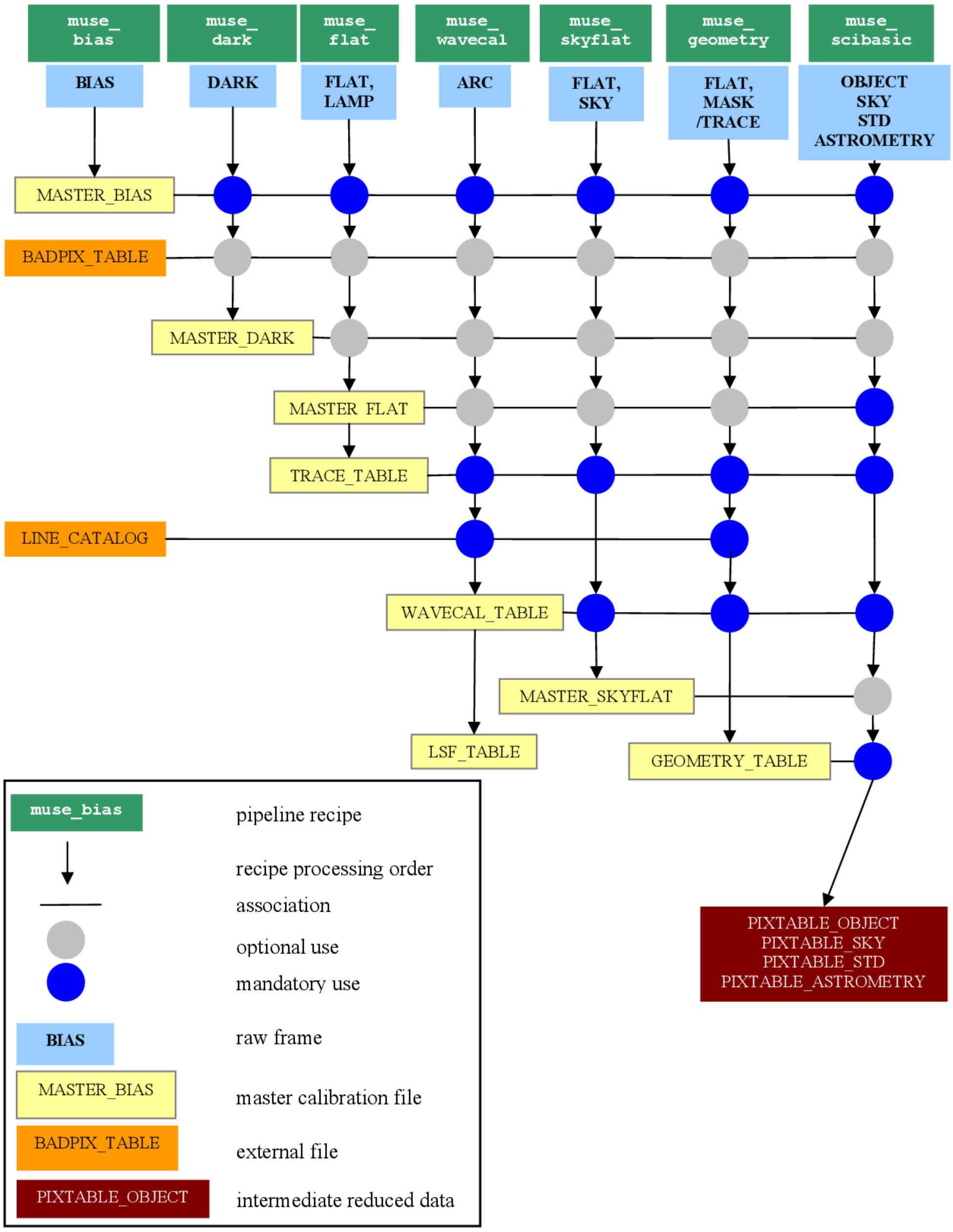

Figure 2.2: The first stage of the MUSE data reduction cascade. It shows the basic calibration recipes and the pre-processing recipe `muse_scibasic`, together with the "Association map" indicating the required and the optional input for each of the recipes.

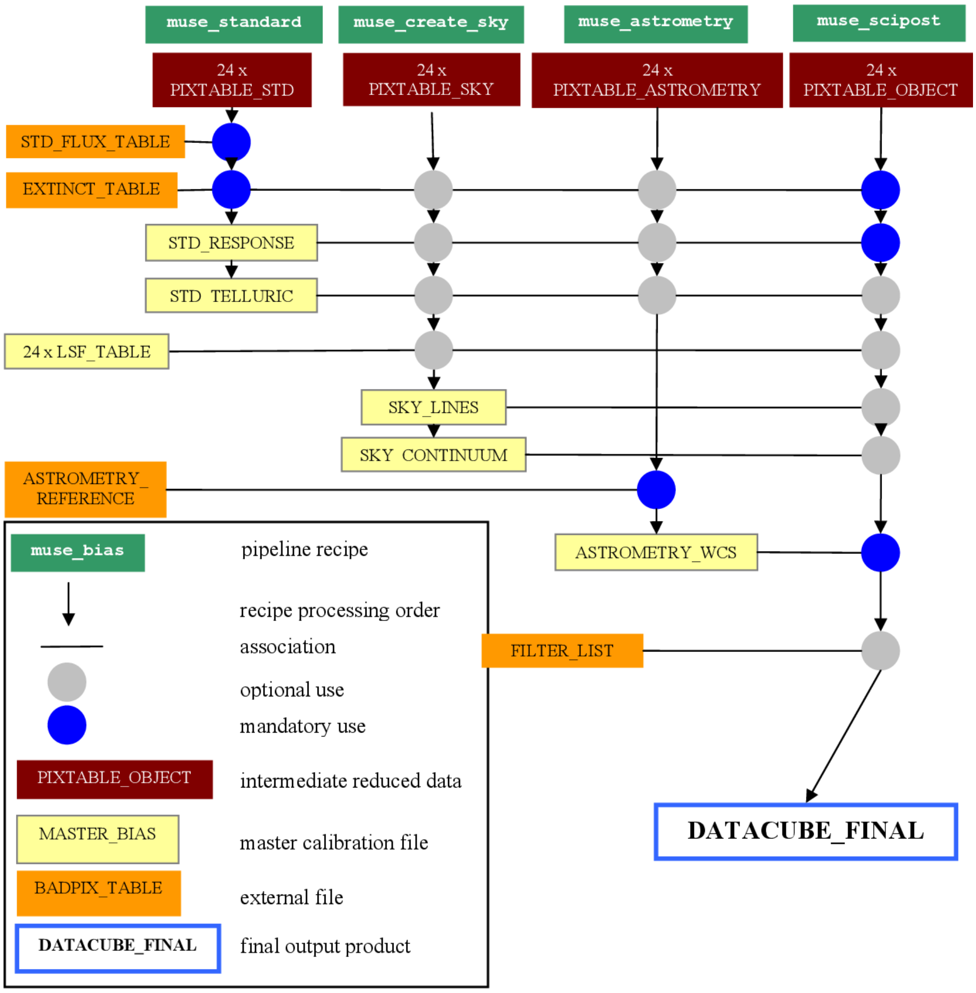

Figure 2.3: The second stage of the MUSE data reduction cascade. These recipes start from the output of the pre-processing recipe `muse_scibasic` and combine the data from all 24 IFUs into a single, fully reduced and calibrated data cube. Again the "Association map" indicates the required and the optional input for each of the recipes.

3 Installation

3.1 System Requirements – Please Read Carefully!

The processing of MUSE data is very demanding in terms of computing resources. In particular it requires a machine with sufficient memory installed. Less critical but still important is the number of available CPU cores and the amount of available disk space.

Because of the memory constraints, the MUSE DRS is **only** supported on 64-bit platforms. The recommended platform is a powerful workstation with a recent 64-bit Linux system.

The minimum system configuration is:

- 32 GB of memory
- 4 CPU cores (physical cores)
- 1 TB of free disk space
- GCC 4.4.6 (or newer)

The recommended configuration of the target machine for creating the final data cube from a single MUSE observation and the required set of calibrations is:

- 64 GB of memory
- 24 CPU cores (physical cores)
- 4 TB of free disk space
- GCC 4.8.2 (or newer)

The peak memory consumption depends on the number of input exposures used and the size of the final data cube. When a data cube is created from a single MUSE exposure the peak memory usage may vary between approximately 18 GB and 25 GB depending on the orientation of the field of view. Peak memory consumption is reached at a rotation angle of $\pm 45°$ and $\pm 135°$ with respect to North. On disk the size of the final data cube file may vary between 4 GB and 5.3 GB for a single observation.

If it is forseen to combine several MUSE observations, the memory needed depends on the maximum number of observations to combine. In general the memory consumption grows linearly with the number of observations.

Finally, when it comes to creating MUSE mosaics, one should be aware that the size of the data cube may become really huge, and the required memory grows accordingly! For details on combining MUSE exposures refer to Section 6.6.

The same memory requirements apply to the MUSE calibration recipes. The minimum memory estimate of 32 GB given before is suitable for processing standard calibration sets of 5 exposures. Adding exposures to the input of the recipes requires additional memory.

However, since the calibration recipes can work on a single IFU at a time, the memory requirements can be relaxed for these recipes at the price of reduced performance. For the creation of the final data cube the only possibility to reduce the required memory is to limit the wavelength range of the output data cube to an appropriate size (cf. Section 7.1).

3.2 Installing the Software

The MUSE Data Reduction Software is distributed as a standard pipeline kit package and can be obtained from the ESO web pages at http://www.eso.org/sci/software/pipelines. Apart from the MUSE DRS itself, the distributed package contains all dependencies needed for the installation, the tools to run the MUSE DRS recipes, this cookbook, and the installer utility for the kit.

Using the installer the MUSE DRS can be installed on recent versions of any major Linux distribution, as well as Mac OS X. However, the recommended target platform for using the MUSE DRS is a 64-bit Linux system, since Mac OS X imposes certain restrictions when it comes to running the MUSE DRS.

To install the MUSE DRS unpack the kit in a temporary location, go to the top level directory of the unpacked distribution package and execute the installer script as shown in the following example.

Note: The installation script uses the compiler which is found first in the path! If more than one compiler are installed on the system one should make sure that an appropriate 64-bit compiler will be found first when the installation script is executed!

```
1> tar -zxf muse-kit-X.Y.Z.tar.gz
2> cd muse-kit-X.Y.Z.tar.gz
3> ./install_pipeline
```

Then follow the instructions on the screen. Once the script finishes successfully and the path variables have been set, the installation of the MUSE DRS is complete.

3.3 Toolchain Support

ESO offers three different tools to process data obtained with one of the VLT instruments. One command line tool *EsoRex*, and two GUI based tools, *Gasgano* and *Reflex* respectively.

This manual will focus on the *EsoRex* command line tool, which offers at the moment the best control on the reduction process.

As an alternative one may consider using *Reflex* and the MUSE workflow to process MUSE observations. The latter is a convenient way to execute a fixed setup of the reduction chain as it includes automatic data organization. A first version of a *Reflex* workflow for MUSE is already available. See [RD8] and [RD9] for details on how to use this MUSE *Reflex* workflow and on *Reflex* itself.

While *Gasgano* is useful as file browser for exploring MUSE data sets, it is **not** recommended to use it to run the MUSE DRS recipes. Using *Gasgano* to run MUSE recipes will therefore not be discussed further in this manual.

3.4 Hints on Running the MUSE Pipeline Recipes

In order to have a good performance when processing MUSE data, the MUSE DRS recipes are multi-threaded using the OpenMP thread model, and thus may use all the CPU cores that are available to speed up the processing.

While this mode has to be explicitly enabled using a recipe parameter for the MUSE calibration recipes and the preprocessing recipe, this is always used in the second stage recipes of the MUSE DRS. It is assumed that this is the standard way to run the MUSE DRS.

However, it is not always the best solution to simply try to use all available CPUs on the machine. For instance, reducing the number of allowed threads can be used to reduce the memory consumption of the first stage recipes (of course this increases the recipes execution time). The maximum number of threads the MUSE DRS recipes (actually any application using OpenMP) may can be set using the environment variable `OMP_NUM_THREADS`[1].

In addition, modern computer architectures have certain features which may make it necessary to control the way high performance applications are executed on a particular machine, in order to get to a good performance. This applies also to the MUSE DRS.

What may be necessary is to make sure that the individual threads the application runs stick to the CPU where they were started in first place. There are several so called thread pinning tools available which can be used for this. For example Linux distributions come with the `taskset` utility, or, one can use the *Likwid* Lightweight Performance Tools which are available at https://code.google.com/p/likwid (**Linux only!**).

Neither setting the environment variable `OMP_NUM_THREADS` nor using any thread-pinning tool is necessary as long as one gets a good performance. To allow for a comparison of the performance, the benchmarks for the ESO baseline system are given in Appendix B.

The standard setup at ESO are 24 threads (i.e. 1 thread per IFU, `OMP_NUM_THREADS=24`) with the threads pinned to the machines physical CPU cores for all recipes (with the exception of the geometric calibration recipe which can be run only with a reduced number of threads). A short overview on how to use the thread pinning tools can be found in Appendix C.

A Note for Mac OS X Users

On Mac OS X the MUSE DRS is always built without support for running multi-threaded. This is due to the lack of essential features in the OpenMP implementation of Mac OS X. As a consequence on Mac OS X the MUSE DRS will always run in single-threaded mode. Turning on the multi-threaded mode for the calibration recipes and the preprocessing recipe has no effect, and the data would be processed sequentially, one IFU after the other.

3.5 Hints on Using 3rd-Party Tools

Any 3rd-party tool which is used to visualize, or inspect the product files created by the MUSE DRS should be 64-bit applications, since the products from the MUSE DRS can be larger than 2 GB. Files larger than 2 GB

[1] Actually for some implementations the default maximum number of threads is 1 or 2. In this case `OMP_NUM_THREADS` must be used to enlarge the number of allowed threads.

cannot be handled by 32-bit applications, unless they have been specifically build for that.

This applies in particular to the visualization tool used to inspect the final data cube product with file sizes of 4 GB or more.

4 Known Issues

The MUSE DRS is still being worked on, taking into account the results of the commissioning runs preceding the MUSE Science Verification. As a result of the commissioning runs a number of issues and areas of improvement have been identified.

Users processing MUSE data with the MUSE DRS version 0.18.5 should be aware of the following limitations which are present in this current pre-release version of the MUSE Pipeline:

- The recipe **muse_skyflat** and its product `MASTER_SKYFLAT` will likely be replaced by an illumination correction.
- The recipe **muse_astrometry** may fail to identify (enough) stars and either fail completely or give a wrong result (the computed pixel scales may be far from the expected result of $0.2''/\mathrm{spaxel}$)
- The recipe **muse_standard** does not create smooth results (not smooth enough), the flux calibration may be wrong for individual wavelengths; the correction of the telluric absorption is not optimal and may give bad results in the relevant wavelength bands.
- Handling of the line-spread function will likely be replaced by a more robust method (and file format).
- The cosmic ray cleaning is not optimal and needs extra tweaking for each cube.

All these issues will be addressed in the forthcoming releases of the MUSE DRS. For the second and third issue a workaround is available (the necessary master calibrations are distributes as part of the MUSE Pipeline package).

Another issue, specifically affects data taken during the first MUSE Science Verification run. For flat fields, which are taken at low ambient temperature (temperatures lower than $\approx 7\,^{\circ}\mathrm{C}$), the tracing of the edges of the slices may fail. Data taken after this first SV run should not be affected since the issue will be fixed at the instrument level.
In order to be able to complete the processing of the SV data if such a failure happens if **muse_flat**, a recovery procedure is described in Section 8.2.

5 Data Description

5.1 Raw Data

The MUSE raw data of all 24 IFUs is stored in a single FITS file as individual FITS extensions. When MUSE raw data is retrieved from the ESO archive they will have the standard ESO archive file names which are made up of instrument identifier followed by a time stamp. The time stamp corresponds to the contents of the FITS header keywords `MJD-OBS` and `DATE-OBS` respectively, i.e. to the date and time when the exposure has been taken (a difference of 1 ms between the file name and the contents of the keywords may be present). In the case of MUSE, the files returned by the archive are **tile compressed**, which is indicated by the file name suffix `.fits.fz` instead of the regular `.fits` suffix, so that the file name of a MUSE raw data file will look like

```
MUSE.2014-02-20T23:31:38.542.fits.fz
```

These tile compressed files may be unpacked using the `funpack` tool which is distributed as part of the CFITSIO package (see Section D for the web site). If the MUSE DRS recipes are executed directly from the command line using *EsoRex* there is no need to uncompress the MUSE raw data files, since the MUSE DRS will do this on the fly. This is however not true for *Reflex* which does **not** accept tile compressed files.

The MUSE raw data files are composed of 24 FITS extensions, one for each IFU, which contain the detector data and the IFU/detector specific keywords, and a primary FITS HDU which contains only keywords. The keywords in this primary HDU contain almost all the information which is needed to correctly identify and process the individual files. In addition, science exposures will contain an extra three FITS extensions which contain information from the MUSE Slow Guiding System (SGS), if the SGS is activated (see Section A for details).

The 24 extensions containing the detector data of the IFUs can be identified using the contents of the `EXTNAME` FITS keyword. The extensions are called `CHAN01`[2], `CHAN02`, etc. For technical reasons, the numbering of the FITS extensions does **not** correspond to the numbering of the MUSE channels, so that individual channels should be looked up by name. However, the sequence of the channels as they are stored in the FITS file is fixed.

5.2 Static Calibration Data

In addition to the calibration data which are created by the MUSE calibration recipes the MUSE DRS requires a number of so called “static” calibrations[3], which are prepared manually, and which are part of the MUSE DRS distribution package. After the installation procedure is complete, the static calibrations are located in `<installation prefix>/calib/muse-X.Y.Z/cal` unless a different place has been specified during the installation.

The set of static calibrations is summarized in the following with short description of the contents of each of the files. For a detailed description of the data layout please refer to Appendix A.

[2] The term “channel” is equivalent to IFU and the two terms may be used interchangably.

[3] Here “static” means that these files are not created by a recipe, but prepared manually by other means. They may however change from one pipeline release to the next.

`astrometric_catalog.fits`	The catalog positions of the astrometric reference object for all the fields used according to the MUSE calibration plan. Each field is stored in a separate extension.
`badpix_table.fits`	The detector positions of well known bad pixels. The data for each channel is stored in a separate extension. The positions listed here will be propagated as part of the data quality extension of the product files.
`extinct_table.fits`	Atmospheric extinction at the Paranal observatory as a function of wavelength [Å] in units of mag airmass^{-1}.
`filter_list.fits`	The filter transmission as a function of wavelength [Å] for various filters which are used to define the wavelength band for the reconstruction of field-of-view images from the observed spectra. Each transmission curve is stored in a separate extension.
`line_catalog.fits`	A list of air wavelengths [Å] of arc lamp lines. Additional information on relative flux, originating ion and line "quality" flags are also present and may be used by the DRS.
`geometry_table_wfm.fits`	The wide-field mode geometrical calibration, i.e. the mapping of the individual slices from the CCD of each IFU into the MUSE field-of-view
`sky_lines.fits`	A list of OH line transitions and other sky lines used for modelling and correction of the sky background.
`std_flux_table.fits`	The spectra of the spectrophotometric standard stars used according to the MUSE calibration plan. The spectra of the each standard star are stored in a separate extension. For each standard star the air wavelength [Å], the flux [erg s^{-1} cm^{-2} Å^{-1}] and its error are given.

For calibration files like the astrometric catalog, the filter list or the standard flux table, the name of the field, the filter or the object is the name of the extension and can be obtained by looking up the `EXTNAME` header keyword.

In addition to this set of static calibrations, the distribution kit will also contain a few master calibration files, which in principle can be created using the provided MUSE DRS recipes. These files, the lsf-table (`lsf_table.fits`), the astrometric calibration (`astrometry_wcs_wfm.fits`), and the two response curves (`std_response_wfm-e.fits`, `std_response_wfm-n.fits`) contain the model of the MUSE line spread function, the WFM astrometric calibration, and the WFM response curves for the extended and nominal wavelength range respectively. These files are provided as a convenience since their creation is either very time consuming (lsf-table), where found to be stable in the analysis of the available commissioning data and/or may be used as a fallback solution in view of the mentioned limitations of the current pipeline version (cf. Section 1.1).

Strictly speaking also the geometry table could be created using the MUSE DRS. However this requires a large, specific data set and expert knowledge to get to a good result. Thus the geometry table is, and will be distributed by ESO as a static calibration.

5.3 Pipeline Products

The MUSE DRS is designed to propagate errors and data quality information together with the processed data from the beginning to the very end of the data reduction sequence. The errors and data quality information is stored in the MUSE DRS product files following the convention described in [RD5]. In the following the general layout of the products carrying this information is summarized. A detailed description of the format of all products can be found in Appendix A.

5.3.1 MUSE Images

A MUSE image, i.e. the reduced image of a single CCD, as it is, or can be created by almost every MUSE recipe, has three 2D FITS image extensions:

`DATA`	The reduced image.
`DQ`	The data quality flags encoded in an integer value according to the Euro3D standard (cf. [RD6]).
`STAT`	The variance of the reduced data.

The labels `DATA`, `DQ`, and `STAT` refer to the name of the respective FITS extension (i.e. the `EXTNAME` keyword).

5.3.2 MUSE Pixel Tables

The pixel table format is used to store the intermediate products of the pre-processing recipe **muse_scibasic** and a few post-processing recipes. This format is a tabular format which contains 1 row for each CCD pixel of a single IFU, where the value of each pixel is stored together with its “coordinates” (position and wavelength). This format has been introduced to avoid intermediate re-sampling of the data. The pixel table keeps the un-resampled pixel value until the very end of the data reduction. What is changed instead in the different processing steps are only the “coordinates” which are assigned to each pixel. In addition to the value of each CCD pixel and its position, the data quality and the statistical error are also recorded.

5.3.3 MUSE Data Cubes

The final output of the MUSE pipeline are the MUSE data cubes. Similar to the MUSE image format the data cubes consist of up to three 3D FITS cube (`NAXIS=3` volume data) extensions. The FITS cube extensions are also named `DATA`, `DQ`, and `STAT`. However, by default, the data quality extension is not present, instead pixels which do not have a clean data quality status are directly encoded as *Not-a-Number* (NaN) values in the `DATA` extension itself.

In an extended form, the data cube may also contain one or more — one for each selected filter — 2D FITS image extensions with the reconstructed FOV images and the corresponding variance images. However, by default the FOV images are saved as individual files in the standard MUSE image format.

5.3.4 Combined Product Data

By default the MUSE calibration recipes write product files which contain the data belonging to a single IFU, and therefore write 24 files for each kind of product. For example the recipe **muse_bias** creates 24 master bias products on disk. This keeps the file size is small and the product files can be easily handled by standard tools (visualization tools, FITS header viewers, etc).

In addition to that there is a combined format for the product files, where the results for the individual IFUs are stored in a single FITS file, similar to the MUSE raw data files. In this format the extensions are named as in the individual products with a prefix identifying the IFU from which it originates. For example, a combined master bias file contains 72 FITS extensions (data, error and data quality extensions for 24 IFUs) which are called `CHAN<nifu>.DATA`, `CHAN<nifu>.DQ`, and `CHAN<nifu>.STAT`, with nifu $= 01 \ldots 24$.

With the exception of the pixel tables the combined format is available for all pipeline products which would otherwise be saved to disk as 24 individual files. The MUSE recipes can read combined products as well as products stored as individual files. For different calibrations, the two product formats may even be mixed. For instance the master calibration file `lsf_table.fits` which is distributed as part of the MUSE pipeline kit is stored as a combined product.
With the current version of the pipeline writing combined products is not yet supported. It will become available in version 1.0 of the MUSE pipeline.

6 Data Reduction Cookbook

6.1 Getting Started with EsoRex

EsoRex is a command-line tool which can be used to execute not only the MUSE DRS recipes, but the recipes of all standard VLT/VLTI instrument pipelines. With *EsoRex* in your path, the general structure of an *EsoRex* command line is

```
1> esorex [esorex options] [recipe [recipe options] [sof [sof]...]]
```

where options appearing before the recipe name are options for *EsoRex* itself, and options given after the recipe name are options which affect the recipe.

All available *EsoRex* options can be listed with the command

```
1> esorex --help
```

and the full list of available parameters of a specific recipe can be obtained with the command

```
1> esorex --help <recipe name>
```

The output of this command shows as parameter values the current setting, i.e. all modifications from a configuration file or the command line are already applied.

The listing of all recipes known to *EsoRex* can be obtained with the command

```
1> esorex --recipes
```

The last arguments of an *EsoRex* command are the so-called *set-of-frames*. A *set-of-frames* is a simple text file which contains a list of input data files for the recipe. Each input file is followed by an unique identifier (frame classification or frame tag), indicating the contents of this file. The input files have to be given as an absolute path, however *EsoRex* allows the use of environment variables so that a common directory prefix can be abreviated. Individual lines may be commented out by putting the hash character (#) in the first column. An example of a *set-of-frames* is shown in the following:

```
1> cat bias.sof
/data/muse/raw/MUSE.2014-02-21T09:48:53.153.fits BIAS
$RAW_DATA/MUSE.2014-02-21T09:50:36.645.fits BIAS
$RAW_DATA/MUSE.2014-02-21T09:52:16.513.fits BIAS
$RAW_DATA/MUSE.2014-02-21T09:53:47.996.fits BIAS
#$RAW_DATA/MUSE.2014-02-21T09:55:04.515.fits BIAS
```

These *set-of-frames* files will have to be created by the user using a text editor, for instance. Which classification has to be used with which recipe will be shown in section 6.2.2.

Finally, if more than one *set-of-frames* is given on the command-line *EsoRex* concatenates them into a single *set-of-frames*.

6.2 Data Organization

Running the MUSE pipeline recipes using *EsoRex* requires that the user prepares the *set-of-frames* file. To be able to do this one has to find the correct input files for each recipe and assigns the correct frame tag and one has to find calibration file which were taken with a matching instrument configuration. The following sections will summarize which FITS header keywords are useful for that, what are the valid frame tags and which header keywords should be checked in order to find matching calibration frames.

6.2.1 Useful Header Keywords

The following table summarizes FITS header keywords which provide useful information about the observation, instrument configuration and status. The keywords are grouped by their context and intended use, respectively.

Keyword Name	**Description**
Keywords for frame classification:	
`INSTRUME`	Name of the instrument
`DPR.CATG`	Raw data frame product category
`DPR.TYPE`	Raw data frame product type (i.e. the type of observation)
`DPR.TECH`	Raw data frame observation technique
`TPL.ID`	Name of the template used to create the exposure
`PRO.CATG`	Pipeline product category (i.e. the type of the product)
Keywords describing the observation:	
`OBJECT`	Target name
`RA`	Telescope pointing RA (J2000) [deg]
`DEC`	Telescope pointing DEC (J2000) [deg]
`MJD-OBS`	Modified Julien Date at the start of the exposure
`DATE-OBS`	Human readable version of `MJD-OBS`
`OBS.NAME`	Name of the observation block
`OBS.START`	Start time of the observation block
`OBS.TARG.NAME`	Target name
`TPL.START`	Start time of the template (within the observation block)
`TPL.EXPNO`	Exposure sequence number within the template
`TPL.NEXP`	Number of exposures within the template
`TEL.AIRM.START`	Airmass at exposure start
`TEL.AIRM.END`	Airmass at the end of the exposure
`TEL.PARANG.START`	Parallactic angle at exposure start (from site monitor)
`TEL.PARANG.END`	Parallactic angle at exposure end (from site monitor)
`TEL.AMBI.FWHM.START`	Observatory seeing at exposure start (from site monitor)
`TEL.AMBI.FWHM.END`	Observatory seeing at exposure end (from site monitor)
`TEL.AMBI.RHUM`	Observatory ambient relative humidity (from site monitor)
`TEL.AMBI.TEMP`	Observatory ambient temperature (from site monitor)
Keywords describing the instrument/detector setup:	
`EXPTIME`	Total integration time [s]

Keyword Name	**Description**
`INS.MODE`	Instrument mode (field mode WFM/NFM, AO status, nominal/extended wavelength range)
`INS.OPTI1.NAME`	Wavelength range cutoff filter setting "Blue" or "Clear" for nominal and extended wavelength range, respectively
`INS.OPTI2.NAME`	Field mode setting, "WFM" or "NFM"
`INS.LAMP1.ST`	Blue continuum lamp on/off
`INS.SHUT1.ST`	Blue continuum lamp shutter open/closed
`INS.LAMP2.ST`	Red continuum lamp on/off
`INS.SHUT2.ST`	Red continuum lamp shutter open/closed
`INS.LAMP3.ST`	Neon arc lamp on/off
`INS.SHUT3.ST`	Neon arc lamp shutter open/closed
`INS.LAMP4.ST`	Xenon arc lamp on/off
`INS.SHUT4.ST`	Xenon arc lamp shutter open/closed
`INS.LAMP5.ST`	HgCd arc lamp on/off
`INS.SHUT5.ST`	HgCd arc lamp shutter open/closed
`INS.TEMP4.VAL`	Ambient temperature [°C]
`DET.BINX`	Detector binning along X axis (rows)
`DET.BINY`	Detector binning along Y axis (columns)
`DET.READ.CURNAME`	Detector readout mode name
`DET.CHIP.NAME`	Detector chip (instrument channel) name
`DET.CHIP.LIVE`	Detector alive
Other keywords:	
`EXTNAME`	Name of the FITS extension

Almost all product file which are created by the MUSE pipeline recipes also contain a number of *Quality Control Parameters* (QC parameters). These QC parameters are values which are computed by the MUSE recipes as indicators of the quality of the raw data and the reduction process. They are available from the FITS header of the pipeline products as hierarchical keywords starting with the leading group component "`QC.`".

6.2.2 Data Classification and Association

In order to create the *set-of-frames* for a particulare MUSE recipe, both, the raw input data files and the calibration data files have to classified, i.e. the frame tag which indicates the type of data the file contains has to be determined. The result of this, i.e. the frame tag has to be given in the second column of the *set-of-frames* (cf. section 6.1).

The type of a MUSE raw data files is fully determined by a unique combination of the header keywords `DPR.CATG`, `DPR.TYPE`, and `DPR.TECH`. The type of a MUSE pipeline product is completely determined by the header keyword `PRO.CATG`, and the keyword value is also the frame tag to be used in the *set-of-frames*. The latter also applies to the static calibration files.

Table 6.1 summarizes the the valid raw data frame tags which are defined for the MUSE, together with the corresponding combination of `DPR` keywords and the name of the MUSE recipe used to process them. To classify MUSE raw data files it is almost always sufficient to just look at the keyword `DPR.TYPE`. In particular, the keyword `DPR.TECH` is always equal to "`IFU`" for all MUSE raw data files.

Although the input files to the second stage recipes are strictly speaking pipeline product files, and therefore

Frame tag	Recipe	DPR.CATG	DPR.TYPE	DPR.TECH
BIAS	muse_bias	CALIB	BIAS	IFU
DARK	muse_dark	CALIB	DARK	IFU
FLAT	muse_flat	CALIB	FLAT,LAMP	IFU
SKYFLAT	muse_skyflat	CALIB	FLAT,SKY	IFU
ARC	muse_wavecal	CALIB	WAVE	IFU
MASK	muse_geometry	CALIB	WAVE,MASK	IFU
STD	muse_scibasic	CALIB	STD	IFU
ASTROMETRY	muse_scibasic	CALIB	ASTROMETRY	IFU
SKY	muse_scibasic	SCIENCE	SKY	IFU
OBJECT	muse_scibasic	SCIENCE	OBJECT	IFU

Table 6.1: Raw data frame tags of the first stage recipes

their frame tag is given by the value of the keyword PRO.CATG, they are listed in Table 6.2 for the sake of completeness. Also listed here is a utility recipe which can be used to combine the fully reduced pixel tables which can be created by **muse_scipost** as intermediate products (for details see Section 6.5).

Frame tag	Recipe	PRO.CATG
PIXTABLE_STD	muse_standard	PIXTABLE_STD
PIXTABLE_SKY	muse_create_sky	PIXTABLE_SKY
PIXTABLE_ASTROMETRY	muse_astrometry	PIXTABLE_ASTROMETRY
PIXTABLE_OBJECT	muse_scipost	PIXTABLE_OBJECT
PIXTABLE_OBJECT	muse_exp_combine	PIXTABLE_REDUCED

Table 6.2: "Raw" data frame tags of the second stage recipes

Once one has accumulated a substatial number of MUSE files, possibly even spread over several directories, the *Gasgano* file browser may be a handy tool to not loose track of the data. *Gasgano* is able to scan the files in some specified directories and applies the classification rules for MUSE to them. The resulting frame tag will then be shown in the *Gasgano* GUI in the column *Classification*.

The last step in creating *set-of-frames* files as input for the MUSE recipes is to find the appropriate calibrations, both, static calibrations and master calibrations. This means that one has to select calibrations which originate from the same (or compatible) instrument and/or detector configuration.

The basic set of header keywords which is useful for finding matching calibrations is just made of a few keywords. Since there is only a single detector configuration available for MUSE, there is no real need to verify that the detector parameters of the raw data frames and the calibrations match (in principle the header keywords DET.BINX, DET.BINY, and DET.READ.CURID or DET.READ.CURNAME could be used).

To find a calibrations with a matching instrument setup one can use one of the header keywords INS.MODE and INS.OPTI2.NAME, depending whether the whole instrument configuration should match, or only the field mode. The keyword INS.MODE provides information on the field mode (wide or narrow field mode), whether the AO was used, and, whether the nominal or the extended wavelength range (i.e. with or without the blue cutoff filter in place) was selected[4]. The keyword `INS.OPTI2.NAME` contains only the setup of the field mode. It may be used in cases where the other parameters of the instrument setup do not matter.

Finally, if one obtains calibrations as part of an archive request from the ESO archive, then the ESO archive took already care that they match the raw data of the request. Thus, there may be no need to redo the data association.

6.3 Basic Reduction

Now it is time to start processing the data. During the basic reduction the master calibrations are created from raw calibration data, and they are applied in the pre-processing step. The different steps of the basic reduction are shown in Figure 2.2 and will be executed from left to right.

As already mentioned in Section 2.2, the recipes of this first processing stage are working on the data of a single CCD, and there are basically 3 ways to run the recipes:

- manual mode
- serial mode
- parallel mode (using threads)

where the different modes are selected with the recipe option "`--nifu`". For the manual mode one has to provide the number of the IFU to process, i.e. a number in the range from 1 to 24. With this mode one can also process several IFUs in parallel by launching more than one *EsoRex* command. The serial mode is selected with "`--nifu=0`". In this case all IFUs are processed one after the other with a single *EsoRex* command. Parallel processing of all IFUs is selected with "`--nifu=-1`". All three option can use the same *set-of-frames*, provided that it contains the entries for all 24 IFUs.

In the following examples the data is processed in parallel, i.e. using the recipe option "`--nifu=-1`"; the prefix command for pinning the threads is omitted in order to keep the command line examples easy to read. How to call *EsoRex* with pinning the threads is shown in Section C. For reasons of clarity the environment variable `OMP_NUM_THREADS` is explicitly set for each *EsoRex* command, although it would be sufficient to do it once for the whole session before the first recipe is run.

6.3.1 Bias

In the first processing step the raw bias frames are combined into a master bias frame using the **muse_bias** recipe. The created master bias will then be used in the subsequent reduction steps.

First the location and the raw frame tag of at least 3 raw bias frames is put into the *set-of-frames*:

```
1> cat bias.sof
/data/muse/raw/bias/MUSE.2014-02-21T09:48:53.153.fits BIAS
/data/muse/raw/bias/MUSE.2014-02-21T09:50:36.645.fits BIAS
/data/muse/raw/bias/MUSE.2014-02-21T09:52:16.513.fits BIAS
/data/muse/raw/bias/MUSE.2014-02-21T09:53:47.996.fits BIAS
/data/muse/raw/bias/MUSE.2014-02-21T09:55:04.515.fits BIAS
```

[4]Note that currently neither the narrow field mode, nor is the adaptive optics available!

To process the data in parallel, the number of threads is limited to 24, and all 24 IFUs are then processed with the shown *EsoRex* command:

```
1> OMP_NUM_THREADS=24 esorex --log-file=bias.log muse_bias --nifu=-1 \
   bias.sof
```

Once the *EsoRex* command has finished, the master bias has been created in the current working directory, together with the log file `bias.log`[5] of the *EsoRex* run.

By default[6] the name of the product files created by the MUSE recipes are derived from the value of the `PRO.CATG` header keyword (i.e. their frame tag), with suffixes and sequence numbers added as needed. The last 2 digit sequence number refers to the MUSE channel number.

Thus, after the running **muse_bias**, the current working directory contains now the master bias product stored in 24 FITS files:

```
1> ls -1 *.fits
MASTER_BIAS-01.fits
MASTER_BIAS-02.fits
MASTER_BIAS-03.fits
...
MASTER_BIAS-24.fits
```

An example of a MUSE master bias is shown in Figure 6.1. For the full description of the **muse_bias** recipe please refer to Section 9.

6.3.2 Dark

In this next step a set of raw dark frames is combined into a master dark frame using the recipe **muse_dark**. Since the dark current of the MUSE CCDs is small, it is unlikely that the master dark frame will be needed in the following processing steps. Thus this step is optional.

Following the same procedure as before, the *set-of-frames* is prepared containing at least 3 raw dark frames, and the 24 master bias files from the previous processing step which are the required calibrations. Assuming the master bias files are located in a directory `$MUSE_CAL` the SOF and the *EsoRex* command will look as shown here:

```
1> cat dark.sof
/data/muse/raw/dark/MUSE.2014-02-11T20:32:24.014.fits DARK
/data/muse/raw/dark/MUSE.2014-02-11T20:33:31.876.fits DARK
/data/muse/raw/dark/MUSE.2014-02-11T20:34:31.876.fits DARK
$MUSE_CAL/MASTER_BIAS-01.fits MASTER_BIAS
$MUSE_CAL/MASTER_BIAS-02.fits MASTER_BIAS
$MUSE_CAL/MASTER_BIAS-03.fits MASTER_BIAS
```

[5] Note that in parallel mode the messages written to the terminal and/or the log-file may not appear in order but interlaced depending on the execution order of the different threads, which, in general, is not predictable.

[6] i.e. if *EsoRex* is executed with the option "`--suppress-prefix=true`"; which is the built-in default setting.

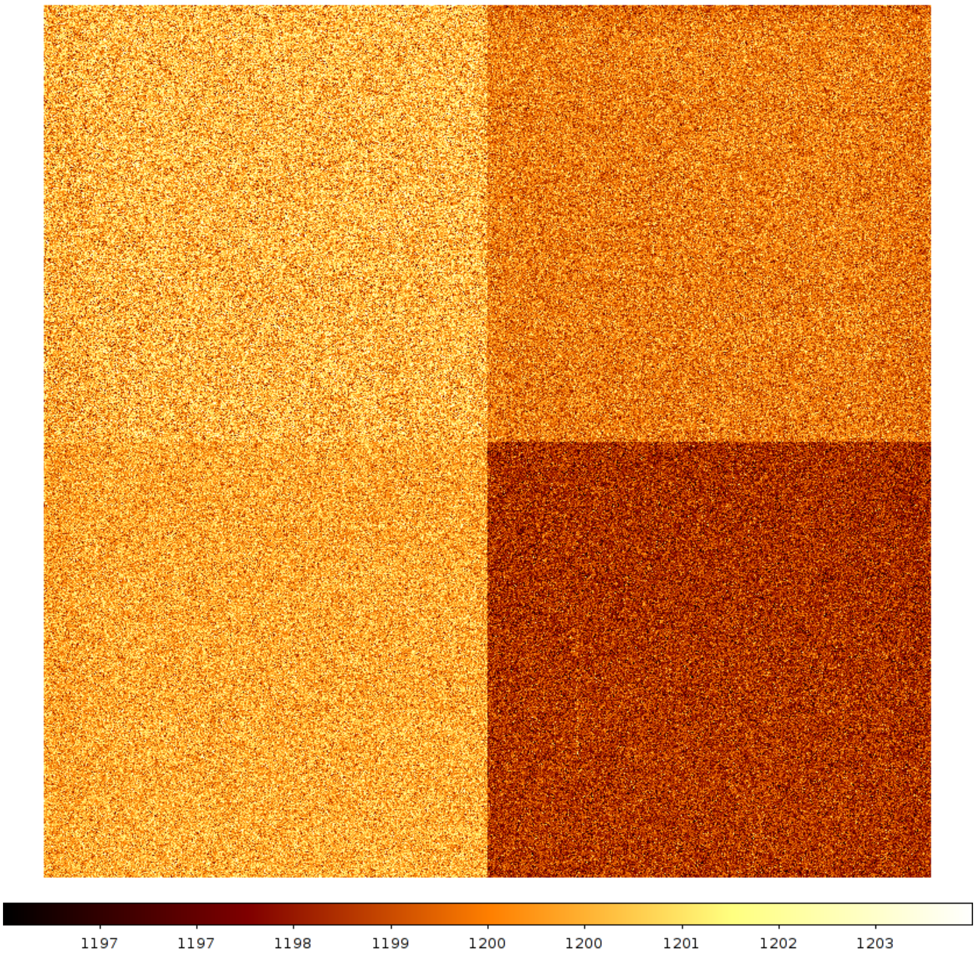

Figure 6.1: Example of a MUSE master bias of a single IFU (channel 1), as it is created by the recipe **muse_bias**.

```
...
$MUSE_CAL/MASTER_BIAS-24.fits MASTER_BIAS
2> OMP_NUM_THREADS=24 esorex --log-file=dark.log muse_dark --nifu=-1 \
   dark.sof
```

The result of this *EsoRex* command is the master dark frame stored as 24 FITS files:

```
1> ls -1 *.fits
MASTER_DARK-01.fits
...
MASTER_DARK-24.fits
```

In the following processing steps, the master dark will not be used. For the full description of the **muse_dark** recipe please refer to Section 9.

6.3.3 Flat Field

In this processing step the recipe **muse_flat** is used to combine raw (lamp) flat field frames into a master flat frame. In addition to that the slices are located on the image, and their edges are traced along the wavelength axis (vertical axis). Finally bright and dark pixels are located.

The *set-of-frames* must contain at least 3 raw flat field frames and the master bias frame. Optionally the master dark frame may added too, but it has been omitted in the following. Instead the bad pixel table has been added to make the tracing of the slices robust against “dead” columns. Currently this is may only be needed for channel 21, and the currently distributed bad pixel table contains only the data for this channel (actually, adding bad columns to the bad pixel table may degrade the tracing solution, especially if they are located near the edge of a slice).

```
1> cat flat.sof
/data/muse/raw/flat/MUSE.2014-02-21T12:14:59.316.fits FLAT
/data/muse/raw/flat/MUSE.2014-02-21T12:16:25.212.fits FLAT
/data/muse/raw/flat/MUSE.2014-02-21T12:18:10.540.fits FLAT
/data/muse/raw/flat/MUSE.2014-02-21T12:19:34.837.fits FLAT
/data/muse/raw/flat/MUSE.2014-02-21T12:20:48.296.fits FLAT
$MUSE_CAL/MASTER_BIAS-01.fits MASTER_BIAS
$MUSE_CAL/MASTER_BIAS-02.fits MASTER_BIAS
...
$MUSE_CAL/MASTER_BIAS-24.fits MASTER_BIAS
$MUSE_CAL/badpix_table.fits BADPIX_TABLE
2> OMP_NUM_THREADS=24 esorex --log-file=flat.log muse_flat --nifu=-1 \
   flat.sof
```

The products of this *EsoRex* command are the master flat frame and the trace tables, each of them stored as 24 FITS files:

```
1> ls -1 *.fits
MASTER_FLAT-01.fits
...
MASTER_FLAT-24.fits
TRACE_TABLE-01.fits
...
TRACE_TABLE-24.fits
```

An example of a MUSE master flat is shown in Figure 6.2. For the full description of the **muse_flat** recipe please refer to Section 9.

6.3.4 Wavelength Calibration

In this processing step the wavelength solution for each IFU is created using the recipe **muse_wavecal**. To create the dispersion solution the recipe needs at least three raw arc-lamp frame as input. One raw frame for each available arc-lamp. It is possible to use more than one of these sets as input.

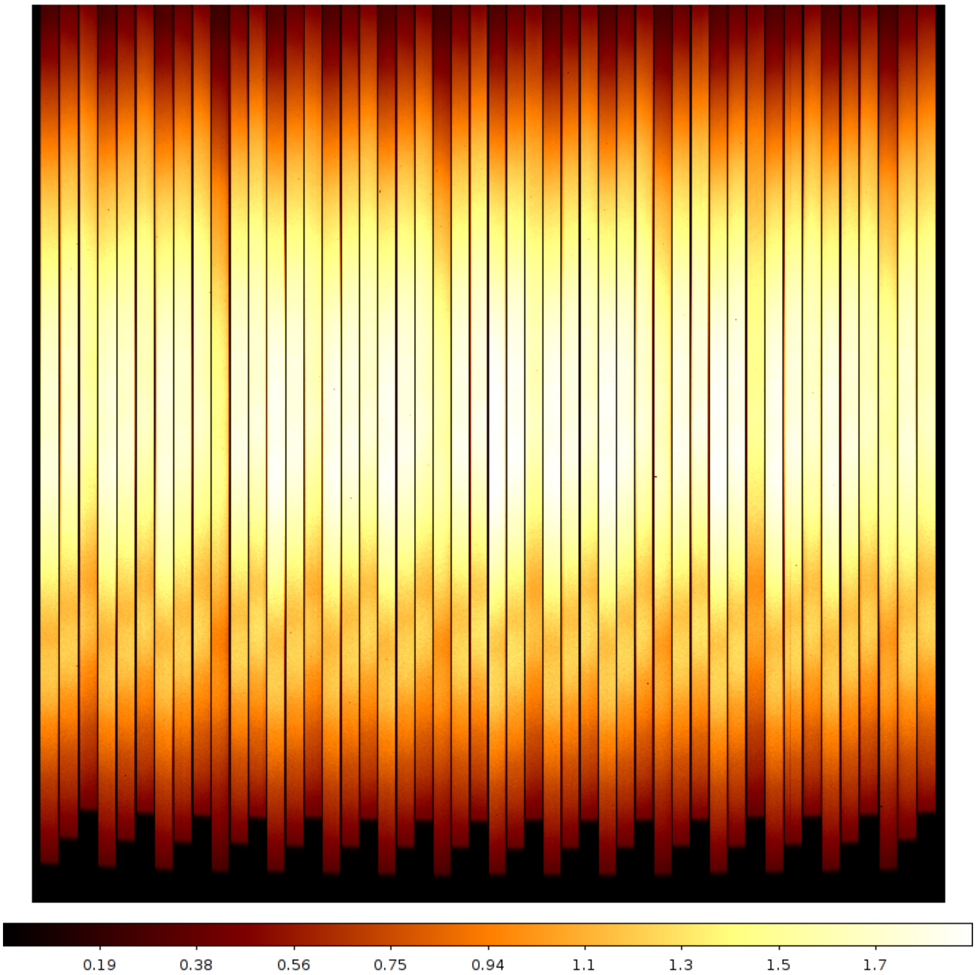

Figure 6.2: Example of a MUSE master flat field of a single IFU (channel 1), as it is created by the recipe **muse_flat**.

These sets of three arc-lamp frames are created by the MUSE calibration template and therefore the a complete set should always be available. In addition, this can be verified by looking at the FITS header keywords indicating the lamp shutter status `INS.SHUT3.ST`, `INS.SHUT4.ST`, and `INS.SHUT5.ST`[7]. Only one of the keywords should indicate an open lamp shutter for a single arc-lamp frame, and for a complete input set one needs 3 such arc-lamp frames, one for each shutter.

Dark correction and flat field correction will be applied if the master dark frame and the master flat field frame are present in the *set-of-frames* respectively. However applying the two corrections is not needed for the arc-lamp frames, and thus the input data for these correction are omitted in the *set-of-frames* shown in the following.

What is needed from the previous processing steps are the trace tables and the master bias frame, and from the static calibrations the line catalog is required. The *set-of-frames* and the *EsoRex* command line to create the

[7]To verify that the lamps a actually active the keywords `INS.LAMP3.ST`, `INS.LAMP4.ST`, and `INS.LAMP5.ST` should be checked.

wavelength solution are shown here:

```
1> cat wavecal.sof
/data/muse/raw/wave/MUSE.2014-02-19T10:58:29.838.fits ARC
/data/muse/raw/wave/MUSE.2014-02-19T11:00:03.593.fits ARC
/data/muse/raw/wave/MUSE.2014-02-19T11:02:10.205.fits ARC
$MUSE_CAL/MASTER_BIAS-01.fits MASTER_BIAS
$MUSE_CAL/MASTER_BIAS-02.fits MASTER_BIAS
...
$MUSE_CAL/MASTER_BIAS-24.fits MASTER_BIAS
$MUSE_CAL/TRACE_TABLE-01.fits TRACE_TABLE
...
$MUSE_CAL/TRACE_TABLE-24.fits TRACE_TABLE
$MUSE_CAL/line_catalog.fits LINE_CATALOG
2> OMP_NUM_THREADS=24 esorex --log-file=wavecal.log muse_wavecal --nifu=-1 \
   --lsf --resample --residuals wavecal.sof
```

The primary products created by this command are the wavelength solution and the lsf-table. The latter contains the line spread function computed for each slice, which is used later during the sky subtraction. The creation of the lsf-table is enabled with the recipe option “`--lsf`”. Note that creating the lsf-table increases the execution time of the recipe considerably.

As an alternative to creating the lsf-table with **muse_wavecal**, the lsf-table that comes as part of the MUSE pipeline package could be used in the following reduction steps. The analysis of MUSE commissioning data show that the parameters of the line-spread-function are stable. Note that the lsf-table included in the pipeline distribution is stored as a combined product, and replaces all 24 `LSF_TABLE` entries in the SOF show before.

In addition to the primary products the shown *EsoRex* command also creates for each IFU a product file with the fit residuals of the arc-lines, reduced arc-lamp images (for each lamp separately and a combined one), and a resampled arc-lamp image which can be used for a visual inspection of the wavelength solution. An example is shown in Figure 6.3.

For the full description of the **muse_wavecal** recipe please refer to Section 9.

6.3.5 Skyflat

Twilight flats are part of the regular calibrations taken for MUSE. In this processing step the raw sky-flat frames are combined into a master sky-flat frame using the the recipe **muse_skyflat**.

Note that the actual pixel data of the resulting master sky-flat is not used by the subsequent processing steps. What is used instead for the later illumination correction[8] are the integrated flux value in the header of the master sky-flat frame, as an estimate of the relative throughput of the IFUs.

The recipe needs as input at least 3 raw sky-flat frames, the master bias frame, the trace tables and the wavelength solution. The *set-of-frames* and the *EsoRex* command for creating the master sky-flat are shown:

```
1> cat skyflat.sof
/data/muse/raw/MUSE.2014-02-20T23:31:38.542.fits SKYFLAT
```

[8] At the time of writing, the way how the illumination correction is done is subject to change.

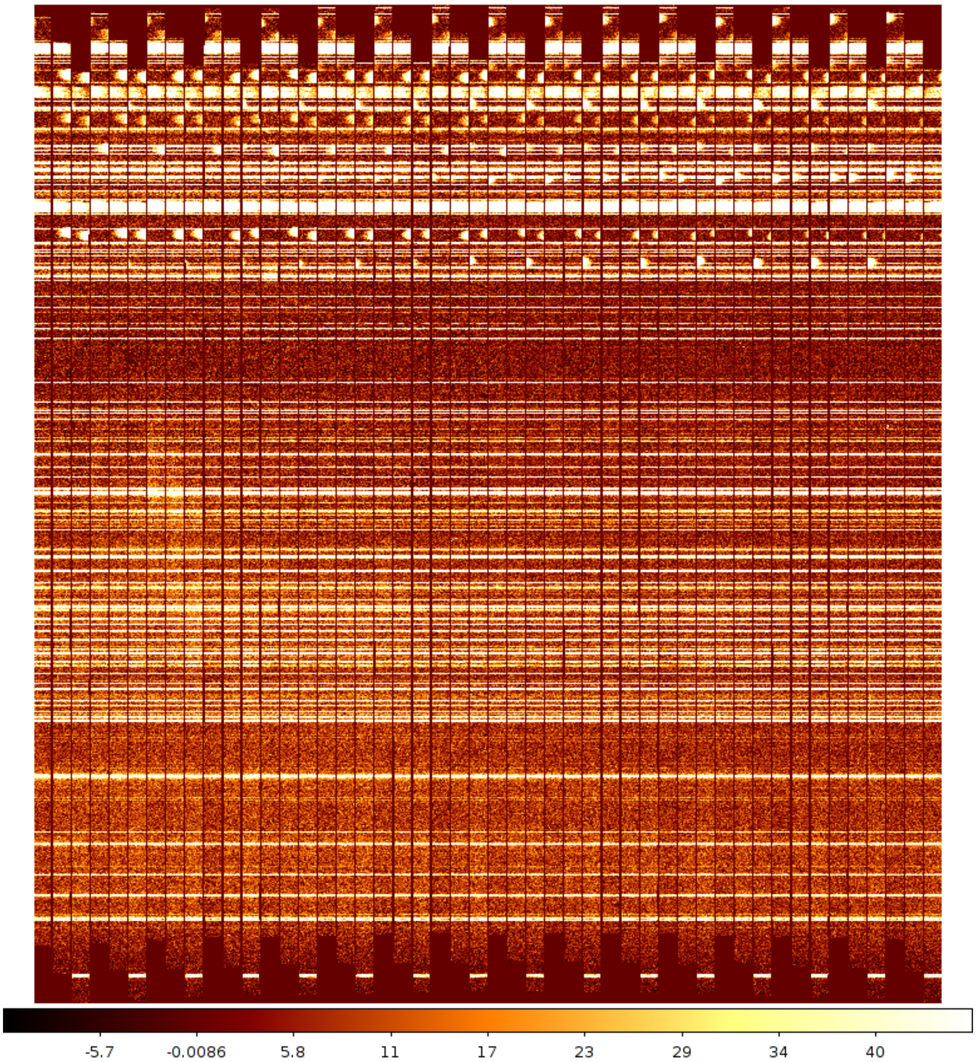

Figure 6.3: Example of a resampled arc-lamp frame of a single IFU (channel 1), as it is created by the recipe **muse_wavecal**. This image is the result of combining the three individual arc-lamp images and applying the determined wavelength solution. As it is expected from a good wavelength solution the arc-lines appear as straight, horizontal lines, i.e. they are now on the same wavelength.

```
/data/muse/raw/MUSE.2014-02-20T23:33:49.258.fits SKYFLAT
/data/muse/raw/MUSE.2014-02-20T23:36:21.511.fits SKYFLAT
$MUSE_CAL/MASTER_BIAS-01.fits MASTER_BIAS
...
$MUSE_CAL/MASTER_BIAS-24.fits MASTER_BIAS
$MUSE_CAL/TRACE_TABLE-01.fits TRACE_TABLE
...
$MUSE_CAL/TRACE_TABLE-24.fits TRACE_TABLE
$MUSE_CAL/WAVECAL_TABLE-01.fits WAVECAL_TABLE
...
```

```
$MUSE_CAL/WAVE_TABLE-24.fits WAVECAL_TABLE
2> OMP_NUM_THREADS=24 esorex --log-file=skyflat.log muse_skyflat --nifu=-1 \
   skyflat.sof
```

For the full description of the **muse_skyflat** recipe please refer to Section 9.

6.3.6 Instrument Geometry

The instrument geometry or geometrical calibration determines where in the field-of-view each slice of each IFU is located. The result is the geometry table which can be created using the **muse_geometry** recipe.

However, this recipe requires special input data and some expert knowledge to get to a good result! For this reason geometry tables are carefully prepared in advance. They are considered as a static calibration which is distributed as part of the MUSE pipeline release package. Therefore, the description of **muse_geometry** is skipped.

6.4 Observation Pre-processing

At this point all necessary calibrations have been created to remove the instrument signature from the on-sky exposures using the pre-processing recipe **muse_scibasic**. This also converts the observations to the pixel table format which is the input format for the post-processing recipes.

Pixel tables can be saved as a true FITS binary table, or as FITS images. To benefit from a more efficient way to read and write the results, it is suggested to use the FITS image based format. This is done by setting the environment variable `MUSE_PIXTABLE_SAVE_AS_IMAGE` as shown below.

In the example the usage of **muse_scibasic** is shown for a scientific exposure as input (i.e. for the raw frame tag `OBJECT`). But it has to be applied in the same way to the observations of standard stars, astrometric fields and empty sky fields (i.e. the raw frame tags `STD`, `ASTROMETRY`, and `SKY`).

For each of these four types of observations the recipe takes one or more raw exposures as input. The required calibrations are master bias, master flat, trace table, wavelength solution and the geometry table. Optionally a master sky-flat and a bad pixel table may be used. The resulting *set-of-frames* and the *EsoRex* command to run **muse_scibasic** are shown here:

```
1> cat object.sof
/data/muse/raw/object/MUSE.2014-02-21T01:50:23.690.fits OBJECT
$MUSE_CAL/MASTER_BIAS-01.fits MASTER_BIAS
...
$MUSE_CAL/MASTER_BIAS-24.fits MASTER_BIAS
$MUSE_CAL/MASTER_FLAT-01.fits MASTER_FLAT
...
$MUSE_CAL/MASTER_FLAT-24.fits MASTER_FLAT
$MUSE_CAL/MASTER_SKYFLAT-01.fits MASTER_SKYFLAT
...
$MUSE_CAL/MASTER_SKYFLAT-24.fits MASTER_SKYFLAT
$MUSE_CAL/TRACE_TABLE-01.fits TRACE_TABLE
```

```
...
$MUSE_CAL/TRACE_TABLE-24.fits TRACE_TABLE
$MUSE_CAL/WAVECAL_TABLE-01.fits WAVE_TABLE
...
$MUSE_CAL/WAVE_TABLE-24.fits WAVECAL_TABLE
$MUSE_CAL/badpix_table.fits BADPIX_TABLE
$MUSE_CAL/geometry_table.fits GEOMETRY_TABLE
2> export MUSE_PIXTABLE_SAVE_AS_IMAGE=1
3> OMP_NUM_THREADS=24 esorex --log-file=object.log muse_scibasic --nifu=-1 \
   object.sof
```

The command shown before creates the pixel table stored as 24 FITS files for each input exposure, and one reduced image for each IFU.

```
1> ls -1 *.fits
OBJECT_RED_0001-01.fits
...
OBJECT_RED_0001-24.fits
PIXTABLE_OBJECT_0001-01.fits
...
PIXTABLE_OBJECT_0001-24.fits
```

The product file names are derived from the frame tag of the raw input exposure, which, in this case, was "OBJECT". For the other kind of on-sky exposures the product files are named in the same way but with the OBJECT-part replaced by the corresponding frame tag.

The 4 digit number indicates the index of the input exposure, which in this case is always "0001" since there was only a single raw science exposure present in the SOF. If more than one raw exposure is present the index will run from 1 to the number of exposures.

The full description of the **muse_scibasic** recipe can be found in Section 9.

6.5 Observation Post-Processing

This section covers the second stage of the MUSE data reduction cascade shown in Figure 2.3. Again the processing steps are executed from left to right, although only the response computation is mandatory for creating the final data cube.

6.5.1 Flux Calibration

In this first processing step the response curve for the flux calibration of the science observations is created from a pre-processed standard star observation. The raw standard star observation has already been processed with **muse_scibasic** and resulting 24 pixel tables are now located in the current working directory.

In addition to the set of 24 pixel tables, the recipe requires an exintion table, and a standard flux table. Optionally a list of predefined telluric region may be present in the input SOF (which is omitted in the following).

The standard flux table is a catalog of reference spectra of spectrophotometric standard stars. The current version of this catalog, which is included in the MUSE pipeline distribution as `std_flux_table.fits`, will eventually contain spectra of all spectrophotometric standard stars which are regularly observed with MUSE. The standard stars in the standard flux table are looked up by their coordinates.

Currently reference spectra for the following standard stars are available:

- GD71
- GD108
- GD153
- HD49798
- CD-32 9927
- GJ 754.1 A
- LDS 749B
- Feige 110

The complete *set-of-frames* for computing the response curve is shown here, followed by the *EsoRex* command line to launch the recipe.

```
1> cat std.sof
PIXTABLE_STD_0001-01.fits PIXTABLE_STD
...
PIXTABLE_STD_0001-24.fits PIXTABLE_STD
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/std_flux_table.fits STD_FLUX_TABLE
2> OMP_NUM_THREADS=24 esorex --log-file=std.log muse_standard \
   --filter=white std.sof
```

The recipe creates a data cube of the standard star observation, and the response curve and the telluric correction spectrum.

```
1> ls -1 *.fits
DATACUBE_STD_0001.fits
STD_RESPONSE_0001.fits
STD_TELLURIC_0001.fits
```

Again the sequence number is an exposure index, which runs from 1 to the number of exposures present in the SOF.

The data cube can be visualized using the ESO 3D Visualization tool (cf. RD[7], Section D) or `ds9` for instance. In addition to the data cube itself, the data cube product file contains a reconstructed FOV image for each filter given on the command line (option `--filter`). The default filter setting is "white" to produce a white light image. All other allowed filters are the ones defined in the filter list calibration file (`filter_list.fits`;

the filter names are given by the keyword EXTNAME). If a filter from the filter list is chosen the filter list must also be present in the SOF.

If more than one standard star observation is present in the input SOF the response computation is done separately for each of the input exposures.

For the full description of the **muse_standard** recipe please refer to Section 9.

6.5.2 Sky Creation

In this processing step a model of the sky spectrum is created using the recipe **muse_create_sky**. It is only needed if the object that one wants to process fills much of the field-of-view. Otherwise, if the science exposure contains enough sky, the sky subtraction is done directly when the science exposure is processed.

The recipe requires that exposure(s) of a mostly empty sky field are available and have been pre-processed as shown in Section 6.4. It also recommended use the lsf-table(s) which have been created during the wavelength calibration (cf. Section 6.3.4). They contain the parameters of the MUSE line-spread-function which are used to create a smooth sky continuum model by subtracting the fitted sky lines model from the full sky spectrum.

The required calibrations are the list of sky lines, the extinction table and the response curve, and, optionally, the telluric correction. The input *set-of-frames*, and the *EsoRex* command line to run **muse_create_sky** using default settings are shown:

```
1> cat sky.sof
PIXTABLE_SKY_0001-01.fits PIXTABLE_SKY
...
PIXTABLE_SKY_0001-24.fits PIXTABLE_SKY
$MUSE_CAL/LSF_TABLE-01.fits LSF_TABLE
...
$MUSE_CAL/LSF_TABLE-24.fits LSF_TABLE
$MUSE_CAL/STD_RESPONSE_0001.fits STD_RESPONSE
$MUSE_CAL/STD_TELLURIC_0001.fits STD_TELLURIC
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/sky_lines.fits SKY_LINES
2> OMP_NUM_THREADS=24 esorex --log-file=sky.log muse_create_sky \
   sky.sof
```

The result of the above command are the products

```
1> ls -1 *.fits
IMAGE_FOV.fits
SKY_CONTINUUM.fits
SKY_LINES.fits
SKY_MASK.fits
SKY_SPECTRUM.fits
```

These are the reconstructed whitelight field-of-view image, the estimated sky continuum spectrum, estimated fluxes of the sky lines, image mask of the sky regions derived from the white light image, and the sky spectrum obtained from the sky regions defined by the mask.

Note that the sky continuum is always created, even if no lsf-table(s) are present in the input SOF. However, a reasonably smooth version of the sky continuum will only be obtained if the lsf-table(s) are used.

For the full description of the **muse_create_sky** recipe please refer to Section 9.

6.5.3 Astrometry

This processing step uses **muse_astrometry** to create the astrometric calibration from the pre-processed pixel tables of an exposure of an astrometric field. The obtained astrometric calibration is needed to create the world coordinate system for the science exposure and assign right ascension and declination to the spaxels of the data cube.

In addition to the input pixel tables the recipe needs an astrometric reference catalog. The astrometric catalog that comes with the MUSE pipeline distribution will eventually contain the positions of the reference stars in all astrometric fields which are regularly observed with MUSE. Currently the distributed catalog includes 10 fields (2 fields for each of the 5 globular clusters).

Optionally the extinction table, a response curve, and a telluric correction may be present in the input *set-of-frames*. If the optional calibrations are present, the corresponding corrections are performed before the reference objects are identified and the WCS solution is computed. However, experience shows that the improvement that can be achieved by using the optional calibrations is very limited, if it is present at all.

The following shows the input *set-of-frames* and the *EsoRex* command to create the WCS solution:

```
1> cat astrometry.sof
PIXTABLE_ASTROMETRY_0001-01.fits PIXTABLE_ASTROMETRY
...
PIXTABLE_ASTROMETRY_0001-24.fits PIXTABLE_ASTROMETRY
$MUSE_CAL/STD_RESPONSE_0001.fits STD_RESPONSE
$MUSE_CAL/STD_TELLURIC_0001.fits STD_TELLURIC
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/astrometric_catalog.fits ASTROMETRY_REFERENCE
2> OMP_NUM_THREADS=24 esorex --log-file=astrometry.log muse_astrometry \
   astrometry.sof
```

The results are the data cube and the computes WCS solution:

```
1> ls -1 *.fits
ASTROMETRY_WCS_0001.fits
DATACUBE_ASTROMETRY_0001.fits
```

The pipeline distribution comes with a carefully verified WCS solution as a master calibration, which can be used in the subsequent reduction steps to speed up the processing. Since the WCS solution was found to be stable when analyzing the available commissioning data, there is no need to recreate the astrometric calibration, and this step may safely be skipped.

For the full description of the **muse_astrometry** recipe please refer to Section 9.

6.5.4 Science Observations

At this point all calibrations are in place and one can start with the post-processing of the science observations using **muse_scipost**. The recipes applies the previously created on-sky calibrations and creates the final data cube from the pre-processed pixel tables of scientific observations.

In the following example a full data cube is created from a single exposure, the sky background is removed and the flux calibration and the astrometric calibration, created in the preceding processing steps, are applied. Finally four field-of-view images are created from the data cube, each of them covering the wavelength band of one of the filters given on the command line.

Thus, the input *set-of-frames* will contain the 24 pre-processed pixel tables of the exposure of the science target, the estimated fluxes of the sky lines, the response curve, the telluric correction, the extinction table, the astrometric calibration, and the list of filter transmission curves.

The input *set-of-frames* and the *EsoRex* command to run **muse_scipost** are show in the following. Note that, depending on the orientation of the original scientific exposure (cf. Section 3.1), this command will need between approximately 18 GB and 25 GB of RAM in order finish sucessfully.

```
1> cat scipost.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
$MUSE_CAL/LSF_TABLE-01.fits LSF_TABLE
...
$MUSE_CAL/LSF_TABLE-24.fits LSF_TABLE
$MUSE_CAL/STD_RESPONSE_0001.fits STD_RESPONSE
$MUSE_CAL/STD_TELLURIC_0001.fits STD_TELLURIC
$MUSE_CAL/ASTROMETRY_WCS_0001.fits ASTROMETRY_WCS
$MUSE_CAL/sky_lines.fits SKY_LINES
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/filter_list.fits FILTER_LIST
2> OMP_NUM_THREADS=24 esorex --log-file=scipost.log muse_scipost \
   --filter=white,Johnson_V,Cousins_R,Cousins_I scipost.sof
```

The product files created by this command are:

```
1> ls -1 *.fits
DATACUBE_FINAL.fits
IMAGE_FOV_0001.fits
IMAGE_FOV_0002.fits
IMAGE_FOV_0003.fits
IMAGE_FOV_0004.fits
```

The 4 digit sequence number of the created field-of-view images refers to the different filters. The “ordering” of the field-of-view images corresponds to the order of the filters on the command line.

As mentioned before, running **muse_scipost** to process a single exposure requires already a quite capable computer (in terms of memory). And, contrary to the basic calibration recipe reducing the number of threads will not reduce the memory needed. It is also not possible to process an observation by splitting it into smaller tiles.

However, it is possible to limit the wavelength range to process using the recipe options `--lambdamin` and `--lambdamax`, which will reduce the memory usage (cf. Section 7.1).

The recipe **muse_scipost** accepts the pixel tables of more than one exposure as input. In this case the exposures will be combined into a single data cube by default. However, in order to obtain the expected results when combining exposures read Section 6.6 first!

As long as single science exposures are concerned, the data processing has finished at this point and one can start with the scientific analysis of the data.

For the full description of the **muse_scipost** recipe please refer to Section 9.

6.6 Combining Exposures

This section shows how multiple exposures are combined efficiently and correctly, and should help to bypass some pitfalls. The combination of exposures, i.e. the combination of the pre-processed pixel tables of a scientific exposure can be done with two different methods (recipes). The exposures can be combined directly in **muse_scipost**, or one can use the utility recipe **muse_exp_combine**. The latter uses a fully reduced pixel table, which is an intermediate product of **muse_scipost**.

There are two issues which have to be considered when multiple exposures are combined, the correction of coordinate offsets, and limiting the wavelength range.

6.6.1 Correcting Coordinate Offsets

When combining multiple exposures, the pipeline recipes **muse_scipost** and **muse_exp_combine** usually do 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° rotations, condition 2 is often not true, and the user has to provide offset corrections to the pipeline.[9] The reference may be some external data or one of the exposures itself. For instance, the reconstructed field-of-view images may be used for measuring the coordinate offsets (creating these images can be done quickly with **muse_scipost** using a vary narrow range in wavelength).

The offset corrections can then be applied in two ways:

[9] This is due to a slight decentering of the axis of the derotator, leading to a "derotator wobble". Future versions of the pipeline should be able to automatically correct this effect.

Edit the FITS headers:

Before starting **muse_scipost** or **muse_exp_combine** to create a combined cube, edit the main FITS headers of the input pixel tables and replace the `RA` and `DEC` header keywords with corrected values that are measured externally.

Provide offsets:

This can be done using two environment variables `MUSE_XCOMBINE_RA_OFFSETS` and `MUSE_XCOMBINE_DEC_OFFSETS`. Each of these variables has to contain as many floating-point numbers, separated by commas, as exposures to combine are involved. 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:

$$\begin{aligned} \mathtt{RA\_OFFSET} &= \mathtt{RA}_{\text{measured}} - \mathtt{RA}_{\text{reference}} \\ \mathtt{DEC\_OFFSET} &= \mathtt{DEC}_{\text{measured}} - \mathtt{DEC}_{\text{reference}} \end{aligned}$$

It is important that the offsets are given in the order of increasing `DATE-OBS` of the exposures involved. If one of the exposures has been chosen as the reference the offsets of 0 for this exposure have to be explicitly given when setting the environment variables.

6.6.2 Limiting Wavelength Ranges

All recipes reading pixel tables support the `--lambdamin` and `--lambdamax` recipe parameters to restrict the wavelength range they are working on. These recipe parameters expect a wavelength given in Angstrom, and this can be used to limit the amount of memory required by the recipes (see Section 7.1 for details).

For the combination of exposures the limiting factor in terms of memory is the size of the output data cube, which has to be constructed in memory. Thus, in limiting the wavelength range the output data cube can cover a larger area on the sky.

6.6.3 Combining Exposures using muse_scipost

The following example shows how three exposures are combined using **muse_scipost** alone. The input *set-of-frames* now contains the pixel tables of the three exposures (note the 4-digit index in the names of the pixel tables). Apart from the additional pixel tables the *set-of-frames* is identical to the one show previously for processing a single exposure.

The *EsoRex* command line shows the three coordinate offsets for the 3 input exposures where the first exposure is used as the coordinate reference. The wavelength range is restricted to the intervall 6900 Å to 7100 Å.

```
1> cat scipost_combine.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0002-01.fits PIXTABLE_OBJECT
...
```

```
PIXTABLE_OBJECT_0002-24.fits PIXTABLE_OBJECT
PIXTABLE_OBJECT_0003-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0003-24.fits PIXTABLE_OBJECT
$MUSE_CAL/LSF_TABLE-01.fits LSF_TABLE
...
$MUSE_CAL/LSF_TABLE-24.fits LSF_TABLE
$MUSE_CAL/STD_RESPONSE_0001.fits STD_RESPONSE
$MUSE_CAL/STD_TELLURIC_0001.fits STD_TELLURIC
$MUSE_CAL/ASTROMETRY_WCS_0001.fits ASTROMETRY_WCS
$MUSE_CAL/sky_lines.fits SKY_LINES
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/filter_list.fits FILTER_LIST
2> OMP_NUM_THREADS=24 \
   MUSE_XCOMBINE_RA_OFFSETS="0.0000000,-5.4920943e-05,-0.00031381379" \
   MUSE_XCOMBINE_DEC_OFFSETS="0.0000000,8.3338105e-05,7.2223397e-05" \
   esorex --log-file=scipost_combine.log muse_scipost \
   --lambdamin=6900. --lambdamax=7100. scipost_combine.sof
```

The product files created by this command are:

```
1> ls -1 *.fits
DATACUBE_FINAL.fits
IMAGE_FOV_0001.fits
```

The final data cube covers the combined field-of-view of the individual input exposures, but only the small wavelength band defined by the recipe parameters. By default the recipe `muse_scipost` creates a white light field-of-view image, which in the above case covers only the limited wavelength range.

6.6.4 Combining Exposures using muse_exp_combine

With the previous method, one has to re-run the full science post-processing in order to get the data cubes for a different wavelength band or combination of input fields. A slightly more flexible approach is shown here.

Assuming that the computer used to run **muse_scipost** is capable of processing a single exposure without restricting the wavelength range. In this case one can create a fully reduced pixel table for the first exposure using the following *EsoRex* command:

```
1> cat scipost_1.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
$MUSE_CAL/LSF_TABLE-01.fits LSF_TABLE
...
$MUSE_CAL/LSF_TABLE-24.fits LSF_TABLE
$MUSE_CAL/STD_RESPONSE_0001.fits STD_RESPONSE
$MUSE_CAL/STD_TELLURIC_0001.fits STD_TELLURIC
$MUSE_CAL/ASTROMETRY_WCS_0001.fits ASTROMETRY_WCS
```

```
$MUSE_CAL/sky_lines.fits SKY_LINES
$MUSE_CAL/extinct_table.fits EXTINCT_TABLE
$MUSE_CAL/filter_list.fits FILTER_LIST
2> OMP_NUM_THREADS=24 esorex --log-file=scipost_1.log muse_scipost \
   --save_individual scipost_1.sof
```

The product files created by this commands are:

```
1> ls -1 *.fits
DATACUBE_FINAL_0001.fits
IMAGE_FOV_0001.fits
PIXTABLE_REDUCED_0001.fits
```

which are then renamed to prevent overwriting them with the *EsoRex* calls needed to process the remaining exposures:

```
1> mv DATACUBE_FINAL.fits DATACUBE_FINAL_EXP01.fits
2> mv IMAGE_FOV_0001.fits IMAGE_FOV_0001_EXP01.fits
3> mv PIXTABLE_REDUCED_0001.fits PIXTABLE_REDUCED_0001_EXP01.fits
```

By using the recipe parameter `--save_individual` a fully reduced pixel table is created as an additional product. Fully reduced means that all on-sky calibrations have been applied, and the only processing step missing is the resampling into the final data cube.

Proceeding in the same way with the other exposures one finally ends up with one fully reduced pixel table for each exposure. These can now be used as input for the recipe **muse_exp_combine** to create a final, combined data cube. The advantage is that creating cubes for different wavelength bands or combination of the exposures now involves only the resampling step.

To get to the same final data cube as in the previous section, the command to be issued is:

```
1> cat exp_combine.sof
PIXTABLE_REDUCED_0001_EXP01.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP02.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP03.fits PIXTABLE_REDUCED
$MUSE_CAL/filter_list.fits FILTER_LIST
2> OMP_NUM_THREADS=24 \
   MUSE_XCOMBINE_RA_OFFSETS="0.0000000,-5.4920943e-05,-0.00031381379" \
   MUSE_XCOMBINE_DEC_OFFSETS="0.0000000,8.3338105e-05,7.2223397e-05" \
   esorex --log-file=exp_combine.log muse_exp_combine \
   --lambdamin=6900. --lambdamax=7100. exp_combine.sof
```

The product files created by this command are:

```
1> ls -1 *.fits
DATACUBE_FINAL.fits
IMAGE_FOV_0001.fits
```

And finally, there are also other scenarios where combining the fully reduced pixel tables using the recipe **muse_exp_combine** is useful. For instance when science observations, which are accompanied by dedicated sky observations, should be combined. In this case each of science observations has to be calibrated separately using the corresponding sky observation. After this is done the fully reduced science observations can be combined with **muse_exp_combine**.

7 Tips & Tricks

7.1 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.[10] 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.

7.2 Verification Tools

Usually, the QC parameters as documented in Section 9 as well as the messages written to the terminal (and the log-file) by the recipe should give a good basis to verify that the processing worked as expected. But in some cases, a visual verification is necessary, and tools which help with the visualization and the verification of the results may be needed. A few such tools are shipped with the MUSE pipeline and get installed into `<installation prefix>/bin`. They are described in this section.

The visual tools use gnuplot[11] for plotting[12]. All tools mentioned here give a usage hint when called without parameters.

7.2.1 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:

```
1> 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:

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

[10] 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.

[11] Available from http://www.gnuplot.info/.

[12] 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.

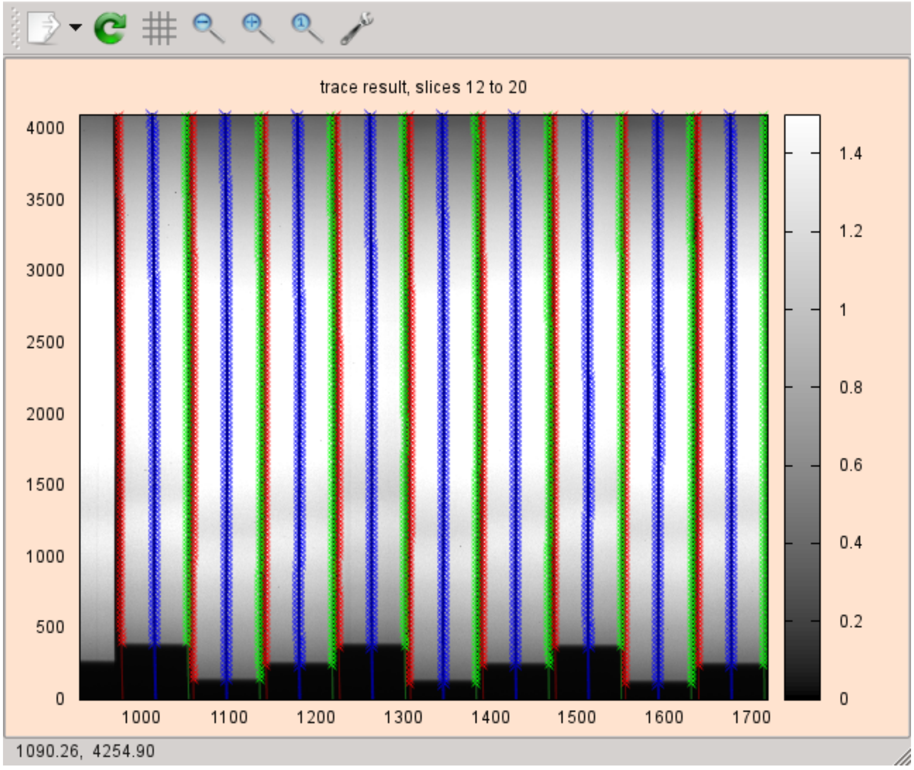

Figure 7.1: The graphical window showing the output of the `muse_trace_plot_samples` tool, then plotting slices 12 to 20 in IFU 6, using the trace samples table, the trace table, and the master flat-field image (see text for details).

Tip: when the default gnuplot setup is used (with the `x11`, `wxt`, or `qt` "terminals"), one can use the right mouse button on the plot window 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:

```
1> 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:

```
1> 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 Figure 7.1.

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 center of the CCD. The

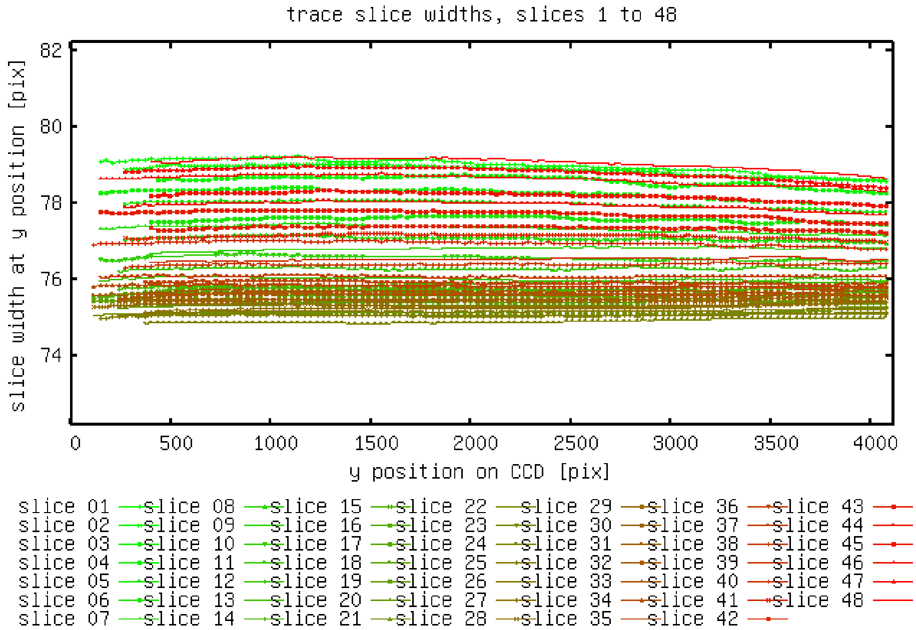

Figure 7.2: The graphical window showing the output of the `muse_trace_plot_widths` tool, plotting slices 1 to 48 of IFU 6, using the trace samples table (see text for details).

tool `muse_trace_plot_widths` was written to help to 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 Figure 7.2. 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).

7.2.2 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.

```
1> 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 Figure 7.3. 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:

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

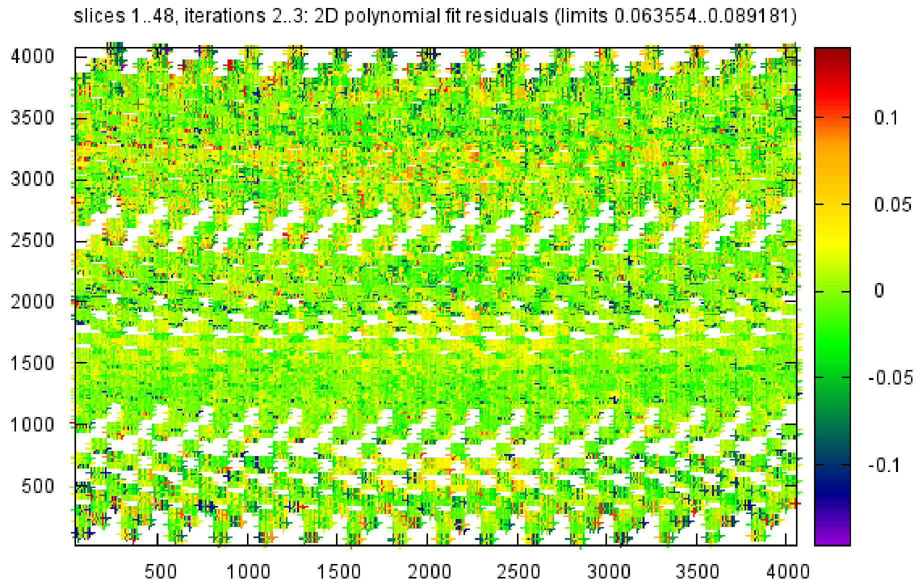

Figure 7.3: The graphical window showing the output of the `muse_wave_plot_residuals` tool, plotting all slices of IFU 10, using the wavelength calibration residuals table (see text for details).

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.

For a more in detail inspection of 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 use the tables of the same recipe 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`. Figure 7.4 shows the output of the command

```
1> muse_wave_plot_column -s 12 -r WAVECAL_TABLE-10.fits WAVECAL_RESIDUALS-10.fits
```

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 Ne I 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 the default parameters of **muse_wavecal** (`--fitweighting=cerrscatter`) the weak lines are hence weighted much less in the fit of the wavelength solution than the bright lines.

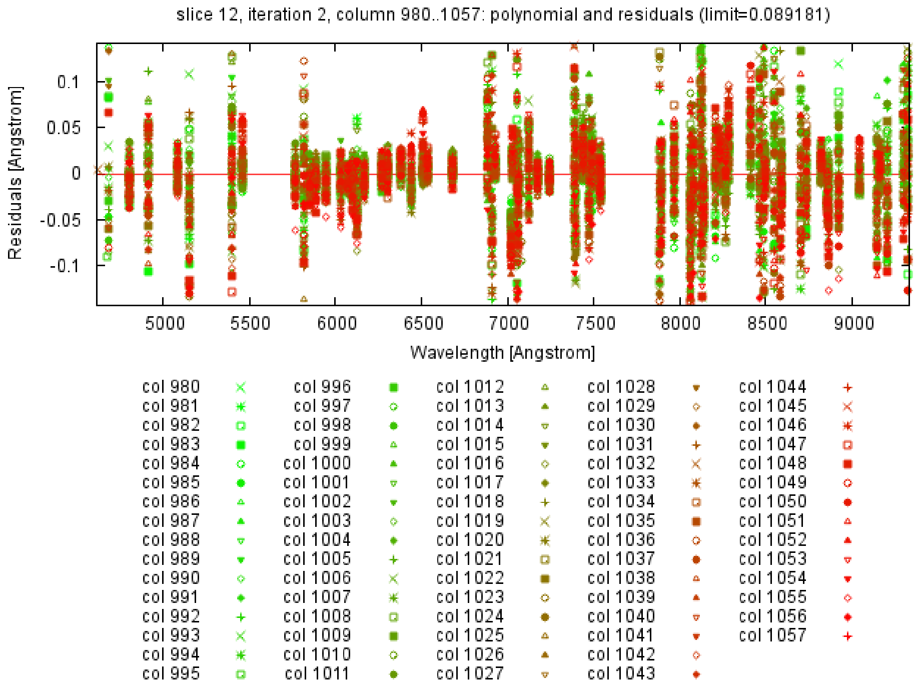

Figure 7.4: The graphical window showing the output of the `muse_wave_plot_column` tool, plotting slice 12 of IFU 10, using the wavelength calibration residuals and wavelength calibration tables (see text for details).

7.3 Miscellaneous Tools

7.3.1 Handling of MUSE pixel tables

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). Because of that a tool, `muse_pixtable_dump`, was added which decodes these values and shows them in a readable form. 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`:

```
1> muse_pixtable_dump -s 100000 -c 10 PIXTABLE_OBJECT_0001-01.fits
```

This command results in the output:

```
# MUSE pixel table "PIXTABLE_OBJECT_0001-01.fits", showing 10 rows starting at index 100000 of 13514329
# xpos          ypos             lambda     data        dq          stat          weight     exposure IFU xCCD yCCD xRaw yRaw slice
#          pix              pix  Angstrom        (flux)       flag   (flux**2)   ----------   No      No   pix  pix  pix  pix No
# flux    in [count]
# flux**2 in [count**2]
  -78.72976685  -141.46130371  6346.222   1.35237e+01 0x00000000 1.62160e+01  0.0000e+00    0       1   47 1438   79 1470   1
  -79.66138458  -141.48033142  6346.263   8.97605e+00 0x00000000 1.28346e+01  0.0000e+00    0       1   48 1438   80 1470   1
  -80.59300232  -141.49934387  6346.304   1.71657e+01 0x00000000 1.90208e+01  0.0000e+00    0       1   49 1438   81 1470   1
  -81.52462769  -141.51837158  6346.346   1.49865e+01 0x00000000 1.73776e+01  0.0000e+00    0       1   50 1438   82 1470   1
  -82.45624542  -141.53739929  6346.388   1.54953e+01 0x00000000 1.70677e+01  0.0000e+00    0       1   51 1438   83 1470   1
  -83.38786316  -141.55642700  6346.430   1.67681e+01 0x00000000 1.85787e+01  0.0000e+00    0       1   52 1438   84 1470   1
  -84.31948090  -141.57545471  6346.472   7.34210e+00 0x00000000 1.16375e+01  0.0000e+00    0       1   53 1438   85 1470   1
  -85.25110626  -141.59446716  6346.515   1.18395e+01 0x00000000 1.47541e+01  0.0000e+00    0       1   54 1438   86 1470   1
  -86.18272400  -141.61349487  6346.558   1.56828e+01 0x00000000 1.79335e+01  0.0000e+00    0       1   55 1438   87 1470   1
  -87.11434174  -141.63252258  6346.601   1.29650e+01 0x00000000 1.54910e+01  0.0000e+00    0       1   56 1438   88 1470   1
```

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:

```
1> 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 echos 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
MUSE pixel table "pt1-12_small.fits" (7383 rows) saved
```

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

```
1> muse_pixtable_erase_slice PIXTABLE_OBJECT_0005-14.fits 14 10 \
   PIXTABLE_OBJECT_0005-14_e10.fits
```

it removes slice 10 (numbering on the CCD) from a pixel table of IFU 14 as produced by **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:

```
1> muse_pixtable_erase_slice PIXTABLE_REDUCED_0001.fits 14 10 \
   PIXTABLE_REDUCED_0001_e1410.fits
```

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

7.3.2 Handling of MUSE bad pixel maps

Three tools exist to create or supplement MUSE bad pixel tables (the `BADPIX_TABLE` files), that are optionally read by every recipe that starts from raw data (see Section 6.4).

If one wants to create such a bad pixel table from scratch, one can start with one of the image-based master calibrations. These contain a `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 `muse_badpix_from_dq` tool:

```
1> muse_badpix_from_dq MASTER_FLAT-10.fits BADPIX_TABLE-10.fits
```

This would create a new table containing all bad pixels that were detected by the **muse_flat** recipe (including those that were present in all inputs used in that run of **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 `DQ` extension into an existing table:

```
1> muse_badpix_from_dq -i BADPIX_TABLE_in.fits MASTER_FLAT-10.fits \
   BADPIX_TABLE_out.fits
```

If one has manually recorded single bad pixels in an ASCII file or measured regions of bad pixels, one can use `muse_badpix_from_ascii` or `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:

```
1> muse_badpix_from_ascii bad_pixels.ascii 12 BADPIX_TABLE_12.fits
2> muse_badpix_from_region [10:12,100:2000] 256 12 BADPIX_TABLE_12.fits
```

`muse_badpix_from_region` requires the region to be in the format `[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 `-t` argument has to be set:

```
1> muse_badpix_from_ascii -t bad_pixels.ascii 12 BADPIX_TABLE_12.fits
2> muse_badpix_from_region -t [10:12,100:2000] 256 12 BADPIX_TABLE_12.fits
```

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

```
1> muse_badpix_from_ascii -i BADPIX_TABLE_existing.fits bad_pixels.ascii \
   12 BADPIX_TABLE_12.fits
2> muse_badpix_from_region -i BADPIX_TABLE_existing.fits [10:12,100:2000] 256 \
   12 BADPIX_TABLE_12.fits
```

8 Troubleshooting

8.1 Typical Problems

In many cases, the `ERROR` and `WARNING` messages of the MUSE pipeline alert the user of a problem with the data or the reduction. In the following, a few typical cases and solutions for them are described.

8.2 Failed Tracing

In some cases, vignetting of the slices on the edge of an IFU is severe enough to cause the tracing of the slices to fail when running the **muse_flat** recipe. If this happens, a typical error message would be

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

and 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 skyflat handling (**muse_skyflat**). Most critically, the science reduction (**muse_scibasic**) will stop when detecting such a broken trace table.

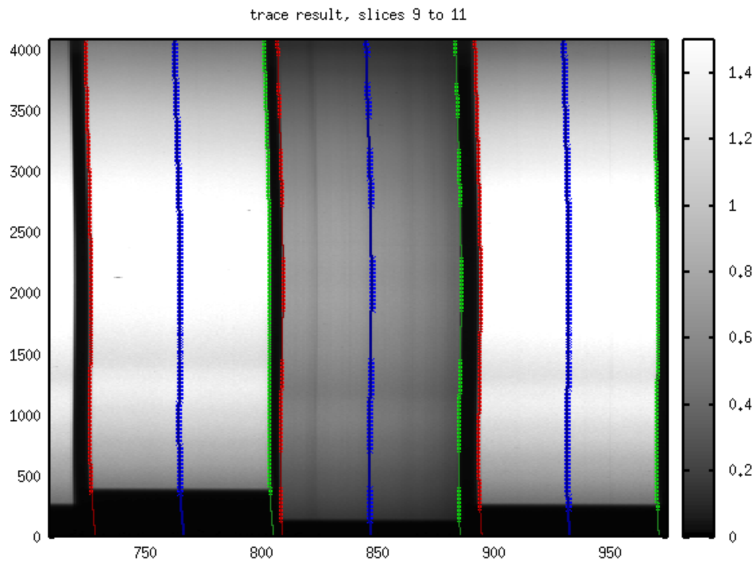

Figure 8.1: The graphical window showing the output of the `muse_trace_plot_samples` tool (see text for details).

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. Note that for too low values, 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

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

(see Figure 8.1) to visually verify that the trace table contains a good description of the slice location (see Section 7.2 for details).

Specifically for some of the data sets taken during the first MUSE Science Verification run the tracing of a few slices of IFU 6 may fail if the flat fields have been taken at temperatures lower than approximately 7 °C (cf. temperature given in the FITS keyword `INS.TEMP4.VAL` of the raw flat field frame). In order to recover from that, and finish the processing of the SV data sets, one can follow the recovery procedure shown here:

1. Run **muse_flat** on a set of flat fields taken at an ambient temperature of approximately 8 °C and create the master flat field and the trace tables. Data sets of flat fields taken at this temperature are available from the ESO archive.
   As an alternative a set of verified trace tables for IFU 6 is available from the pipeline download page at http://www.eso.org/sci/software/pipelines. The set contains the trace table for both, the nominal and the extended wavelength range.

2. Run **muse_flat** on the set of flat fields where the recipe fails to create all tracing solutions (which in general should be the flat fields closest in time and temperature with respect to the science observation). This will create 47 product files instead of the expected 48, since the trace table for IFU 6 has not been created.

3. Then, when running **muse_wavecal** on a set of arc-lamp frames and **muse_scibasic** on an on-sky observation (science, standard star, etc) use the 47 products from the previous step (low temperature flat fields), i.e. all 24 master flat fields and the 23 trace tables. To compensate for the missing trace table of IFU 6 one should use the trace table of IFU 6 created from the high temperature flat fields in the first step.

4. Since the the flux on the slices that are affected is very low, one may choose to remove them from pixel tables created by **muse_scibasic**, and thus from the final combined cube.
   To do so the tool `muse_pixtable_erase_slice` (cf. Section 7.2) is provided. The general syntax of the command is:

   ```
   muse_pixtable_erase_slice <pixtable-in> <ifu> <slice> <pixtable-out>
   ```

   For instance, to remove the most problematic slice of IFU 6, which is slice 10, one should run:

   ```
   1> muse_pixtable_erase_slice PIXTABLE_OBJECT_0001-06.fits 6 10 \
      PIXTABLE_OBJECT_0001-06_ERASED.fits
   ```

5. At this point one can continue as usual with running **muse_scipost**.

8.3 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.

Note that when the recipes runs with multiple threads the messages from a single thread on the screen (or in the logfile) may be interlaced with messages from other threads. The messages from a single thread can however be identified by the thread index number (e.g. `[tid=005]`) which is printed as part of the message.

The logfile should also be included in any bug report one is going to submit. In this case, if possible, the failing recipe should be re-run with the log level set to "`debug`".

8.4 Debugging Options

Certain environment variables for testing and debugging were created while developing the pipeline. Many of them might prove useful, if some data cannot be reduced with the usual options that are exposed through the recipe interface. Also, one might want to dig deeper into the analysis of what is done to the data during the reduction process.

As these are environment variables and not recipe options, they are set outside of the recipe call. In the bash shell one can simply add this as a prefix to the *EsoRex* command like:

```
1> MUSE_DEBUG_WAVECAL=1 esorex muse_wavecal wavecal.sof
```

which defines the environment variable only for this execution of *EsoRex*. To use it across several *EsoRex* executions (or your bash script) one can use the following prior to running the command

```
1> export MUSE_DEBUG_WAVECAL=1
```

A complete list of the environment variables can be found at the end of the `README` file, which, by default, can be found in `<`*`installation prefix`*`>/share/doc/esopipes/muse-X.Y.Z`. Remember that these variables are only exist to support recipe debugging. They should be used with caution, since they might generate a large number of files, output that one may be unfamiliar with, or cause unexpected side effects.

9 Recipe Reference

This section is essentially the reference manual of the MUSE pipeline recipes. In the following the documentation of the invididual pipeline recipes is provided, in terms of input data, recipe parameters, output products, and QC parameters created.

9.1 muse_bias

Combine several separate bias images into one master bias file.

9.1.1 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.

9.1.2 Input frames

Category	Type	Constraint (min)
BIAS	raw	required (3)
BADPIX_TABLE	calib	optional

9.1.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of image combination to use.
nlow	int	**1**	Number of minimum pixels to reject with minmax.
nhigh	int	**1**	Number of maximum pixels to reject with minmax.
nkeep	int	**1**	Number of pixels to keep with minmax.
lsigma	double	**3**	Low sigma for pixel rejection with sigclip.
hsigma	double	**3**	High sigma for pixel rejection with sigclip.
losigmabadpix	double	**30.**	Low sigma to find dark columns in the combined bias
hisigmabadpix	double	**3.**	High sigma to find bright columns in the combined bias

9.1.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_BIAS	Master bias

9.1.5 Quality control parameters

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

`QC.BIAS.INPUTi.NSATURATED` Number of saturated pixels in raw bias i in input list

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

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

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

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

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

`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

`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

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

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

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

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

9.2 muse_dark

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

9.2.1 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.

9.2.2 Input frames

Category	Type	Constraint (min)
DARK	raw	required (3)
MASTER_BIAS	calib	required (1)
BADPIX_TABLE	calib	optional

9.2.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of image combination to use.
nlow	int	**1**	Number of minimum pixels to reject with minmax.
nhigh	int	**1**	Number of maximum pixels to reject with minmax.
nkeep	int	**1**	Number of pixels to keep with minmax.
lsigma	double	**3**	Low sigma for pixel rejection with sigclip.
hsigma	double	**3**	High sigma for pixel rejection with sigclip.
scale	boolean	**true**	Scale the individual images to a common exposure time before combining them.
normalize	double	**3600.**	Normalize the master dark to this exposure time (in seconds). To disable normalization, set this to a negative value.
hotsigma	double	**5**	Sigma level, in terms of median deviation above the median dark level, above which a pixel is detected and marked as 'hot'.

9.2.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_DARK	Master dark

9.2.5 Quality control parameters

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

`QC.DARK.INPUTi.NSATURATED` Number of saturated pixels in raw dark i in input list
`QC.DARK.MASTER.NBADPIX` Number of bad pixels determined from master dark

`QC.DARK.MASTER.MEDIAN` Median value of the master dark
`QC.DARK.MASTER.MEAN` Mean value of the master dark
`QC.DARK.MASTER.STDEV` Standard deviation of the master dark
`QC.DARK.MASTER.MIN` Minimum value of the master dark
`QC.DARK.MASTER.MAX` Maximum value of the master dark
`QC.DARK.MASTER.DC` Dark current measured on master dark in randomly placed windows
`QC.DARK.MASTER.DCERR` Dark current error measured on master dark in randomly placed windows
`QC.DARK.MASTER.NSATURATED` Number of saturated pixels in output data

9.3 muse_flat

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

9.3.1 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.

9.3.2 Input frames

Category	Type	Constraint (min)
FLAT	raw	required (3)
MASTER_BIAS	calib	required (1)
MASTER_DARK	calib	optional, usually not used
BADPIX_TABLE	calib	optional

9.3.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of combination to use
nlow	int	**1**	Number of minimum pixels to reject with minmax
nhigh	int	**1**	Number of maximum pixels to reject with minmax
nkeep	int	**1**	Number of pixels to keep with minmax
lsigma	double	**3**	Low sigma for pixel rejection with sigclip
hsigma	double	**3**	High sigma for pixel rejection with sigclip
scale	boolean	**true**	Scale the individual images to a common exposure time before combining them.
normalize	boolean	**true**	Normalize the master flat to the average flux
trace	boolean	**true**	Trace the position of the slices on the master flat
nsum	int	**32**	Number of lines over which to average when tracing
order	int	**5**	Order of polynomial fit to the trace
edgefrac	double	**0.5**	Fractional change required to identify edge when tracing
losigmabadpix	double	**5.**	Low sigma to find dark pixels in the master flat

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
hisigmabadpix	double	**5.**	High sigma to find bright pixels in the master flat
samples	boolean	**false**	Create a table containing all tracing sample points.

9.3.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_FLAT	Master flat
TRACE_TABLE	Tracing table for all slices
TRACE_SAMPLES	Table containing all tracing sample points, if --samples=true

9.3.5 Quality control parameters

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

`QC.FLAT.INPUTi.MEDIAN` Median value of raw flat i in input list
`QC.FLAT.INPUTi.MEAN` Mean value of raw flat i in input list
`QC.FLAT.INPUTi.STDEV` Standard deviation of raw flat i in input list
`QC.FLAT.INPUTi.MIN` Minimum value of raw flat i in input list
`QC.FLAT.INPUTi.MAX` Maximum value of raw flat i in input list
`QC.FLAT.INPUTi.NSATURATED` Number of saturated pixels in raw flat i in input list
`QC.FLAT.MASTER.MEDIAN` Median value of the master flat before normalization
`QC.FLAT.MASTER.MEAN` Mean value of the master flat before normalization
`QC.FLAT.MASTER.STDEV` Standard deviation of the master flat before normalization
`QC.FLAT.MASTER.MIN` Minimum value of the master flat before normalization
`QC.FLAT.MASTER.MAX` Maximum value of the master flat before normalization
`QC.FLAT.MASTER.INTFLUX` Flux value, integrated over the whole master flat field before normalization
`QC.FLAT.MASTER.NSATURATED` Number of saturated pixels in output data
`QC.FLAT.MASTER.SLICEj.MEAN` Mean value around the vertical center of slice j before normalization
`QC.FLAT.MASTER.SLICEj.STDEV` Standard deviation around the vertical center of slice j before normalization
`QC.TRACE.SLICE_L.XPOS` Location of midpoint of leftmost slice
`QC.TRACE.SLICE_L.TILT` Tilt of leftmost slice, measured as angle from vertical direction
`QC.TRACE.SLICE_R.XPOS` Location of midpoint of rightmost slice
`QC.TRACE.SLICE_R.TILT` Tilt of rightmost slice, measured as angle from vertical direction
`QC.TRACE.SLICEj.MAXSLOPE` The maximum slope of the derived tracing functions of slice j within

the CCD.

`QC.TRACE.WIDTHS.MEDIAN` Median width of slices
`QC.TRACE.WIDTHS.MEAN` Mean width of slices
`QC.TRACE.WIDTHS.STDEV` Standard deviation of widths of slices
`QC.TRACE.WIDTHS.MIN` Minimum width of slices
`QC.TRACE.WIDTHS.MAX` Maximum width of slices
`QC.TRACE.GAPS.MEDIAN` Median of gaps between slices
`QC.TRACE.GAPS.MEAN` Mean of gaps between slices
`QC.TRACE.GAPS.STDEV` Standard deviation of gaps between slices
`QC.TRACE.GAPS.MIN` Minimum of gap between slices
`QC.TRACE.GAPS.MAX` Maximum gap between slices

9.4 muse_wavecal

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

9.4.1 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 --lsf or --resample are 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).

If --lsf is switched on, the line spread function is determined for each slice and saved in the LSF table for later use with sky subtraction.

9.4.2 Input frames

Category	Type	Constraint (min)
ARC	raw	required (1)
MASTER_BIAS	calib	required (1)
MASTER_DARK	calib	optional, usually not used
MASTER_FLAT	calib	optional, usually not used
TRACE_TABLE	calib	required (1)
LINE_CATALOG	calib	required (1)
BADPIX_TABLE	calib	optional
LSF_TABLE	calib	optional

9.4.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of lampwise image combination to use.
lampwise	boolean	**true**	Identify and measure the arc emission lines on images separately for each lamp setup.
sigma	double	**1.0**	Sigma level used to detect arc emission lines above the median background level in the S/N image of the central column of each slice
dres	double	**0.05**	The allowed range of resolutions for pattern matching (of detected arc lines to line list) in fractions relative to the expected value
tolerance	double	**0.1**	Tolerance for pattern matching (of detected arc lines to line list)
xorder	int	**2**	Order of the polynomial for the horizontal curvature within each slice
yorder	int	**6**	Order of the polynomial used to fit the dispersion relation

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
linesigma	double	**-1.0**	Sigma level for iterative rejection of deviant fits for each arc line within each slice, a negative value means to use the default (2.5).
residuals	boolean	**false**	Create a table containing residuals of the fits to the data of all arc lines. This is useful to assess the quality of the wavelength solution in detail.
fitsigma	double	**-1.0**	Sigma level for iterative rejection of deviant datapoints during the final polynomial wavelength solution within each slice, a negative value means to use the default (3.0).
fitweighting	string	**cerrscatter**, uniform, cerr, scatter, cerrscatter	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.
resample	boolean	**false**	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!
wavemap	boolean	**false**	Create a wavelength map of the input images
lsf	boolean	**false**	Fit the line spread function for each slice and save the result in the output LSF_TABLE.
lsf-residuals	boolean	**false**	Store the residual pixel table after the LSF fit.

9.4.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
WAVECAL_TABLE	Wavelength calibration table
WAVECAL_RESIDUALS	Fit residuals of all arc lines (if --residuals=true)
ARC_RED_LAMP	Reduced ARC image, per lamp
ARC_RED	Reduced master ARC image
ARC_RESAMPLED	Resampled arc images (if --resampled=true)
WAVE_MAP	Wavelength map of the input images
LSF_TABLE	Slice specific LSF parameters (if --lsf=true)
LSF_RESIDUALS	Subtracted pixel table after LSF fit (if --lsf=true and --lsf_residuals=true)

9.4.5 Quality control parameters

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

`QC.WAVECAL.SLICEj.LINES.NDET` Number of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.NID` Number of identified arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.PEAK.MEAN` Mean peak count level above background of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.PEAK.STDEV` Standard deviation of peak count level above background of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.PEAK.MIN` Peak count level above background of the faintest line in slice j

`QC.WAVECAL.SLICEj.LINES.PEAK.MAX` Peak count level above background of the brightest line in slice j

`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!

`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!

`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!

`QC.WAVECAL.SLICEj.LINES.FWHM.MEAN` Mean FWHM of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.FWHM.STDEV` Standard deviation of FWHM of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.FWHM.MIN` Minimum FWHM of detected arc lines in slice j

`QC.WAVECAL.SLICEj.LINES.FWHM.MAX` Maximum FWHM of detected arc lines in slice j

`QC.WAVECAL.SLICEj.RESOL` Mean spectral resolution R determined in slice j

`QC.WAVECAL.SLICEj.FIT.NLINES` Number of arc lines used in calibration solution fit in slice j

`QC.WAVECAL.SLICEj.FIT.RMS` RMS of the wavelength calibration fit in slice j

`QC.WAVECAL.SLICEj.DWLEN.BOTTOM` Difference in wavelength computed for the bottom left and bottom right corners of the slice on the CCD

`QC.WAVECAL.SLICEj.DWLEN.TOP` Difference in wavelength computed for the top left and top right corners of the slice on the CCD

`QC.WAVECAL.SLICEj.WLPOS` Position of wavelength given in WLEN in slice j

`QC.WAVECAL.SLICEj.WLEN` Wavelength associated to WLPOS in slice j

`QC.WAVECAL.INPUTi.NSATURATED` Number of saturated pixels in raw arc i in input list

`QC.WAVECAL.NSATURATED` Number of saturated pixels in output data

`QC.WAVECAL.LAMPl.INPUTi.NSATURATED` Number of saturated pixels in raw arc i of lamp l in input list

`QC.LSFl.AWAV` Wavelength of line l

`QC.IFUm.SLICEj.LSFl.WIDTH` LSF width in IFU i, slice j at the wavelength of line l

`QC.IFUm.SLICEj.LSFl.OFFSET` Wavelength calibration offset in IFU i, slice j at the wavelength of line l

`QC.IFUm.SLICEj.LSFl.FWHM` FWHM of the LSF in IFU i, slice j at the wavelength of line l

`QC.IFUm.SLICEj.WAVECAL.ERROR` Relative error of wavelength calibration of IFU i, slice j

9.5 muse_skyflat

Combine several separate (twilight) sky flat-fields into one master skyflat, and compute the integrated flux within the given wavelength range of the IFU.

9.5.1 Description

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, and combines the exposures using input parameters. Bad pixels are then interpolated from surrounding good ones before computing the integrated flux in all slices by summing the pixel values of all within the boundaries given by the trace table and the --lambdamin and --lambdamax parameters.

The integrated flux value is then later used as estimate for the relative throughput of each IFU, to scale values to a common flux level.

Since twilight sky flat-fields will have an illumination similar to that of science exposures, they should preferably be used as input to this recipe. Normal lamp flat-fields can be used, if twilight sky flats are not available.

The output image is not further used by the pipeline, only the FITS header with the integrated flux is propagated to the science data.

9.5.2 Input frames

Category	Type	Constraint (min)
SKYFLAT	raw	required (3)
MASTER_BIAS	calib	required (1)
MASTER_DARK	calib	optional, usually not used
BADPIX_TABLE	calib	optional
TRACE_TABLE	calib	required (1)
WAVECAL_TABLE	calib	required (1)

9.5.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of combination to use
nlow	int	**1**	Number of minimum pixels to reject with minmax
nhigh	int	**1**	Number of maximum pixels to reject with minmax
nkeep	int	**1**	Number of pixels to keep with minmax
lsigma	double	**3**	Low sigma for pixel rejection with sigclip
hsigma	double	**3**	High sigma for pixel rejection with sigclip
scale	boolean	**false**	Scale the individual images to a common exposure time before combining them.
lambdamin	double	**4800.**	Minimum wavelength to use for flux integration.
lambdamax	double	**9300.**	Maximum wavelength to use for flux integration.
normalize	boolean	**true**	Normalize the master skyflat to the flux integrated over the given wavelength range.

9.5.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_SKYFLAT	Master twilight skyflat

9.5.5 Quality control parameters

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

`QC.SKYFLAT.INPUTi.MEDIAN` Median value of raw flat i in input list
`QC.SKYFLAT.INPUTi.MEAN` Mean value of raw flat i in input list
`QC.SKYFLAT.INPUTi.STDEV` Standard deviation of raw flat i in input list
`QC.SKYFLAT.INPUTi.MIN` Minimum value of raw flat i in input list
`QC.SKYFLAT.INPUTi.MAX` Maximum value of raw flat i in input list
`QC.SKYFLAT.INPUTi.NSATURATED` Number of saturated pixels in raw flat i in input list
`QC.SKYFLAT.MASTER.MEDIAN` Median value of the master flat
`QC.SKYFLAT.MASTER.MEAN` Mean value of the master flat
`QC.SKYFLAT.MASTER.STDEV` Standard deviation of the master flat
`QC.SKYFLAT.MASTER.MIN` Minimum value of the master flat
`QC.SKYFLAT.MASTER.MAX` Maximum value of the master flat
`QC.SKYFLAT.MASTER.NSATURATED` Number of saturated pixels in output data
`QC.SKYFLAT.MASTER.INTFLUX` Flux integrated over all slices
`QC.SKYFLAT.MASTER.SLICEj.INTFLUX` Flux integrated over slice j

9.6 muse_geometry

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

9.6.1 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.

9.6.2 Input frames

Category	Type	Constraint (min)
MASK	raw	required (50)
MASTER_BIAS	calib	required (1)
MASTER_DARK	calib	optional, usually not used
MASTER_FLAT	calib	optional
TRACE_TABLE	calib	required (1)
WAVECAL_TABLE	calib	required (1)
LINE_CATALOG	calib	required (1)
BADPIX_TABLE	calib	optional
MASK_CHECK	calib	optional

9.6.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
ifu1	int	**1**	First IFU to analyze.
ifu2	int	**24**	Last IFU to analyze.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
sigma	double	**2.2**	Sigma detection level for spot detection, in terms of median deviation above the median.
centroid	string	**gaussian**, barycenter, gaussian	Type of centroiding and FWHM determination to use for all spot measurements: simple barycenter method or using a Gaussian fit.
lambdamin	double	**6800.**	When passing any MASK_CHECK frames in the input, use this lower wavelength cut before reconstructing the image.
lambdamax	double	**7200.**	When passing any MASK_CHECK frames in the input, use this upper wavelength cut before reconstructing the image.

9.6.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASK_REDUCED	Reduced pinhole mask images
MASK_COMBINED	Combined pinhole mask image
SPOTS_TABLE	Measurements of all detected spots on all input images.
GEOMETRY_TABLE	Relative positions of the slices in the field of view
GEOMETRY_CUBE	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.
GEOMETRY_CHECK	Optional field of view image to check the geometry calibration, integrated over the wavelength range given in the parameters.

9.6.5 Quality control parameters

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

`QC.GEO.EXPi.FWHM.MEAN` Average FWHM of all bright spots in exposure k.
`QC.GEO.EXPi.FWHM.MEDIAN` Median FWHM of all bright spots in exposure k.
`QC.GEO.EXPi.FWHM.STDEV` Standard deviation of FWHM of all bright spots in exposure k.
`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.
`QC.GEO.IFUm.WLENl` Nominal wavelength of arc line l.
`QC.GEO.IFUm.WLENl.FLUX.MEAN` Average integrated flux in all spots at reference wavelength l.
`QC.GEO.IFUm.WLENl.FLUX.MEDIAN` Median integrated flux in all spots at reference wavelength l.

`QC.GEO.IFUm.WLENl.FLUX.STDEV` Standard deviation of integrated flux in all spots at reference wavelength l.

`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.

9.7 muse_scibasic

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

9.7.1 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 sky-flat.

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. Both the reduced image and the pixel table are saved.

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.

9.7.2 Input frames

Category	Type	Constraint (min)
OBJECT	raw	required (1)
MASTER_BIAS	calib	required (1)
MASTER_DARK	calib	optional, usually not used
MASTER_FLAT	calib	required (1)
TRACE_TABLE	calib	required (1)
WAVECAL_TABLE	calib	required (1)
GEOMETRY_TABLE	calib	required (1)
MASTER_SKYFLAT	calib	optional
BADPIX_TABLE	calib	optional

Possible raw input frame categories are OBJECT, STD, SKY, ASTROMETRY, and REDUCED.

9.7.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
nifu	int	**0**	IFU to handle. If set to 0, all IFUs are processed serially. If set to -1, all IFUs are processed in parallel.
overscan	string	**vpoly**	If this is "none", stop when detecting discrepant overscan levels (see ovscsigma), for "offset" 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 "vpoly", a polynomial is fit to the vertical overscan and subtracted from the whole quadrant.
ovscreject	string	**dcr**	This influences how values are rejected when computing overscan statistics. Either no rejection at all ("none"), rejection using the DCR algorithm ("dcr"), or rejection using an iterative constant fit ("fit").
ovscsigma	double	**3.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than |ovscsigma x stdev|, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit. Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data region of the CCD that are ignored when computing statistics or fits.
crop	boolean	**true**	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).
cr	string	**none**, none, dcr	Type of cosmic ray cleaning to use (for quick-look data processing).
xbox	int	**15**	Search box size in x. Only used if cr=dcr.
ybox	int	**40**	Search box size in y. Only used if cr=dcr.
passes	int	**2**	Maximum number of cleaning passes. Only used if cr=dcr.
thres	double	**5.8**	Threshold for detection gap in factors of standard deviation. Only used if cr=dcr.
combine	string	**none**, none, average, median, minmax, sigclip	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 "none"!

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
nlow	int	**1**	Number of minimum pixels to reject with minmax
nhigh	int	**1**	Number of maximum pixels to reject with minmax
nkeep	int	**1**	Number of pixels to keep with minmax
lsigma	double	**3**	Low sigma for pixel rejection with sigclip
hsigma	double	**3**	High sigma for pixel rejection with sigclip
scale	boolean	**true**	Scale the individual images to a common exposure time before combining them.
saveimage	boolean	**true**	Save the pre-processed CCD-based image of each input exposure before it is transformed into a pixel table.
skylines	string	**5577.339,6300.304**	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.
skyhalfwidth	double	**5.**	Half-width of the extraction box (in Angstrom) around each sky emission line.
skybinsize	double	**0.1**	Size of the bins (in Angstrom per pixel) for the intermediate spectrum to do the Gaussian fit to each sky emission line.
resample	boolean	**false**	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!
dlambda	double	**1.25**	Wavelength step (in Angstrom per pixel) to use for resampling.

9.7.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
OBJECT_RED	Pre-processed CCD-based images (if --saveimage=true)
OBJECT_RESAMPLED	Resampled 2D image (if --resample=true)
PIXTABLE_OBJECT	Output pixel table

9.7.5 Quality control parameters

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

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

9.8 muse_standard

Create a flux response curve from a standard star exposure.

9.8.1 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. Use the brightest object as the standard star 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 cuve is then linearly extrapolated to the largest possible MUSE wavelength range and smoothed (with a 15 Angstrom median filter). 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.

9.8.2 Input frames

Category	Type	Constraint (min)
PIXTABLE_STD	raw	required (1)
EXTINCT_TABLE	calib	required (1)
STD_FLUX_TABLE	calib	required (1)
TELLURIC_REGIONS	calib	optional
FILTER_LIST	calib	optional

9.8.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
profile	string	**moffat**, gaussian, moffat, circle, square	Type of flux integration to use. "gaussian" and "moffat" use 2D profile fitting, circle and square are non-optimal flux integrators.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).
lambdamax	double	**10000.**	Cut off the data above this wavelength after loading the pixel table(s).

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	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.
darcheck	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.

9.8.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_STD	Reduced standard star field exposure
STD_RESPONSE	Response curve as derived from standard star(s)
STD_TELLURIC	Telluric absorption as derived from standard star(s)

9.8.5 Quality control parameters

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

`QC.STANDARD.POSk.X` Position of source k in x-direction in combined frame
`QC.STANDARD.POSk.Y` Position of source k in y-direction in combined frame
`QC.STANDARD.FWHMk.X` FWHM of source k in x-direction in combined frame
`QC.STANDARD.FWHMk.Y` FWHM of source k in y-direction in combined frame

9.9 muse_create_sky

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

9.9.1 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.

9.9.2 Input frames

Category	Type	Constraint (min)
PIXTABLE_SKY	raw	required (1)
EXTINCT_TABLE	calib	required(1)
STD_RESPONSE	calib	required(1)
STD_TELLURIC	calib	optional
SKY_LINES	calib	required (1)
SKY_CONTINUUM	calib	optional
LSF_TABLE	calib	optional
SKY_MASK	calib	optional

9.9.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
fraction	double	**0.05**	Fraction of the image 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.
sampling	double	**1.25**	Spectral sampling of the sky spectrum [Angstrom].
csampling	double	**20.**	Spectral sampling of the continuum spectrum [Angstrom].
cr	string	**cube**, none, cube, spectrum	Type of cosmic ray cleaning to use. "Cube" is the standard CR cleaning which works on a datacube, "spectrum" uses sigma clipping on the spectrum.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).
lambdamax	double	**10000.**	Cut off the data above this wavelength after loading the pixel table(s).

9.9.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
SKY_MASK	Created sky mask
IMAGE_FOV	Whitelight image used to create the sky mask
SKY_SPECTRUM	Sky spectrum within the sky mask
SKY_LINES	Estimated sky line flux table
SKY_CONTINUUM	Estimated continuum flux spectrum

9.9.5 Quality control parameters

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

`QC.SKY.THRESHOLD` Threshold in the white light considered as sky, used to create this mask
`QC.SKY.LINEl.NAME` Name of the strongest line in group k
`QC.SKY.LINEl.AWAV` Wavelength (air) of the strongest line of group k
`QC.SKY.LINEl.FLUX` Flux of the strongest line of group k
`QC.SKY.LINEl.OFFSET` Estimated wavelength offset for the strongest line of group k
`QC.SKY.CONT.FLUX` Total flux of the continuum
`QC.SKY.CONT.MAXDEV` Maximum (absolute value) of the derivative of the continuum spectrum

9.10 muse_astrometry

Compute an astrometric solution.

9.10.1 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.

9.10.2 Input frames

Category	Type	Constraint (min)
PIXTABLE_ASTROMETRY	raw	required (1)
ASTROMETRY_REFERENCE	calib	required (1)
EXTINCT_TABLE	calib	optional
STD_RESPONSE	calib	optional
STD_TELLURIC	calib	optional

9.10.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
centroid	string	**moffat**, gaussian, moffat, box	Centroiding method to use for objects in the field of view. "gaussian" and "moffat" use 2D fits to derive the centroid, "box" is a simple centroid in a square box.
detsigma	double	**3.**	Source detection sigma level to use.
radius	double	**5.**	Initial radius in pixels for pattern matching identification in the astrometric field.
faccuracy	double	**5.**	Factor of initial accuracy relative to mean positional accuracy of the measured positions to use for pattern matching.
niter	int	**2**	Number of iterations of the astrometric fit.
rejsigma	double	**3.**	Rejection sigma level of the astrometric fit.
rotcenter	string	**-0.01,-1.20**	Center of rotation of the instrument, given as two comma-separated floating point values in pixels.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).
lambdamax	double	**10000.**	Cut off the data above this wavelength after loading the pixel table(s).

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	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.
darcheck	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.

9.10.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_ASTROMETRY	Reduced astrometry field exposure
ASTROMETRY_WCS	Astrometric solution

9.10.5 Quality control parameters

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

`QC.ASTRO.POSk.X`	Position of source k in x-direction in combined frame
`QC.ASTRO.POSk.Y`	Position of source k in y-direction in combined frame
`QC.ASTRO.FWHMk.X`	FWHM of source k in x-direction in combined frame
`QC.ASTRO.FWHMk.Y`	FWHM of source k in y-direction in combined frame
`QC.ASTRO.NSTARS`	Number of stars identified for the astrometric solution
`QC.ASTRO.SCALE.X`	Computed scale in x-direction
`QC.ASTRO.SCALE.Y`	Computed scale in y-direction
`QC.ASTRO.ANGLE.X`	Computed angle in x-direction
`QC.ASTRO.ANGLE.Y`	Computed angle in y-direction
`QC.ASTRO.MEDRES.X`	Median residuals of astrometric fit in x-direction
`QC.ASTRO.MEDRES.Y`	Median residuals of astrometric fit in y-direction

9.11 muse_scipost

Prepare reduced and combined science products.

9.11.1 Description

Merge pixel tables from all IFUs, apply on-sky calibrations, and resample the data from all exposures into the final datacube.

9.11.2 Input frames

Category	Type	Constraint (min)
PIXTABLE_OBJECT	raw	required (1)
EXTINCT_TABLE	calib	required (1)
STD_RESPONSE	calib	required (1)
STD_TELLURIC	calib	optional
ASTROMETRY_WCS	calib	optional
FILTER_LIST	calib	optional
OUTPUT_WCS	calib	optional
SKY_LINES	calib	optional
SKY_CONTINUUM	calib	optional
LSF_TABLE	calib	optional
SKY_MASK	calib	optional

9.11.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
resample	string	**drizzle**, nearest, linear, quadratic, renka, drizzle, lanczos	The resampling technique to use for the final output cube.
dx	double	**0.0**	Horizontal step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2" for WFM, 0.075" for NFM, 1.0 if data is in pixel units.
dy	double	**0.0**	Vertical step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2" for WFM, 0.075" for NFM, 1.0 if data is in pixel units.
dlambda	double	**0.0**	Wavelength step size (in Angstrom). Natural instrument sampling is used, if this is 0.0

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
crtype	string	**median**, iraf, mean, median	Type of statistics used for detection of cosmic rays during final resampling. "iraf" uses the variance information, "mean" uses standard (mean/stdev) statistics, "median" uses median and median deviation statistics.
crsigma	double	**10.**	Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.
rc	double	**1.25**	Critical radius for the "renka" resampling method.
pixfrac	double	**0.8**	Pixel down-scaling factor for the "drizzle" resampling method.
ld	int	**1**	Number of adjacent pixels to take into account during resampling in all three directions (loop distance); this affects all resampling methods except "nearest".
format	string	**Cube**, Cube, Euro3D, xCube, xEuro3D	Type of output file format, "Cube" is a standard FITS cube with NAXIS=3 and multiple extensions (for data and variance). The extended "x" formats include the reconstructed image(s) in FITS image extensions within the same file.
weight	string	**exptime**, exptime, fwhm	Type of weighting scheme to use when combining multiple exposures.
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.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
skymethod	string	**model**, none, subtract-model, model, row-by-row	The method used to subtract the sky background. "model" should work in all cases, it uses a global sky spectrum model with a local LSF; "row-by-row" works on the basis of single slices and needs sufficient sky in each slice. If "model" is selected, calibration frames for SKY_LINES and LSF_TABLE must be set; SKY_CONTINUUM and SKY_MASK are optional. If "subtract-model" is selected, precalculated sky lines may be subtracted if specified by SKY_LINES, SKY_CONTINUUM, and LSF_TABLE.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).
lambdamax	double	**10000.**	Cut off the data above this wavelength after loading the pixel table(s).
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.
darcheck	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.
save_individual	boolean	**false**	If true, save fully reduced pixel table for each individual exposure as output product.
save_positioned	boolean	**false**	If true, save fully reduced and positioned pixel table for each individual exposure as output product. The difference to save_individual is that here, the output pixel tables have coordinates in RA and DEC. This is useful, if both the relative exposure weighting and the final resampling are to be done externally.
save_combined	boolean	**false**	If true, save the fully reduced and combined pixel table for the full set of exposures. The difference to save_positioned is that here 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.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
skymodel_fraction	double	**0.05**	Fraction of the image 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.
skymodel_sampling	double	**1.25**	Spectral sampling of the sky spectrum [Angstrom].
skymodel_csampling	double	**20.**	Spectral sampling of the continuum spectrum [Angstrom].
astrometry	boolean	**true**	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.
stacked	boolean	**false**	If true, write an additional output file in form of a 2D stacked image (x direction is pseudo-spatial, y direction is wavelength).

9.11.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_FINAL	Output datacube
IMAGE_FOV	Field-of-view images corresponding to the "filter" parameter.
OBJECT_RESAMPLED	Stacked image (if --stacked=true)
PIXTABLE_REDUCED	Fully reduced pixel tables for each exposure (if --save_individual=true)
PIXTABLE_POSITIONED	Fully reduced and positioned pixel table for each individual exposure (if --save_positioned=true)
PIXTABLE_COMBINED	Fully reduced and combined pixel table for the full set of exposures (if --save_combined=true)
SKY_MASK	Created sky mask
SKY_SPECTRUM	Sky spectrum within the sky mask

9.11.5 Quality control parameters

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

`QC.SCIPOST.POSk.X` Position of source k in x-direction in combined frame

`QC.SCIPOST.POSk.Y` Position of source k in y-direction in combined frame
`QC.SCIPOST.FWHMk.X` FWHM of source k in x-direction in combined frame
`QC.SCIPOST.FWHMk.Y` FWHM of source k in y-direction in combined frame
`QC.SKY.THRESHOLD` Threshold in the white light considered as sky, used to create this mask

9.12 muse_exp_combine

Combine several exposures into one datacube.

9.12.1 Description

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

9.12.2 Input frames

Category	Type	Constraint (min)
PIXTABLE_REDUCED	raw	required (2)
FILTER_LIST	calib	optional
OUTPUT_WCS	calib	optional

9.12.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
resample	string	**drizzle**, nearest, linear, quadratic, renka, drizzle, lanczos	The resampling technique to use for the final output cube.
dx	double	**0.0**	Horizontal step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2” for WFM, 0.075” for NFM, 1.0 if data is in pixel units.
dy	double	**0.0**	Vertical step size for resampling (in arcsec or pixel). The following defaults are taken when this value is set to 0.0: 0.2” for WFM, 0.075” for NFM, 1.0 if data is in pixel units.
dlambda	double	**0.0**	Wavelength step size (in Angstrom). Natural instrument sampling is used, if this is 0.0
crtype	string	**median**, iraf, mean, median	Type of statistics used for detection of cosmic rays during final resampling. "iraf" uses the variance information, "mean" uses standard (mean/stdev) statistics, "median" uses median and median deviation statistics.
crsigma	double	**8.**	Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.

Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
rc	double	**1.25**	Critical radius for the "renka" resampling method.
pixfrac	double	**0.6**	Pixel down-scaling factor for the "drizzle" resampling method.
ld	int	**1**	Number of adjacent pixels to take into account during resampling in all three directions (loop distance); this affects all resampling methods except "nearest".
format	string	**Cube**, Cube, Euro3D, xCube, xEuro3D	Type of output file format, "Cube" is a standard FITS cube with NAXIS=3 and multiple extensions (for data and variance). The extended "x" formats include the reconstructed image(s) in FITS image extensions within the same file.
weight	string	**exptime**, exptime, fwhm	Type of weighting scheme to use when combining multiple exposures.
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.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).
lambdamax	double	**10000.**	Cut off the data above this wavelength after loading the pixel table(s).
save_combined	boolean	**false**	If true, save the fully reduced and combined pixel table for the full set of exposures.

9.12.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_FINAL	Output datacube
IMAGE_FOV	Field-of-view images corresponding to the "filter" parameter.
PIXTABLE_COMBINED	Combined pixel table

9.12.5 Quality control parameters

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

`QC.EXPCOMB.POSk.X`	Position of source k in x-direction in combined frame
`QC.EXPCOMB.POSk.Y`	Position of source k in y-direction in combined frame
`QC.EXPCOMB.FWHMk.X`	FWHM of source k in x-direction in combined frame
`QC.EXPCOMB.FWHMk.Y`	FWHM of source k in y-direction in combined frame

A Data Formats

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. Science exposures may have 3 additional extensions if the MUSE SGS is used.

FITS extensions

- Primary extension (Extension 0, header only):
  contains the keywords specific to the exposure and common to all channels, such as information about the observing program, telescope, overall instrument and weather conditions, etc.
- Extensions 1 to 24:
  contains for each channel a short header (with information specific to the detector) and the image frame corresponding to this particular channel. The order of the extensions in the FITS file does not follow the order of the channel numbers so they should be addressed with their extension name `CHAN01`, `CHAN02`, `CHAN03`, ..., `CHAN22`, `CHAN23`, `CHAN24` corresponding to the channel number. The size of the image frame, in the absence of binning, is 4224 x 4240 (including the overscan regions).
- Extension 25 to 27 (for science exposures) contain information from the SGS taken in parallel to the science exposures, when SGS is activated. The SGS will record images with the TCCD and produce stack median images every approx. 2 min. These median images can be average over the entire science exposure to give a deeper image of the region surrounding the MUSE FOV.
  - Extension 25 (SGS_IMG):
    an image of size 1024×1024 contains the average of all the stacked median images taken during the science exposure.
  - Extension 26 (SGS_CUBE):
    a cube of $1024 \times 1024 \times N$ pixels, containing all N stacked median images taken during the science exposures.
  - Extension 27 (SGS_DATA):
    a FITS table containing information from the SGS system in the form of $(4 + NSTARS \times 10) \times N$ entries, for the N measurements done using $NSTARS$ ($NSTARS < 10$) stars detected in the SGS. For

each measurement, the 4 first columns give general information about the time and the offsets sent to the telescope, while the last $10 \times NSTARS$ entries give information on each star.

A.2 Static 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

Column name	Type	Description
`lambda`	double	Wavelength [Angstrom]
`flux`	double	Relative flux
`ion`	string	Ion from which the line originates
`quality`	int	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
`comment`	string	Optional comment

Frame tags

- **LINE_CATALOG**: `PRO.CATG=='LINE_CATALOG'`
  List of arc lines.

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

Column name	Type	Description
name	string	Line name
group	int	Line group id
lambda	double	Air wavelength [Angstrom]
flux	double	Line flux [10**(-20)*erg/(s cm^2 arcsec^2)]
dq	int	Quality of the entry (>0: dont use)

- 'OH_TRANSITIONS': FITS table, optional

Column name	Type	Description
name	string	Transition name; like "OH 8-3 P1e(22.5) 2"
lambda	double	Air wavelength [Angstrom]
v_u	int	Upper transition level
v_l	int	Lower transition level
nu	int	Vibrational momentum
E_u	double	Upper energy [J]
J_u	double	Upper momentum
A	double	Transition probability

Frame tags

- **SKY_LINES**: PRO.CATG=='SKY_LINES'
  Catalog of OH transitions and other sky lines,
  muse_create_sky: Estimated sky line flux table

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 contains 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

Column name	Type	Description
SourceID	string	Source identification
RA	double	Right ascension [deg]
DEC	double	Declination [deg]

Continued on next page

– continued from previous page		
Column name	Type	Description
`filter`	string	Filter name used for column mag
`mag`	double	Object (Vega) magnitude [mag]

Frame tags

- **ASTROMETRY_REFERENCE**: `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

Column name	Type	Description
`lambda`	double	Wavelength [Angstrom]
`extinction`	double	Extinction [mag/airmass]

Frame tags

- **EXTINCT_TABLE**: `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

Column name	Type	Description
`xpos`	int	X position of a bad pixel (on untrimmed raw data) [pix]
`ypos`	int	Y position of a bad pixel (on untrimmed raw data) [pix]
`status`	int	32bit bad pixel mask as defined by Euro3D
`value`	float	Extra value, e.g. depth for traps [count]

Frame tags

- **BADPIX_TABLE**: `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

Column name	Type	Description
`lambda`	double	Wavelength [Angstrom]
`flux`	double	The standard star flux [erg/s/cm**2/Angstrom]
`fluxerr`	double	Error of the standard star flux, optional (optional column) [erg/s/cm**2/Angstrom]

Frame tags

- **STD_FLUX_TABLE**: `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

Column name	Type	Description
lambda	double	Wavelength [Angstrom]
throughput	double	Filter throughput (in fractions of 1)

Frame tags

- **FILTER_LIST**: `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

Column name	Type	Description
lmin	double	Lower limit of the telluric region [Angstrom]
lmax	double	Upper limit of the telluric region [Angstrom]
bgmin	double	Lower limit of the background region [Angstrom]
bgmax	double	Upper limit of the background region [Angstrom]

Frame tags

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

A.3 Recipe Product Files

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 [RD5] for FITS files with data, bad pixel maps, and variance.

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
- **MASTER_SKYFLAT**:
  muse_skyflat: Master twilight skyflat
- **ARC_RED_LAMP**:
  muse_wavecal: Reduced ARC image, per lamp
- **ARC_RED**:
  muse_wavecal: Reduced master ARC image
- **MASK_REDUCED**:
  muse_geometry: Reduced pinhole mask images
- **MASK_COMBINED**:
  muse_geometry: Combined pinhole mask image

- **IMAGE_FOV**:
  muse_create_sky: Whitelight image used to create the sky mask,
  muse_scipost: Field-of-view images corresponding to the "filter" parameter.,
  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.

- **OBJECT_RED**:
  muse_scibasic: Pre-processed CCD-based images (if --saveimage=true)

- **OBJECT_RESAMPLED**:
  muse_scibasic: Resampled 2D image (if --resample=true),
  muse_scipost: Stacked image (if --stacked=true),
  muse_scipost_make_cube: Stacked image (if --stacked=true)

- **STD_RED**:
  muse_scibasic: Pre-processed CCD-based images (if --saveimage=true)

- **STD_RESAMPLED**:
  muse_scibasic: Resampled 2D image (if --resample=true)

- **SKY_RED**:
  muse_scibasic: Pre-processed CCD-based images (if --saveimage=true)

- **SKY_RESAMPLED**:
  muse_scibasic: Resampled 2D image (if --resample=true)

- **ASTROMETRY_RED**:
  muse_scibasic: Pre-processed CCD-based images (if --saveimage=true)

- **ASTROMETRY_RESAMPLED**:
  muse_scibasic: Resampled 2D image (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.

Pixel tables can be stored in files using a table format or an image based format. The different formats can be selected using the environment variable `MUSE_PIXTABLE_SAVE_AS_IMAGE`. The default format is the table format.

FITS extensions
The default table format (`MUSE_PIXTABLE_SAVE_AS_IMAGE` is not set, is:

- FITS table

Column name	Type	Description
`xpos`	float	x position of a pixel within the field of view [pix, rad, deg]
`ypos`	float	y position of a pixel within the field of view [pix, rad, deg]
`lambda`	float	Wavelength assigned to the pixel [Angstrom]
`data`	float	Data value [count, 1e-20 erg/(s cm2 Angstrom)]
`dq`	int	32bit bad pixel status as defined by Euro3D
`stat`	float	The data variance [count2, (1e-20 erg/(s cm2 Angstrom))2]
`origin`	int	Encoded value of IFU and slice number, as well as x and y position in the raw (trimmed) data
`weight`	float	The relative weight of this pixel (optional column)

If `MUSE_PIXTABLE_SAVE_AS_IMAGE=1` each of the table columns is stored as a separate 2D FITS image extension, where the name of the extension corresponds to the respective column name of the default table format. The image dimensions are $1 \times N_{\mathrm{row}}$, where N_{row} is the number of rows of the pixel table.

Frame tags

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

- **PIXTABLE_STD**:
  muse_scibasic: Output pixel table

- **PIXTABLE_SKY**:
  muse_scibasic: Output pixel table

- **PIXTABLE_ASTROMETRY**:
  muse_scibasic: Output pixel table

- **PIXTABLE_REDUCED**:
  muse_scibasic: Output pixel table,
  muse_scipost: Fully reduced pixel tables for each exposure (if --save_individual=true),
  muse_scipost_subtract_sky: Output pixel table(s) for sky subtraction.

- **PIXTABLE_POSITIONED**:
  muse_scipost: Fully reduced and positioned pixel table for each individual exposure (if --save_positioned=true)

- **PIXTABLE_COMBINED**:
  muse_scipost: Fully reduced and combined pixel table for the full set of exposures (if --save_combined=true),

muse_exp_combine: Combined pixel table,
muse_scipost_combine_pixtables: Combined pixel table

A.3.3 DATACUBE

Description

Three FITS NAXIS=3 cubes in three extensions for data values data quality and variance. Depending on recipe parameters, the data quality extension may be omitted; the bad pixel status is then encoded as NaN values in the data and variance extensions. Such datacubes follow the ESO specification [RD5] for FITS files with data, bad pixel maps, and variance.

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 [RD5].)

FITS extensions

- `'DATA'`: 3D FITS image (float)
  Data values
- `'DQ'`: 3D FITS image (int), optional
  Euro 3D data quality. This information is used to propagate information about bad pixels found f.e. in the processing of dark and flat-field exposures.
- `'STAT'`: 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,
  muse_scipost_make_cube: Output datacube

A.3.4 EURO3DCUBE

Description
Euro3D format. See Format Definition Document [RD6] for details.

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

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

- 'E3D_GRP': FITS table

Column name	Type	Description
GROUP_N	int	Group number
G_SHAPE	string	Spaxel shape keyword
G_SIZE1	float	Horizontal size per spaxel [arcsec]
G_ANGLE	float	Angle of spaxel on the sky [deg]
G_SIZE2	float	Vertical size per spaxel [arcsec]
G_POSWAV	float	Wavelength for which the WCS is valid [Angstrom]
G_AIRMAS	float	Airmass

Continued on next page

– continued from previous page

Column name	Type	Description
G_PARANG	float	Parallactic angle [deg]
G_PRESSU	float	Pressure [hPa]
G_TEMPER	float	Temperature [K]
G_HUMID	float	Humidity

Frame tags

- **DATACUBE_FINAL**:
  muse_scipost: Output datacube,
  muse_exp_combine: Output datacube,
  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

Column name	Type	Description
SliceNo	int	Slice number
Width	float	Average slice width
tc0_ij	double	polynomial coefficients for the central trace solution
MSE0	double	mean squared error of fit (central solution)
tc1_ij	double	polynomial coefficients for the left-edge trace solution
MSE1	double	mean squared error of fit (left-edge solution)
tc2_ij	double	polynomial coefficients for the right-edge trace solution
MSE2	double	mean squared error of fit (right-edge solution)

Frame tags

- **TRACE_TABLE**:
  muse_flat: Tracing table for all slices

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

Column name	Type	Description
`slice`	int	Slice number
`y`	float	y position on the CCD [pix]
`mid`	float	Midpoint of the slice at this y position [pix]
`left`	float	Left edge of the slice at this y position [pix]
`right`	float	Right edge of the slice at this y position [pix]

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

Column name	Type	Description
`SliceNo`	int	Slice number
`wlcIJ`	double	Polynomial coefficients for the wavelength solution
`MSE`	double	Mean squared error of fit

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

Column name	Type	Description
`slice`	int	Slice number
`iteration`	int	Iteration
`x`	int	x position on the CCD [pix]
`y`	int	y position on the CCD [pix]
`lambda`	float	Wavelength [Angstrom]
`residual`	double	Residual at this point [Angstrom]
`rejlimit`	double	Rejection limit for this iteration [Angstrom]

Frame tags

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

A.3.9 LSF_TABLE

Description
This file gives a parametrization of the line-spread function for each slice. It can contain data from one or more IFUs. It is a FITS table with at least 48 rows, one for each slice.

FITS extensions

- FITS table

Column name	Type	Description
`ifu`	int	IFU number
`slice`	int	Slice number within the IFU
`sensitivity`	double array	Detector sensitivity, relative to the reference
`offset`	double	Wavelength calibration offset
`refraction`	double	Relative refraction index
`slit_width`	double	Slit width
`bin_width`	double	Bin width

Continued on next page

– continued from previous page

Column name	Type	Description
lsf_width	double array	LSF gauss-hermitean width
hermit3	double array	3th order hermitean coefficient
hermit4	double array	4th order hermitean coefficient
hermit5	double array	5th order hermitean coefficient
hermit6	double array	6th order hermitean coefficient

Frame tags

- **LSF_TABLE**:
  muse_wavecal: Slice specific LSF parameters (if --lsf=true)

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 24x48 = 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

Column name	Type	Description
SubField	int	sub-field number
SliceCCD	int	Slice number on the CCD, counted from left to right
SliceSky	int	Slice number on the sky
x	double	x position within field of view [pix]
y	double	y position within field of view [pix]
angle	double	Rotation angle of slice [deg]
width	double	Width of slice within field of view [pix]
xerr	double	Error estimated of x position within field of view [pix]
yerr	double	Error estimated of y position within field of view [pix]
angleerr	double	Error estimate of rotation angle [deg]
sliceerr	double	Error estimate of slice width [pix]
stack	int	Slicer stack that this slice belongs to (optical numbering)
spot	int	Spot number in this slice
xrel	double	X offset of this spot relative to the slice center [mm]
xrelerr	double	Error of the relative x offset of this spot [mm]

Continued on next page

– continued from previous page

Column name	Type	Description
xc	double	X center of this spot on the CCD [pix]
yc	double	Y center of this spot on the CCD [pix]
dxl	double	Distance to left edge of slice on the CCD [pix]
dxr	double	Distance to right edge of slice on the CCD [pix]
dx	double	Pinhole distance in x on the CCD [pix]
dxerr	double	Error estimate of the pinhole distance in x on the CCD [pix]
vpos	double	(Averaged) vertical position of the mask [mm]
vposerr	double	Error estimated of the (averaged) vertical position of the mask [mm]
flux	double	Flux of the spot as integrated on the CCD image []
lambda	double	Wavelength [Angstrom]

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

Column name	Type	Description
filename	string	(Raw) filename from which this measurement originates
image	int	Number of the image in the series
POSENC2	int	X position of the mask in encoder steps
POSPOS2	double	X position of the mask [mm]
POSENC3	int	Y position of the mask in encoder steps
POSPOS3	double	Y position of the mask [mm]
POSENC4	int	Z position of the mask in encoder steps
POSPOS4	double	Z position of the mask [mm]
VPOS	double	Real vertical position of the mask [mm]

Continued on next page

– continued from previous page

Column name	Type	Description
ScaleFOV	double	Focus scale in VLT focal plane (from the FITS header) [arcsec/mm]
SubField	int	Sub-field number
SliceCCD	int	Slice number as counted on the CCD
lambda	double	Wavelength [Angstrom]
SpotNo	int	Number of this spot within the slice (1 is left, 2 is the central one, 3 is right within the slice)
xc	double	x center of this spot on the CCD [pix]
yc	double	y center of this spot on the CCD [pix]
xfwhm	double	FWHM in x-direction on the CCD [pix]
yfwhm	double	FWHM in y-direction on the CCD [pix]
flux	double	Flux of the spot as integrated on the CCD image
bg	double	Background level around the spot
dxcen	double	distance to center of slice at vertical position yc (positive: right of center) [pix]
twidth	double	trace width of the slice at the vertical CCD position of the spot [pix]

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

Column name	Type	Description
lambda	double	Wavelength [Angstrom]
flux	double	Flux [erg/(s cm^2 arcsec^2)]
fluxerr	double	Error of the flux (optional column) [erg/(s cm^2 arcsec^2)]

Frame tags

- **SKY_SPECTRUM**:

muse_create_sky: Sky spectrum within the sky mask,
muse_scipost: Sky spectrum within the sky mask

- **SKY_CONTINUUM**:
  muse_create_sky: Estimated continuum flux spectrum

A.3.13 STD_RESPONSE

Description
MUSE flux response table.

FITS extensions

- FITS table

Column name	Type	Description
`lambda`	double	wavelength [Angstrom]
`response`	double	instrument response derived from standard star [2.5*log10((count/s/Angstrom)/(erg/s/cm**2/Angstrom))]
`resperr`	double	instrument response error derived from standard star [2.5*log10((count/s/Angstrom)/(erg/s/cm**2/Angstrom))]

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

Column name	Type	Description
`lambda`	double	wavelength [Angstrom]
`ftelluric`	double	the telluric correction factor, normalized to an airmass of 1
`ftellerr`	double	the error of the telluric correction factor

Frame tags

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

B Benchmarks

B.1 The Reference System

The specifications of the reference system used to obtain the benchmark results are summarized in the following table:

CPU:	4 × Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20GHz (8 cores each)
CPU Cores:	64 (32 physical, 32 logical)
System Memory:	64 GB
Storage System:	5 TB, transfer rate ≈ 80 MB/s

B.2 Benchmark results

The benchmark results were obtained running the MUSE DRS recipes with a maximum number of threads as shown in the table, and with the threads pinned to the physical cores using the *Likwid* tools. The results are summarized in the following table. Note that these numbers are rough numbers and should just give an indication of what can be expected.

Recipe	Number of input frames	`OMP_NUM_THREADS`	Execution Time s	Peak Memory Usage GB (typical)
muse_bias	5	24	90	26
muse_flat	5	24	100	27
muse_wavecal	3	24	220	15
muse_skyflat	4	24	150	22
muse_scibasic	1	24	140	15
muse_standard	1	24	290	17
muse_create_sky	1	24	420	17
muse_astrometry	1	24	220	14
muse_scipost	1	24	380	18-25

C Performance Tools

This section summarizes the usage of two thread-pinning tools which can be used on Linux systems to set the CPU affinity of the threaded MUSE recipes. This means that the threads which run when a MUSE recipe is executed stay on the processor where they where started in first place.

This is done for two reasons: *(i)* to make sure that the threads are running on the physical cores only, since the logical cores are less performant, and *(ii)* to avoid that the threads stick to a processor to avoid that data has to be transferred through the machines memory subsustem, which in general has a quite limited bandwidth.

The first tool `taskset` is usually already available on recent Linux system, or the if it is missing it can be obtained by installing the package `util-linux` using the system package manager.

The second tool, *Likwid*, is actually a tool suite, and more flexible and convenient to use, but requires a manual installation. *Likwid* is what is used at ESO.

C.1 Using `taskset`

This command is usually available on modern Linux distribution. For example, to run the **muse_bias** recipe using `taskset` one has to execute the following command line:

```
1> taskset -c 0-5 esorex muse_bias --nifu=-1 bias.sof
```

This will run **muse_bias** on the first 6 CPUs of the machine. For a detailed description of the `taskset` command try:

```
1> man taskset
```

C.2 Using the Likwid Lightweight Performance Tools

This tool suite has been developed to support programmers in writing high performance multi-threaded code. As such it provides a larger set of tools, but as a user only two are really of interest.

As mentioned before, the tools have to be installed manually. This also requires editing the file `config.mk` if one does not have root privileges to do a system-wide installation. However this is straight forward by following the provided instructions in the file `INSTALL`[13].

Once the tool suite has been installed the usage is very similar to `taskset`. However, in addition to the actual thread-pinning tool it provides a second tool which can be used to query the topology of the computer. This first tool, `likwid-topology`, provides information on the machines architechture, thread configuration, cache sizes, and a lot more.

To get these information `likwid-topology` is simply executed:

```
1> likwid-topology
```

[13]Building the access demon, always requires root privileges, however this is not needed and the tools are fully usable without it.

this produces quite some output on the terminal. The information which is useful in this case is the number of threads per CPU core, and the list of CPU cores. In this list, the physical cores appear first.

The actual thread-pinning tool is `likwid-pin`, which is called, again using the **muse_bias** example, as shown here:

```
1> likwid-pin -c N:0-5 esorex muse_bias --nifu=-1 bias.sof
```

where the numbering of the CPUs refers to the numbering show in the output of `likwid-topology`. One advantage of `likwid-pin` is that it automatically sets the environment variable `OMP_NUM_THREADS` according to the list of CPUs specified on the command line, if `OMP_NUM_THREADS` was not already defined before.

For detailed information on the two commands please refer to their man-page.

D Useful links

This section compiles some useful links:

ESO MUSE web pages	http://www.eso.org/sci/facilities/paranal/instruments/muse.html
ESO Instrument Pipeline page	http://www.eso.org/sci/software/pipelines
ESO 3D Viewer	http://casa.nrao.edu/casa_obtaining.shtml
Likwid Lightweight Performance Tools	https://code.google.com/p/likwid
FITS image compression utilities	http://heasarc.nasa.gov/fitsio/fpack