European Organisation for Astronomical Research in the Southern Hemisphere

Programme: GEN

Project/WP: Science Operation Software Department

MUSE Pipeline User Manual

Document Number: ESO-264503

Document Version: 1

Document Type: Manual (MAN)

Released on: 2022-03-10

Document Classification: Public

Prepared by:	Ralf Palsa	
Validated by:	Sandra Castro	
Approved by:	Pascal Ballester	
	Name	

Authors

Name	Affiliation
MUSE Pipeline Team	ESO

Change Record from Previous Version

Affected Section(s)	Changes / Reason / Remarks
All	First, official version.

Contents

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 data cubes from raw MUSE data sets.

The processing of MUSE data is extremely 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 2.11.0. The current release has a few known issues, which are listed in Section 4.

The examples on running individual pipeline recipes in this manual use the `esorex` command and manually created list of input files. Several interfaces to automatically organise the data, create the list of input files and execute the pipeline recipes in the proper sequence are available, see *https://www.eso.org/pipelines* for details.

Please note that the use of Gasgano as a GUI for processing data is deprecated. Its use is no longer recommended and the related section in this manual, as well as support for Gasgano as a data processing GUI application in general will be dropped entirely in a future release.

1.2 Acknowledgements

This manual 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

[RD01]	MUSE User Manual	VLT–MAN–ESO–14670–1477
[RD02]	MUSE Calibration Plan	VLT–MAN–ESO–14670–0500
[RD03]	Gasgano User's Manual	VLT–PRO–ESO-19000–1932
[RD04]	FITS format description for pipeline products with data, error and data quality information	VLT–MAN–ESO–19500–5667
[RD05]	The Euro3D Data Format, Kissler-Patig, et al., Issue 1.2, May 2003	
[RD06]	3D Visualization Tool Manual	VLT–MAN–ESO–19500–5651
[RD07]	Reflex MUSE Tutorial	VLT–MAN–ESO–19540–6195
[RD08]	Reflex User Manual	VLT–MAN–ESO–19000–5037
[RD09]	"A MUSE map of the central Orion Nebula (M42)", Weilbacher, P., et al., 2015, A&A, 582, A114	
[RD10]	"Representations of celestial coordinates in FITS", Calabretta, M.R., Greisen, E.W., 2002, A&A, 395, 1077	
[RD11]	ESO Science Data Products Standard, Issue 5	GEN–SPE–ESO–33000–5335
[RD12]	ESO Science Data Products Standard Addendum, "Integral Field Specroscopy: 3D Data Cubes", 15-07-2015	
[RD13]	"ZAP — enhanced PCA sky subtraction for integral field spectroscopy", Soto, K. T., et al., 2016, MNRAS, 458, 3210	
[RD14]	"The Data Processing Pipeline for the MUSE Instrument", Weilbacher, P., et al., 2020, A&A, 641, A28	
[RD15]	"DAOPHOT: A Computer Program for Crowded-field Stellar Photometry", Stetson, P. B., 1987, PASP, 99, 191	
[RD16]	MUSE Python Data Analysis Framework, Online Documentation	

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
GALACSI	Ground Atmospheric Layer Adaptive Corrector for Spectroscopic Imaging
GCC	GNU Compiler Collection
GUI	Graphical User Interface
HDU	Header Data Unit
HST	Hubble Space Telescope
IFU	Integral Field Unit
LSF	Line Spread Function
MPDAF	MUSE Python Data Analysis Framework
MUSE	Multi Unit Spectroscopic Explorer
NaN	Not a Number
NFM	Narrow Field Mode
OpenMP	Open Multi-Processing
PAF	VLT parameter file format
PCA	Principle Component Analysis
pixel	picture element (of a raster image)
PSF	Point Spread Function
QC	Quality Control
SDP	Science Data Product
SGS	Slow Guiding 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.

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

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 seven basic calibration recipes and a pre-processing 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.
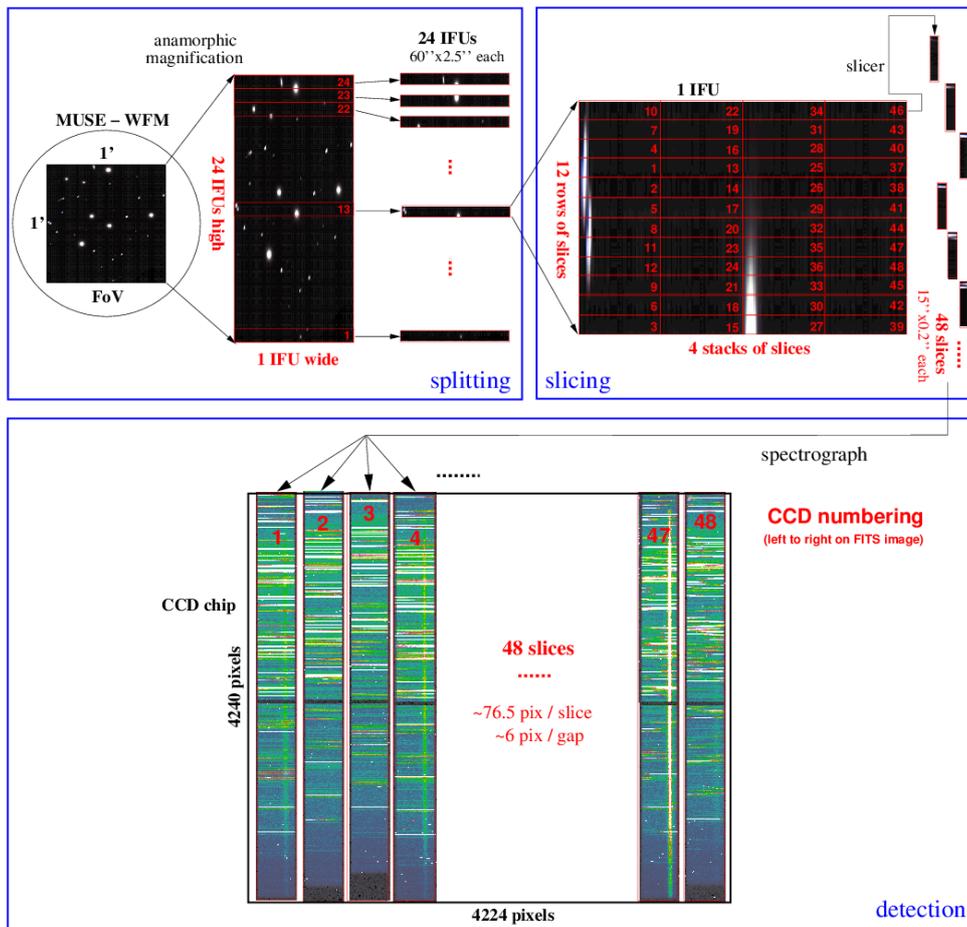
Figure 2.1: Graphical representations of the splitting and slicing procedures in the MUSE instrument. The example shown is for wide-field mode; narrow-field mode operates in the same way with a scaled-down field size. Note that the sizes given are approximate, the real data does not exactly cover a square region on the sky.
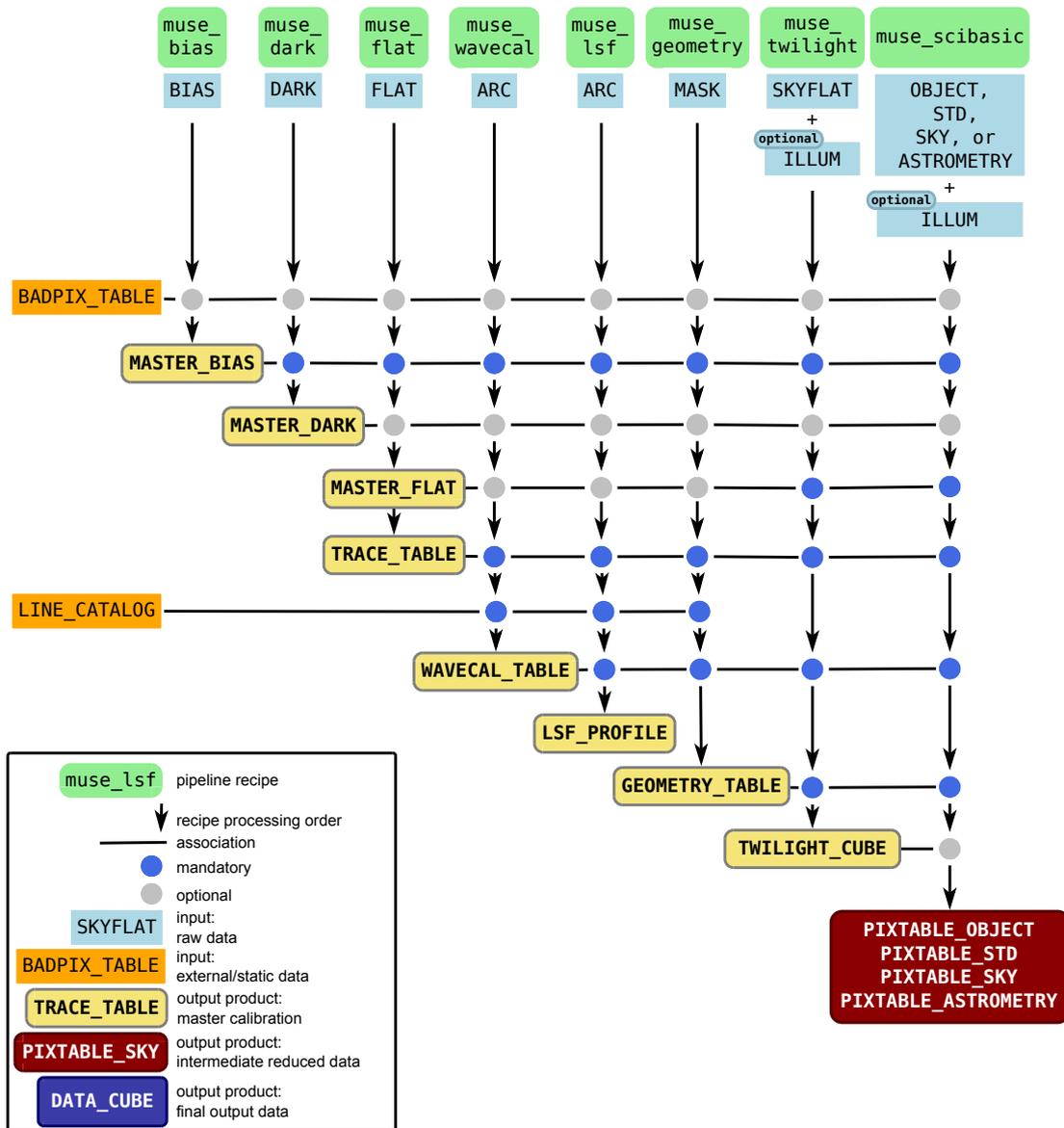
Figure 2.2: The first stage of the MUSE data reduction cascade. It shows the basic calibration recipes and the pre-processing recipe `muse_scibasic`, together with the "Association map" indicating the required and the optional input for each of the recipes.
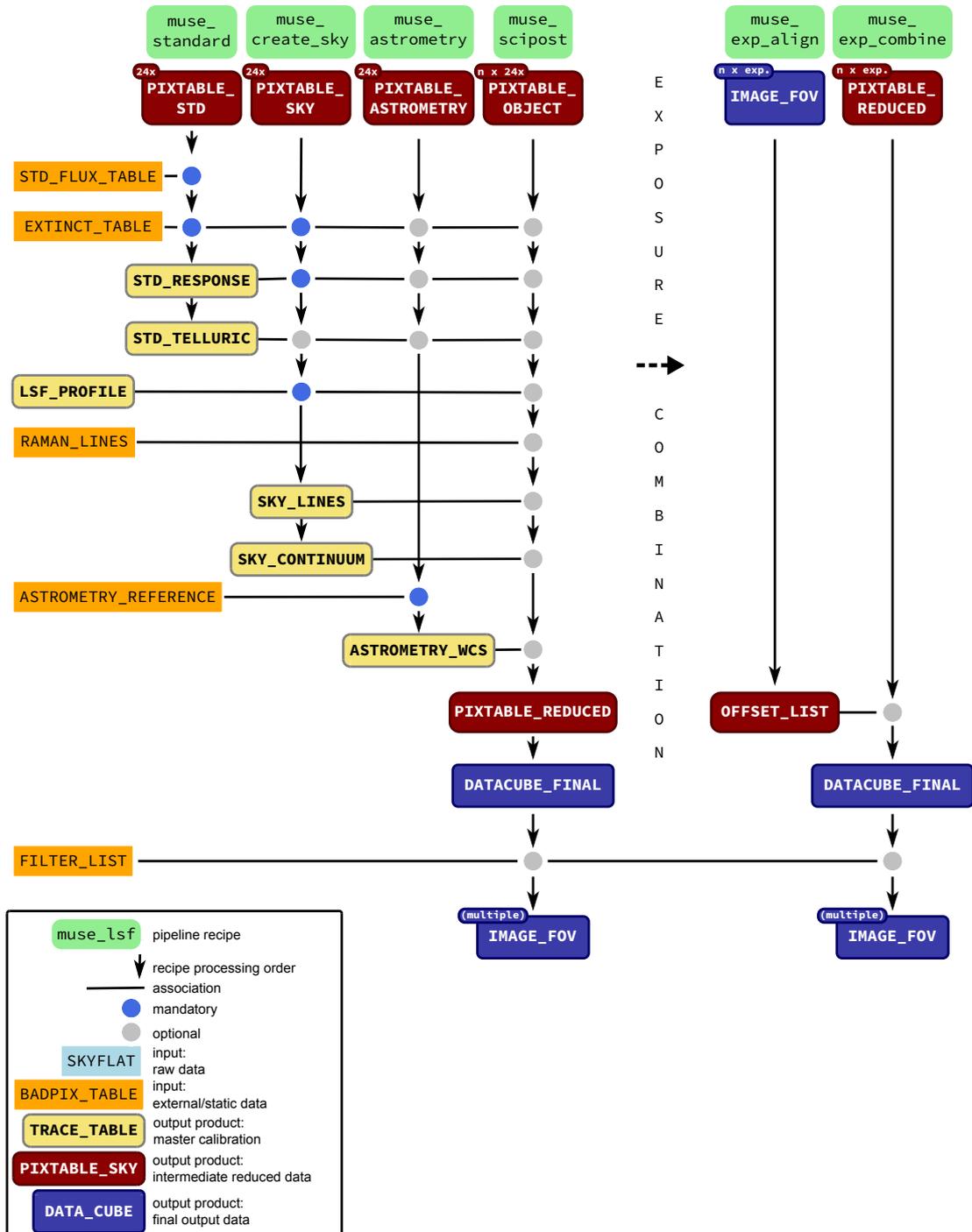
Figure 2.3: The second stage of the MUSE data reduction cascade. These recipes start from the output of the pre-processing recipe `muse_scibasic` and combine the data from all 24 IFUs into a single, fully reduced and calibrated data cube. The two recipes on the right side show the (optional) alignment of several exposures and their combination as an optional, final processing step. 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 foreseen 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 can be installed via several methods, depending on your OS, most of which facilitate easy installation, upgrade and removal. Please see the "ESO Data Reduction Pipelines and Workflow Systems" page (*https://www.eso.org/pipelines*).

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.

The easiest way to run the MUSE data reduction pipeline is to use the *Reflex* workflow that automatically classifies data, triggers all processing steps in the right order, and routes intermediate products correctly. *Reflex* calls individual reduction recipes via the *EsoRex* command. See [RD07] and [RD08] for details on how to use the MUSE *Reflex* workflow and *Reflex* itself. This manual describes how *EsoRex* can be run manually without the use of *Reflex*, and a description of individual recipes, recipe parameters, input files and output products.

ESO also provides *Gasgano*, a file browser that can be used to explore MUSE data sets. *Gasgano* can in principle be used to execute data reduction recipes, but this is not supported for MUSE.

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 pre-processing 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://github.com/RRZE-HPC/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 set-up 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 pre-processing recipe has no effect, and the data would be processed sequentially, one IFU after the other.

---

[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.

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 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 Pipeline is continuously improved. The release notes including the latest improvements and changes is always available at http://www.eso.org/sci/software/pipelines/muse. In addition this sections summarizes the most critical issues which are known at the time the release has been made.

Users processing MUSE data with the MUSE DRS version 2.11.0 should be aware of the following known issues:

- Starting with version 2.0 the format of the response pipeline product has changed. It is now corrected for the average flat field to allow for the computation of the overall system throughput in **muse_standard**. The average flat field spectrum is propagated as an additional extension of the pixel tables created by **muse_scibasic**, and it is applied to all on-sky data when the observation post-processing starts.

  As a consequence, a response product which was created by version 2.0, or newer, of the MUSE DRS can only be used together with pixel tables and a telluric correction table which were also created using the new DRS version. However, care has been taken that the new version of the DRS is backwards compatible, so that it correctly handles old format response products if it is used with pixel tables which were created by a version of the MUSE DRS prior to version 2.0, and also if pixel tables created by DRS version 2.0 or newer are used. In the latter case the flat field correction is not applied. Whether the data was corrected for the flat field spectrum is indicated in the final data cube by the FITS keyword `DRS.MUSE.FLUX.FFCORR`.

- For observations taken after the 10th of March 2017 the vignetting mask is not needed any more, since the vignetting which was usually present in the lower right corner of the MUSE field-of-view has been corrected at the instrument level. When processing data which was taken after the 10th of March the `VIGNETTING_MASK` should not be included in the *set-of-frames* file.

- The sky subtraction will leave artefacts. This may be an issue during the scientific analysis. A way to reduce the sky emission residuals by post-precessing the final data cube is outlined in section 7.3.6.

- The overscan correction method chosen in **muse_bias** using the option `--overscan` must be chosen consistently in subsequent recipes which perform the bias subtraction. Using different overscan correction methods in individual steps leads to wrong corrections and artefacts in the processed data. The pipeline will issue a warning in this case but the recipe will not fail.

Another issue, specifically affects data taken during the first MUSE Science Verification run (June 2014). For flat fields, which are taken at low ambient temperature (temperatures lower than $\approx 7\,°C$), 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 E 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 works only on the uncompressed files[2].

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. Science exposures will also contain extra FITS extensions which contain information from the MUSE Slow Guiding System (SGS), and the telescope Auto Guider (WFM observations), atmospheric turbulence information and the atmospheric turbulence profile (AO mode observations), and data from the Atmospheric Site Monitor (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`[3], `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[4], 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.

---

[2]Due to the format of the MUSE raw data files, it is actually possible to use the compressed files, if the suffix `.fz` is removed. However, in general (other instruments), this may not work!

[3]The term "channel" is equivalent to IFU and the two terms may be used interchangeably.

[4]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.

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.

`astrometry_reference.fits`	The catalogue 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.
`raman_lines.fits`	A list of Raman line transitions of O_2 and N_2 used to correct for the Raman emission line contamination in AO observations.
`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.
`vignetting_mask.fits`	It is a mask to correct for the vignetting which, usually, is present in the lower right corner of the MUSE field-of-view.[5]

For calibration files like the astrometric catalogue, 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 also contains a few master calibration files, which in principle can be created using the provided MUSE DRS recipes. These calibration files are the wide field mode (WFM) and narrow field mode (NFM) astrometric calibrations, the model of the MUSE line spread function (lsf-profiles) and the instrument response curve. The lsf-profiles and the instrument response curves are provided for observations with (*ao*) and without (*noao*) the GALACSI adaptive optics system, and for each of the two wavelength range settings, nominal (*n*) and extended (*e*). The available calibration files are summarized in the following table:

---

[5]The vignetting correction is not needed for observations taken after March, 10th 2017.

Astrometric calibrations:	`astrometry_wcs_wfm.fits`
	`astrometry_wcs_nfm.fits`
Line Spread Function models:	`lsf_profile_slow_wfm-noao-n.fits`
	`lsf_profile_slow_wfm-noao-e.fits`
	`lsf_profile_slow_wfm-ao-n.fits`
	`lsf_profile_slow_wfm-ao-e.fits`
	`lsf_profile_slow_nfm-ao-n.fits`
Instrument response curves:	`std_response_wfm-noao-n.fits`
	`std_response_wfm-noao-e.fits`
	`std_response_wfm-ao-n.fits`
	`std_response_wfm-ao-e.fits`
	`std_response_nfm-ao-n.fits`

These files are provided as a convenience since their creation is either very time consuming (lsf-profiles), where found to be stable in the analysis of the available commissioning data and/or may be used as a fall-back solution in view of the mentioned limitations of the current pipeline version (cf. Section 1.1). Currently there is one exception to this, which is the NFM astrometric calibration. Since it cannot be created reliably yet by the pipeline recipe **muse_astrometry** the provided static calibration should be used!

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.

For static calibrations (geometric calibration and astrometric solution) needed to process observations taken before December 1st, 2014 please refer to Appendix D.

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 [RD04]. 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. [RD05]).
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. For each CCD pixel of a single IFU the pixel table records the value of each pixel 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. Since version 1.0 of the MUSE DRS the default file format of the pixel tables is based on images in contrast to the table format used in previous releases. This change has been introduced for performance reasons[6].

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 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 … 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 files `lsf_profile_slow_wfm-n.fits` and `lsf_profile_slow_wfm-e.fits` which are distributed as part of the MUSE pipeline kit are stored as combined products. As of version 1.0 of

---

[6]In case it should be necessary, the old table format is still available and can be restored by setting the environment variable MUSE_PIXTABLE_SAVE_AS_TABLE=1

the MUSE DRS the recipe option "`--merge`" can be used to enable writing recipe products using the combined format.

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 abbreviated. 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 Julian 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)

Keyword Name	Description
Keywords describing the instrument/detector setup:	
EXPTIME	Total integration time [s]
INS.MODE	Instrument mode (field mode WFM/NFM, AO status, nominal/extended wavelength range)
INS.OPTI1.NAME	Wavelength range cut-off filter and notch filter setting: "Blue" or "Clear" for nominal and extended wavelength range without the notch filter (non-AO mode), and "Blue-Na2" or "Na" with the notch filter (AO mode).
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 [$^\circ$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 particular 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.

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_twilight	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
ILLUM	muse_scibasic, muse_twilight	CALIB	FLAT,LAMP,ILLUM	IFU

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

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

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

Once one has accumulated a substantial 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 cut-off filter in place) was selected[7]. 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 and the recipe products are written using the combined format, i.e. the recipe options "`--nifu=-1`" and "`--merge`" are used in the examples whenever possible. 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.

Note that all following examples use *bash* shell syntax. If a shell other than *bash* is used, the command lines may have to be adapted. In particular this may be true for setting environment variables on the same line prior to the command to be executed.

---

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

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
```

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 \
   --merge 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`[8] of the *EsoRex* run.

By default[9] 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 a single FITS file with 72 FITS extensions:

```
1> ls -1 *.fits
MASTER_BIAS.fits
```

Without the option "`--merge`" the recipe would have produced 24 FITS files with 3 FITS extensions each, where the IFU from which it has been created is indicated by the 2 digit sequence number at the end of the file name (e.g. the MASTER_BIAS of IFU 1 would be stored in the file `MASTER_BIAS-01.fits`).

By default all pre-processing recipes, including **muse_bias**, use the method `vpoly` for the overscan correction (option "`--overscan`"). However, if `offset` is selected as overscan correction method for creating the master bias, this chosen method must also be used in the subsequent reduction steps in order to apply the created master bias in a consistent way!

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.

---

[8]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.

[9]i.e. if *EsoRex* is executed with the option "`--suppress-prefix=true`"; which is the built-in default setting.
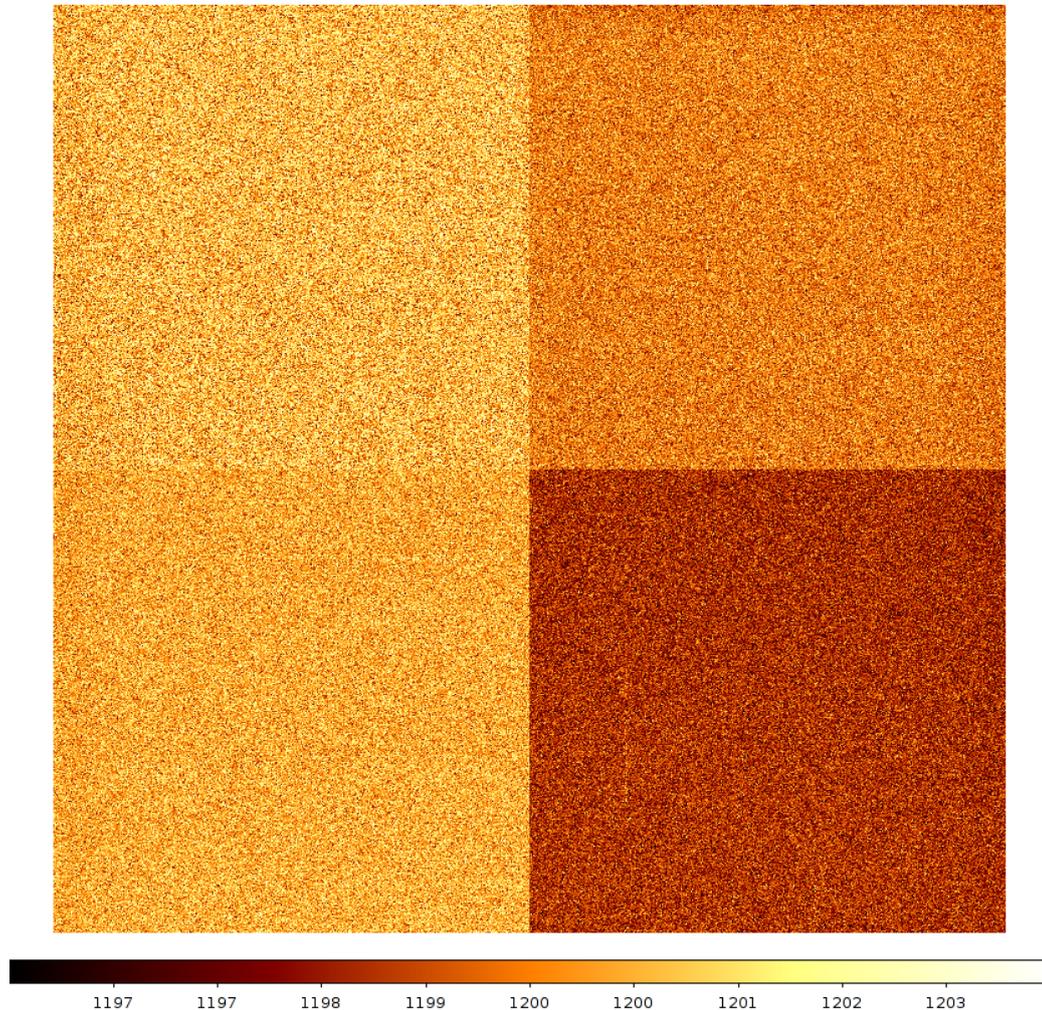
Figure 6.1: Example of a MUSE master bias of a single IFU (channel 1), as it is created by the recipe **muse_bias**.

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.fits MASTER_BIAS

2> OMP_NUM_THREADS=24 esorex --log-file=dark.log muse_dark --nifu=-1 \
   --merge dark.sof
```

The result of this *EsoRex* command is the master dark frame stored as a single FITS file with 72 FITS extensions. FITS files:

```
1> ls -1 *.fits
MASTER_DARK.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. The optional bad pixel table has been used in the following example. However note that the use of a bad pixel table may actually degrade the tracing solution if it contains bad columns, in particular 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.fits MASTER_BIAS
$MUSE_CAL/badpix_table.fits BADPIX_TABLE

2> OMP_NUM_THREADS=24 esorex --log-file=flat.log muse_flat --nifu=-1 \
   --merge flat.sof
```

The products of this *EsoRex* command are the master flat frame and the trace tables, each of them stored as single FITS files containing the master flat field and the tracing solution for all IFUs respectively:

```
1> ls -1 *.fits
MASTER_FLAT.fits
TRACE_TABLE.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.
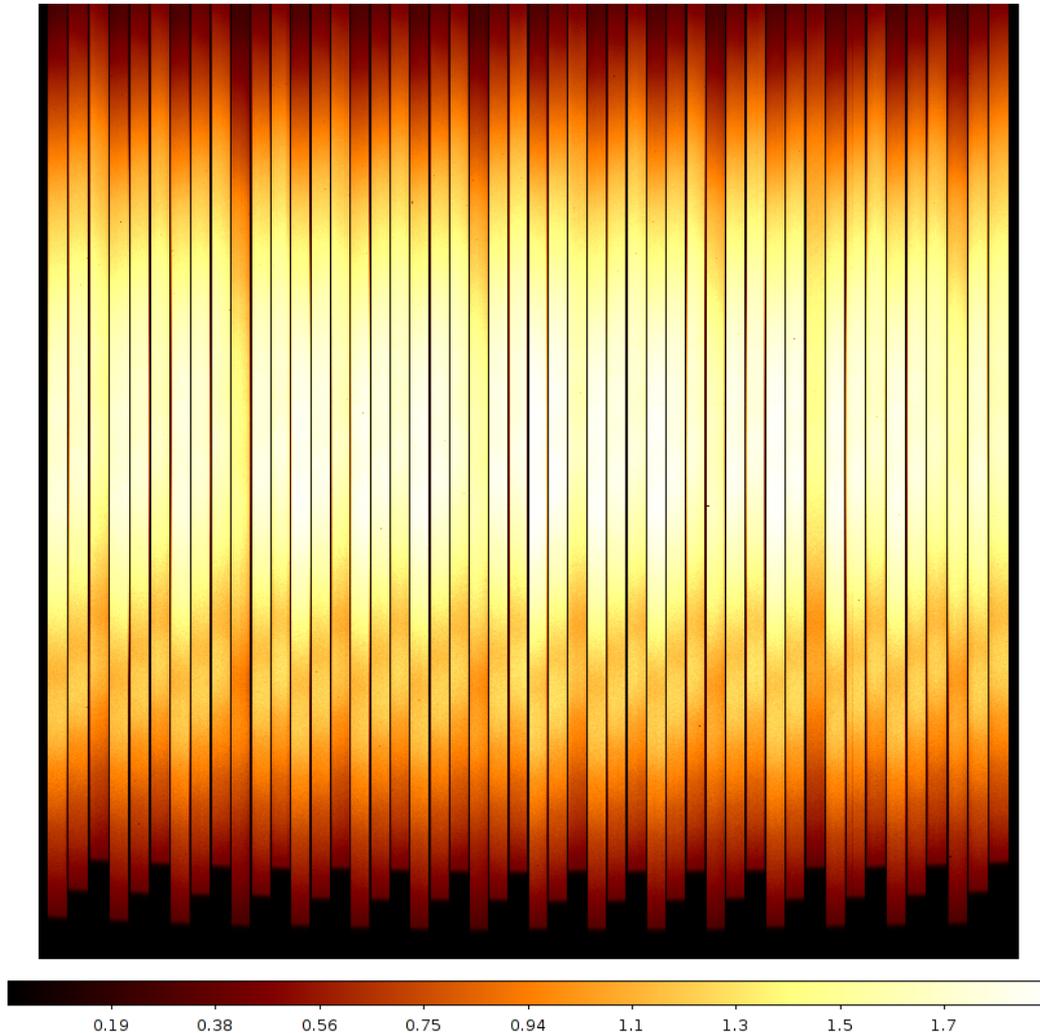
Figure 6.2: Example of a MUSE master flat field of a single IFU (channel 1), as it is created by the recipe **muse_flat**.

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.

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`[10]. 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.

---

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

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 catalogue is required. The *set-of-frames* and the *EsoRex* command line to create the 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.fits MASTER_BIAS
$MUSE_CAL/TRACE_TABLE.fits TRACE_TABLE
$MUSE_CAL/line_catalog.fits LINE_CATALOG

2> OMP_NUM_THREADS=24 esorex --log-file=wavecal.log muse_wavecal --nifu=-1 \
   --resample --residuals --merge wavecal.sof
```

The primary product created by this command is the wavelength solution. In addition to the primary product 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 Line Spread Function

For the subtraction of the sky background in the later steps of the processing of MUSE observations a representation of the MUSE line-spread function (LSF) is needed. The line-spread function can be calculated using the recipe **muse_lsf** which computes the slice and wavelength dependent LSF from a set of arc-lamp spectra. The recipe takes the same set of arc-lamp exposures as it is used with the **muse_wavecal** recipe. But, in order to measure the faint wings of the line-spread function with a reasonable signal-to-noise ratio at least 10 arc-lamp exposures for each arc-lamp should be used.

From the previous processing steps the recipe needs as input the wavelength solution, the trace table, and the master bias, and from the static calibration the line catalogue is needed. Optionally the bad pixel table, the master dark, and the master flat field may be added to the input set-of-frames. The following example shows the minimal contents of the set-of-frames and the *EsoRex* command to compute the line-spread function:

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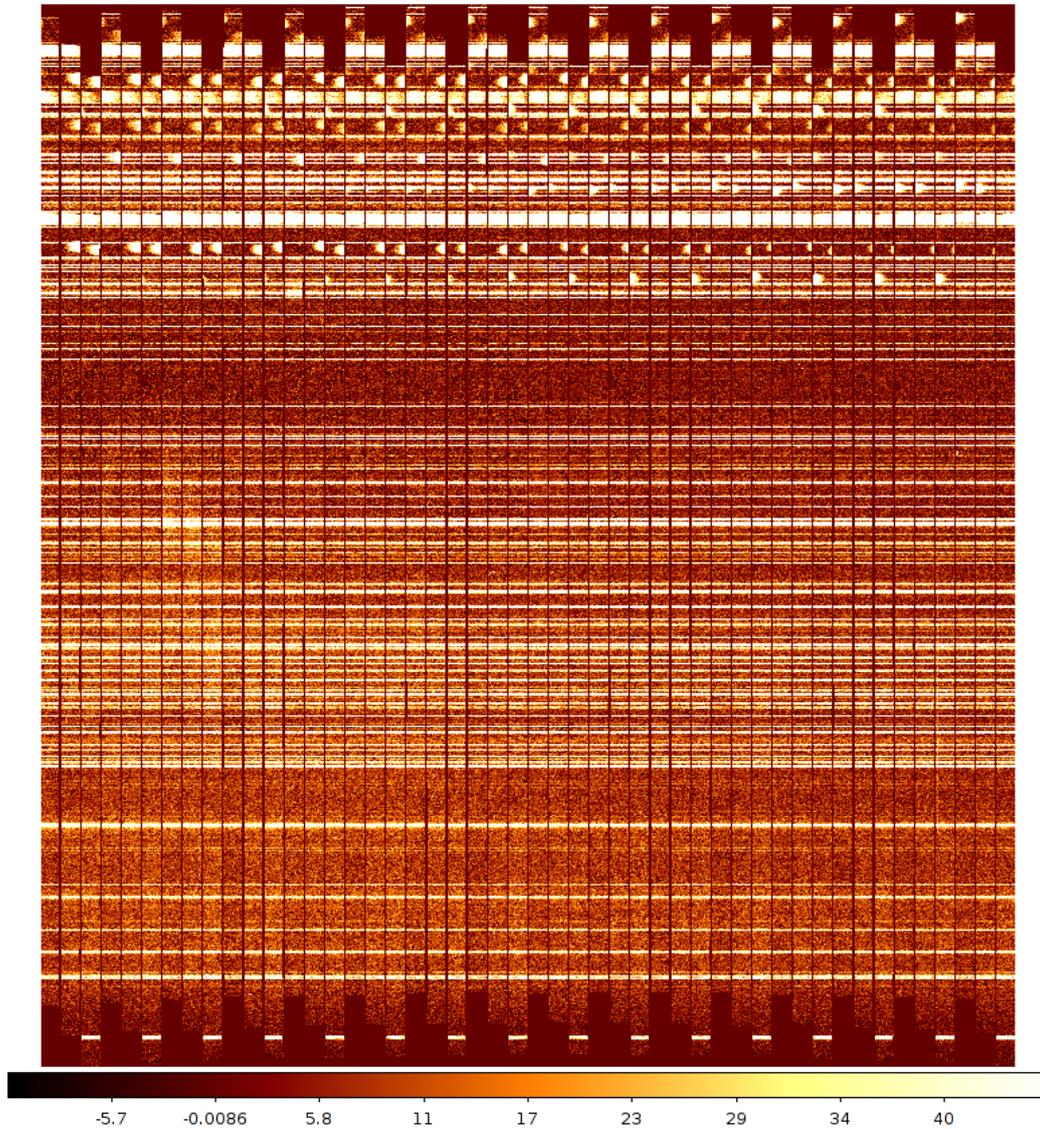
Figure 6.3: Example of a resampled arc-lamp frame of a single IFU (channel 1), as it is created by the recipe **muse_wavecal**. This image is the result of combining the three individual arc-lamp images and applying the determined wavelength solution. As it is expected from a good wavelength solution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

```
$MUSE_CAL/line_catalog.fits LINE_CATALOG

2> OMP_NUM_THREADS=24 esorex --log-file=lsf.log muse_lsf --nifu=-1 \
   --merge lsf.sof
```

The primary product of this *EsoRex* command is the lsf-profile, which is a stack of slice specific images of the line-spread function for each IFU.

Generating an lsf-profile is computational heavy. Using a standard set of 5 exposures per arc-lamp the *EsoRex* command shown before uses approximately 80 GB of memory and takes about 15 minutes. As an alternative to creating the lsf-profile with **muse_lsf**, the lsf-profiles that come 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, so that a frequent update of the lsf-profile is not necessary.

For the full description of the **muse_lsf** 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, an example and a description of the recipe **muse_geometry** is not shown here.

6.3.7 Illumination Correction

Twilight flats are part of the regular calibrations taken for MUSE. In this processing step the raw sky-flat frames are combined into a three-dimensional illumination correction using the recipe **muse_twilight**[11]. In addition the created twilight-cube carries the integrated flux value on to the further processing steps, 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 master flat field, the trace table, the wavelength solution and the geometry table. The *set-of-frames* and the *EsoRex* command for creating the master sky-flat are shown:

```
1> cat twilight.sof
/data/muse/raw/MUSE.2014-02-20T23:31:38.542.fits SKYFLAT
/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.fits MASTER_BIAS
$MUSE_CAL/MASTER_FLAT.fits MASTER_FLAT
$MUSE_CAL/TRACE_TABLE.fits TRACE_TABLE
$MUSE_CAL/WAVECAL_TABLE.fits WAVECAL_TABLE
$MUSE_CAL/geometry_table.fits GEOMETRY_TABLE
$MUSE_CAL/vignetting_mask.fits VIGNETTING_MASK

2> OMP_NUM_THREADS=24 esorex --log-file=twilight.log muse_twilight twilight.sof
```

The vignetting mask is an optional input to the recipe and, if provided, is used to correct vignetted regions in the MUSE field of view. The vignetting is present in the lower right corner of the MUSE field-of-view of

---

[11]Since version 1.0 the recipe **muse_twilight** replaces the recipe **muse_skyflat**

observations taken before March 11th 2017. For observation which were taken on March 11th 2017, or later, the vignetting mask is not needed any more, and should not be used, since the vignetting has been corrected at the instrument. An illumination flat field (frame tag `ILLUM`) may also be used as an additional input to correct for the relative illumination of the slices (see section 6.4 for details).

The primary product of this processing step is the twilight cube, which is a data cube of the smoothed twilight sky. As an additional product the data cube created from the combined twilight images is available. It is the data cube of the twilight sky before the smoothing is applied.

The recipe works on the data of all 24 IFUs simultaneously, in the same way as the post-processing recipes do. Thus the recipe options `--nifu` and `--merge` are not applicable to this recipe.

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

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. During this conversion sky lines are used to correct for offsets in the wavelength solution and, optionally, the data is corrected for the relative illumination of the slices for individual IFUs.

Pixel tables can be saved as FITS images or as a true FITS binary tables. To benefit from the more efficient way to read and write FITS images the results are, by default saved using the FITS image based format[12]. If the pixel tables should be stored a FITS binary tables, then this can be done by setting the environment variable `MUSE_PIXTABLE_SAVE_AS_TABLE=1`.

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.2014-08-04T09:34:48.604.fits OBJECT
/data/muse/raw/object/MUSE.2014-08-04T09:32:59.174.fits ILLUM
$MUSE_CAL/MASTER_BIAS.fits MASTER_BIAS
$MUSE_CAL/MASTER_FLAT.fits MASTER_FLAT
$MUSE_CAL/TWILIGHT_CUBE.fits TWILIGHT_CUBE
$MUSE_CAL/TRACE_TABLE.fits TRACE_TABLE
$MUSE_CAL/WAVECAL_TABLE.fits WAVECAL_TABLE
$MUSE_CAL/badpix_table.fits BADPIX_TABLE
$MUSE_CAL/geometry_table.fits GEOMETRY_TABLE
```

---

[12]With version 1.0 the default format has been changed to FITS images

```
2> OMP_NUM_THREADS=24 esorex --log-file=object.log muse_scibasic --nifu=-1 \
   --merge object.sof
```

Optionally the recipe **muse_scibasic** takes an illumination flat field (frame tag `ILLUM`), as shown in the example. This flat field, is a special **raw** continuum lamp exposure illuminating the whole field-of-view. It is taken regularly during the night, at a rate of 1 exposure every 1–2 hours, as part of the instrument calibration plan. It can be added to the *set-of-frames* as shown in the example. If it is present the recipe `muse_scibasic` corrects for the relative illumination of the slices on a per IFU basis (temperature dependent illumination changes).

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

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

Currently the option `-merge` affects only the products which are not pixel tables, i.e. the reduced images, which may be used for a visual check of the applied corrections. On the other hand these reduced images are usually not needed, and using the recipe option `--saveimage=FALSE` they are not created at all.

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 extinction table, and a standard flux table. Optionally a list of predefined telluric regions may be present in the input SOF (which is omitted in the following).

The standard flux table is a catalogue of reference spectra of spectrophotometric standard stars. The current version of this catalogue, 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 (bad resolution, should not be used!)
- GJ 754.1 A
- LDS 749B
- Feige 110
- LTT 7987
- EG 274
- LTT 3218
- WD 0308-565
- EG 21

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.

Several flux integration methods are available and can be chosen using the option `--profile`. The default is `--profile=auto`, which selects the Moffat profile fit with smoothing of the parameters along the wavelength axis for wide-field mode (WFM) observations, and uses the circular aperture method in the case of narrow-field (NFM) observations. Selecting the circular aperture flux integration may also be useful for WFM observations if artefacts are present in the response curve. In this case, using `--profile=circular` explicitly may result in less "wiggles" in the computed response curve.

The data cube can be visualized using the ESO 3D Visualization tool (cf. [RD06], Section E) 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.

By default the recipe selects the star which is closest to the expected position of the reference source as the spectro photometric standard. If this is inappropriate for the current observation, this can be changed using the recipe option `--select=flux`, which then takes the brightest object in the field.

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 observed object fills the field-of-view to such an extent that a reasonable sky spectrum cannot obtained directly on the observation itself. Otherwise, if the science exposure contains enough sky, the sky subtraction can be 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 requires an lsf-profile which provides the model of the slice and wavelength dependent MUSE line-spread function. This model is used to create a smooth sky continuum model by subtracting the fitted sky lines from the full sky spectrum. It is recommended use an lsf-profile which has been created during the basic reduction. As an alternative the lsf-profile included in the set of static calibrations can be used.

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_PROFILE.fits LSF_PROFILE
$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-profile is present in the input SOF. However, a reasonably smooth version of the sky continuum will only be obtained if an lsf-profile is 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 catalogue. The astrometric catalogue 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 catalogue includes 11 fields of the 6 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/astrometry_reference.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
```

It is necessary to *always* use an astrometric solution together with the geometry table that has been used to compute it. The reason for this is that the astrometric solution may compensate for small inaccuracies in the geometry table, so that a specific astrometric solution should always be used in pairs with its matching geometry table.

If this recipe is used to create an astrometric solution, one should make sure that the observation of the astrometric field and all necessary calibration exposures were taken close in time (and possibly temperature) to the raw data used for the creation of the geometry table[13]. Only then the created astrometric solution and the geometric calibration form a matching pair.

The pipeline distribution comes with a carefully verified WCS solution as a master calibration, which matches the geometry table that comes with the pipeline distribution. Also, since the WCS solution was found to be stable when analysing 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 recipe 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.

Note: The astrometric calibration to be used here should be created from the the same geometry table as the input pixel tables (see Section 6.5.3 for details).

---

[13]It is recommended that both, the geometry table and the astrometric solution are created using the same set of calibration data. In particular this applies to the trace table which is needed for the pre-processing of the astrometric field exposures

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 successfully.

```
1> cat scipost.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
$MUSE_CAL/LSF_PROFILE.fits LSF_PROFILE
$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.

By default the recipe performs a model based sky subtraction, i.e. initially a model of the sky lines and the sky continuum are computed using the darkest regions in the field-of view, and then these models are subtracted from the data. To use the models obtained from a previous run of **muse_create_sky** one can change the sky subtraction method using the option `--skymethod`. If this option is set to `subtract-model` the input models of the sky lines and the sky continuum are subtracted from the observation data as they are.

Self-Calibration

If an observation is dominated by empty sky regions, i.e. the sources cover only a small fraction of the field-of-view, it is possible to use these sky regions to improve the IFU-to-IFU and slice-to-slice flux variations. The method is called *self-calibration* or *autocalibration*. It uses the empty sky background to estimate flux-correction factors in each slice in several wavelength ranges and applies those correction factors after rejecting any outliers. The method cannot be used for fields which contain a big object or are filled by the target.

To use the *autocalibration* method and create additional recipe products to verify its performance the options `--autocalib=deepfield` and `--save=autocal` need to be added to the *EsoRex* command line. When using

the *autocalibration* one should also optimize the sky subtraction to use a larger fraction of the field as sky, as shown in the following example.

Since *autocalibration* benefits from a contiguous sky background definition, external data (e.g. HST images) can be used to better define the sky regions, by creating a SKY_MASK and pass it as input to **muse_scipost**. The mask is an image of integer pixels, where sky regions have a pixel value of 1, and object have a pixel value of 0. Using a gnomonic world coordinate system for the SKY_MASK it can be reused for multiple exposures of the same field. Please refer to Section A.3.12 for a detailed description of the SKY_MASK. In this case an OFFSET_LIST (see Sect. 6.6.1) should also be provided as recipe input to align the exposures with the external world coordinate system.

Thus, using the *autocalibration* method with the previous **muse_scipost** example, the input *set-of-frames* and the *EsoRex* command line would be:

```
1> cat scipost-autocalib.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
SKY_MASK.fits SKY_MASK
OFFSET_LIST.fits OFFSET_LIST
$MUSE_CAL/LSF_PROFILE.fits LSF_PROFILE
$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-autocalib.log \
   muse_scipost --filter=white,Johnson_V,Cousins_R,Cousins_I \
   --autocalib=deepfield --skymodel_fraction=0.75 --save=cube,autocal \
   scipost.sof
```

To verify that the autocalibration gave good results it is recommended to inspect the computed correction factors stored in the AUTOCAL_FACTORS product. These correction factors can be visualized using the function mpdaf.drs.plot_autocal_factors() of the MPDAF package (cf. [RD16]). Please refer to the MPDAF documentation for details.

If a large object is present in the observations and *autocalibration* is not directly possible, because some slices are completely covered by the large target, the following procedure may be used:

1. Run the recipe **muse_scipost** using *autocalibration* as shown in the previous example (i.e. using the option --autocalib=deepfield and saving the autocalibration products) on each exposure.

2. Combine the obtained AUTOCAL_FACTORS products of all exposures using a suitable rejection method to remove outliers and create a "typical" set of correction factors. To perform this combination the MPDAF function mpdaf.drs.merge_autocal_factors() (cf. [RD16]) could be used, for instance.

3. Run the recipe **muse_scipost** again, using the combined AUTOCAL_FACTORS product as recipe input, and using the option autocalib=user.

In some cases, but not always, this results in a smoother background without creating artefacts in objects, especially if the exposures were taken close in time, and the dithering caused the objects to significantly move within the MUSE field-of-view (e.g. within a single OB).

As it is described in the following section, observations using the adaptive optics are affected by Raman scattered light from the lasers. This contamination varies slowly across the field. Since the *autocalibration* assumes a uniform background across the field it is working better if the Raman scattered light contamination is corrected (cf. Section 6.5.4, p. 48) prior to applying the *autocalibration*. If both corrections are enabled the MUSE pipeline will therefore correct first for the Raman scattered light correction to get the best possible result.

Autocalibration is available in the MUSE pipeline since version 2.4. The implementation is based on the original implementation found in MPDAF.

Observations with Adaptive Optics

Observations using the GALACSI adaptive optics system are affected by Raman scattered light from the lasers, which enters the MUSE field. It shows up mainly as emission lines at 6485 Å and at 6827 Å, which vary slowly across the field, at about ± 5 %.

For observations of *nearly empty* fields, the MUSE pipeline provides a dedicated procedure to correct for this contamination. In order to use it, an additional, auxiliary calibration file containing the Raman line transition parameters has to be passed as input to the recipe **muse_scipost**. This calibration file, `raman_lines.fits`, is distributed as part of the MUSE pipeline package.

If the Raman line transition parameters are passed to **muse_scipost**, the light distribution around the Raman lines is computed, using only the sky part of the spectra. Therefore, this only works well if a large fraction of the field is sky background, and it will not work properly in cases where the target covers most of the field. Hence, a larger fraction of the field should be used as sky by setting the option `--skymodel_fraction` to a high value, as shown in the following example:

```
1> cat scipost-raman.sof
PIXTABLE_OBJECT_0001-01.fits PIXTABLE_OBJECT
...
PIXTABLE_OBJECT_0001-24.fits PIXTABLE_OBJECT
$MUSE_CAL/LSF_PROFILE.fits LSF_PROFILE
$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
$MUSE_CAL/raman_lines.fits RAMAN_LINES

2> OMP_NUM_THREADS=24 esorex --log-file=scipost-raman.log \
   muse_scipost --filter=white,Johnson_V,Cousins_R,Cousins_I \
   --skymodel_fraction=0.75 --save=cube,raman,skymodel \
   scipost.sof
```

In the previous example, the option `--save=raman` is used, to create field-of-view images at the Raman wave-

lengths. As it is the case for the *Autocalibration* also the Raman correction may give better results if an optimized sky mask and an offset list is provided as an additional input. However, their use is not strictly necessary. For details please refer to Section 6.5.4, page 46.

For exposures taken in narrow field mode (NFM) the observed object will take a significant part of the field of view. Therefore, the Raman correction is not possible and using the `RAMAN_LINES` calibration as input to **muse_scipost** is not recommended! In this case, the recipe will correct for these peaks as part of the sky continuum during the sky subtraction.

Final Remarks

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.[14] 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:

1. 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.

2. Provide offsets:

   The offsets can be passed to **muse_exp_combine**, and also **muse_scipost**, through the input SOF by providing an offset list (frame tag `OFFSET_LIST`) as an additional input.[15] An offset list is a FITS table which contains one row for each exposure to combine. Each row must specify the field offset (in decimal degrees) using the columns `RA_OFFSET` and `DEC_OFFSET`, and the observation date `DATE_OBS`. The latter is needed to identify the exposure to which the offset has to be applied. The offsets given in `RA_OFFSET` and `DEC_OFFSET` may be zero if no correction should be applied.

   The offsets are computed as the direct difference of the *measured* position to the *reference* position (no $\cos \delta$!), the values are interpreted in units of degrees:

   $$RA\_OFFSET = RA_{measured} - RA_{reference}$$
   $$DEC\_OFFSET = DEC_{measured} - DEC_{reference}$$

   As of version 1.2 of the MUSE DRS the recipe **muse_exp_align** is provided to automatically compute field offsets, if the exposures overlap significantly (e.g. the set of exposures originating from a dither pattern sequence). Using the field-of-view images (`IMAGE_FOV`) created by **muse_scipost** for each exposure, the recipe creates an offset list which can be used as input for the recipe **muse_exp_combine**. For isolated fields, i.e. fields which do not overlap with any other field, the recipe **muse_exp_align** will set the offsets to zero. Thus, in the case of combining fields which do not overlap the offset list must be created by other means.

   The offsets which were applied to each exposure are recorded in the FITS header of the output data cube in the set of `DRS.MUSE.OFFSETi` FITS keywords. They can be used to verify the applied field coordinate offsets.[16]

---

[14]This is due to a slight decentering of the axis of the derotator, leading to a "derotator wobble".

[15]Prior to version 1.2 of the MUSE Pipeline the coordinate offsets had to be provided by setting the environment variables `MUSE_XCOMBINE_RA_OFFSETS` and `MUSE_XCOMBINE_DEC_OFFSETS`. Since version 1.2 these environment variables are no longer supported.

[16]It is also possible to verify the applied offsets using the messages written, in *debug*-mode, by the pipeline recipes **muse_scipost** and **muse_exp_combine** to the terminal and/or the log-file. However, the default `INFO` message only gives the approximate final offsets relative to the first exposure in the sequence.

Since the coordinate offsets which are applied to an exposure are chosen from the `OFFSET_LIST` using the `DATE_OBS` time-stamp, it is also possible to apply an offset to a single exposure by passing the `OFFSET_LIST` for a full set of exposures to **muse_scipost** (using a single exposure in input). This may be useful for testing purposes as well.

6.6.2 Correcting relative fluxes

In case an object was observed in non-photometric conditions and exposures need to be adapted relative to each other or to an absolute flux measurement, the column `FLUX_SCALE` in the offset list table can be used to specify scaling factors different from unity for the individual exposures. As for the spatial offsets, the observation date is used to associate a scaling factor with a particular exposure (cf. section 6.6.1). Both recipes **muse_scipost** and **muse_exp_combine** take the scaling factors provided by an offset list into account, and apply them to both, the exposure data and its associated variance data.

Any applied flux scaling can be verified using two groups of FITS header keywords, `DRS.MUSE.FLUX.SCALEi` and `DRS.MUSE.OFFSETi.DATE-OBS`, in the output data cube. The latter keyword can be used to uniquely identify the exposure to which the scaling was applied.

6.6.3 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.

6.6.4 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) and an offset list providing the corrections of the field coordinates for each input exposure. Apart from the additional pixel tables and the offset list the *set-of-frames* is identical to the one shown previously for processing a single exposure.

The *EsoRex* command line to create a combined data cube is shown in the following example, where, in addition, the wavelength range is restricted to the interval 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
OFFSET_LIST.fits OFFSET_LIST
$MUSE_CAL/LSF_PROFILE.fits LSF_PROFILE
$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_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.

For this example it was assumed that the offset list has been created manually using some FITS editor. In the following section an example is shown where the offset list is created using the utility recipe **muse_exp_align**.

6.6.5 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_PROFILE.fits LSF_PROFILE
$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 Pipeline User Manual

Doc. Number:		ESO-264503
Doc. Version:		1
Released on:		2022-03-10
Page:		53 of 159

```
$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 re-sampling 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 re-sampling step.

As in the previous section, where the exposures were combined using **muse_scipost**, an offset list is needed as input to **muse_exp_combine** to correct for the relative field offsets of the individual input exposures. This time however, instead of manually creating an offset list it is created using the recipe **muse_exp_align**.

The input for this recipe are the field-of-view images obtained by processing the individual exposures with **muse_scipost**. The command to compute the field offsets is shown here:

```
1> cat exp_align.sof
IMAGE_FOV_0001_EXP01.fits IMAGE_FOV
IMAGE_FOV_0001_EXP02.fits IMAGE_FOV
IMAGE_FOV_0001_EXP03.fits IMAGE_FOV

2> esorex --log-file=exp_align.log muse_exp_align exp_align.sof
```

The product files created by this command are:

```
1> ls -1 *.fits
OFFSET_LIST.fits
PREVIEW_FOV.fits
```

```
SOURCE_LIST_0001.fits
SOURCE_LIST_0002.fits
SOURCE_LIST_0003.fits
```

The created offset list contains the corrections of the field in the columns `RA_OFFSET` and `DEC_OFFSET` in units of degrees. The exposure to which these corrections have to be applied is identified by the observation date stored in the column `DATE_OBS`. The offset list also contains an empty column `FLUX_SCALE`. This additional column can be used to manually set flux scaling factors for the individual exposures, which would be taken into account during the combination. An offset list created by **muse_exp_align** can directly be used as input to the recipe **muse_exp_combine**, if no further editing is necessary.

In addition to the offset list, the recipe **muse_exp_align** creates also one list of detected sources for each input field-of-view image. These source lists can be used to validate the set of sources which is used to calculate the field offsets. Source detections which are closer than a minimum distance to a bad pixel in the corresponding field-of-view image are not taken into account when the field offsets are calculated (see Section 9.13 for details). If necessary, the source lists can also be used as ancillary input to **muse_exp_align**, which allows a further, possibly iterative, selection of the source positions from which the field offsets will be computed. If source lists are to be used as input frames, there must be one source list for each input field-of-view. In addition, the source list and its corresponding field-of-view image must have the same values of `MJD-OBS` and `DATE-OBS` appearing in the primary FITS header. Finally, when executing **muse_exp_align** the source detection must be disabled using the expert level parameter `--override_detection=true`.

Since version 2.8 of the MUSE pipeline the **muse_exp_align** recipe also creates a preview image of the combined field-of-view, based on the computed field offsets. This allows an immediate, visual verification of the computed offsets before the actual, possibly time consuming, exposure combination is done using **muse_exp_combine**.

The recipe **muse_exp_align** is meant to be used for exposures which have significant overlap, since it determines the offsets of the field coordinates by comparing the positions of sources in the overlapping regions. If isolated fields, i.e. fields which do not have an overlap with any other field in the input set, are present in the input set their offsets are set to zero, and no offset correction will be applied.

Now, to get to the same final data cube as in the previous section, but using the output of **muse_exp_align**, 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
OFFSET_LIST.fits OFFSET_LIST
$MUSE_CAL/filter_list.fits FILTER_LIST

2> OMP_NUM_THREADS=24 \
   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**.

MUSE Pipeline User Manual

Doc. Number:	ESO-264503
Doc. Version:	1
Released on:	2022-03-10
Page:	56 of 159

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 beneficial 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.[17] 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[18] for plotting[19]. 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
```

Tip: when the default gnuplot set-up 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.

---

[17]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.

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

[19]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.
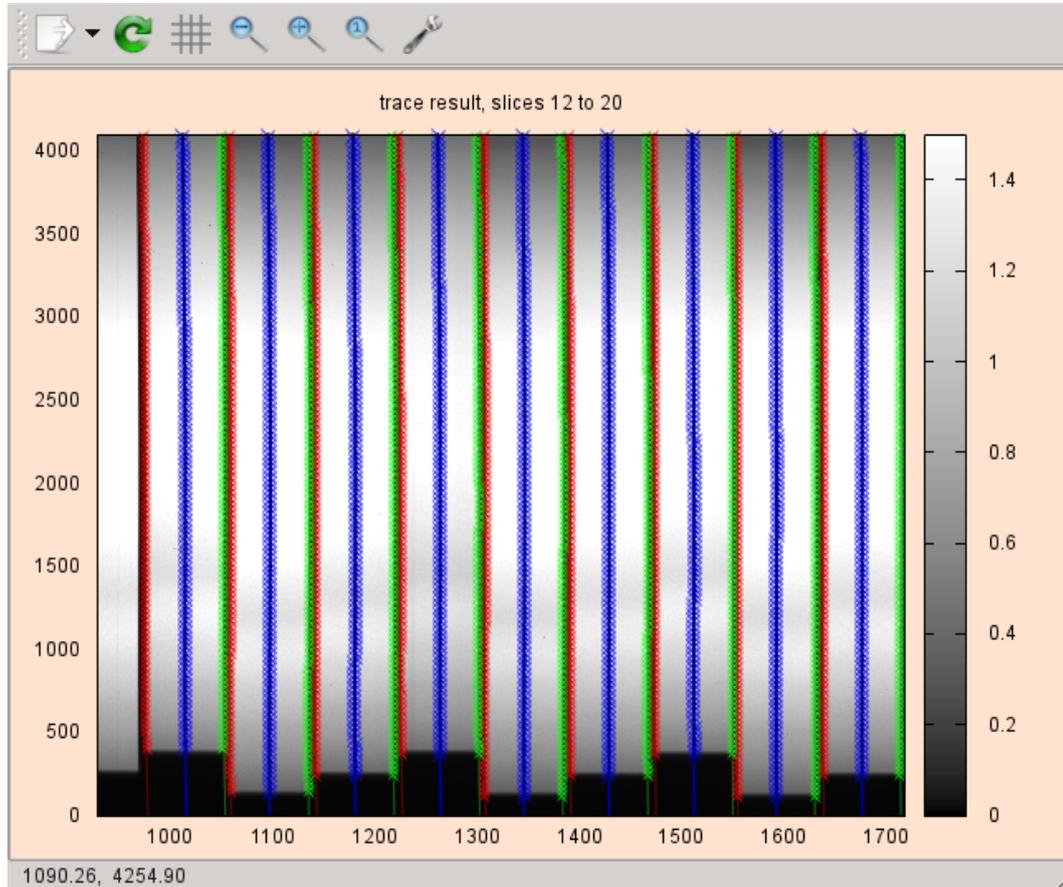
Figure 7.1: The graphical window showing the output of the `muse_trace_plot_samples` tool, then plotting slices 12 to 20 in IFU 6, using the trace samples table, the trace table, and the master flat-field image (see text for details).

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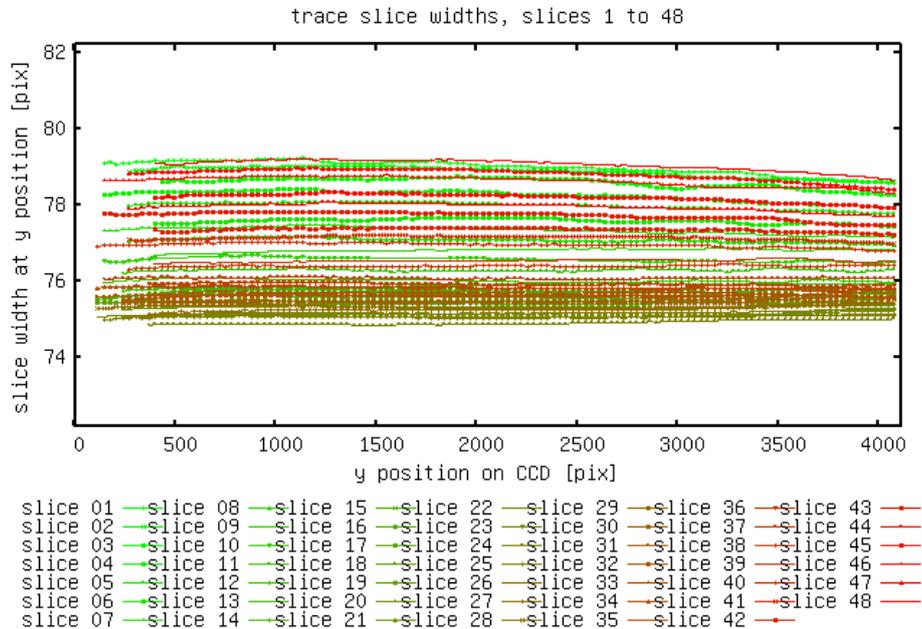
Figure 7.2: The graphical window showing the output of the `muse_trace_plot_widths` tool, plotting slices 1 to 48 of IFU 6, using the trace samples table (see text for details).

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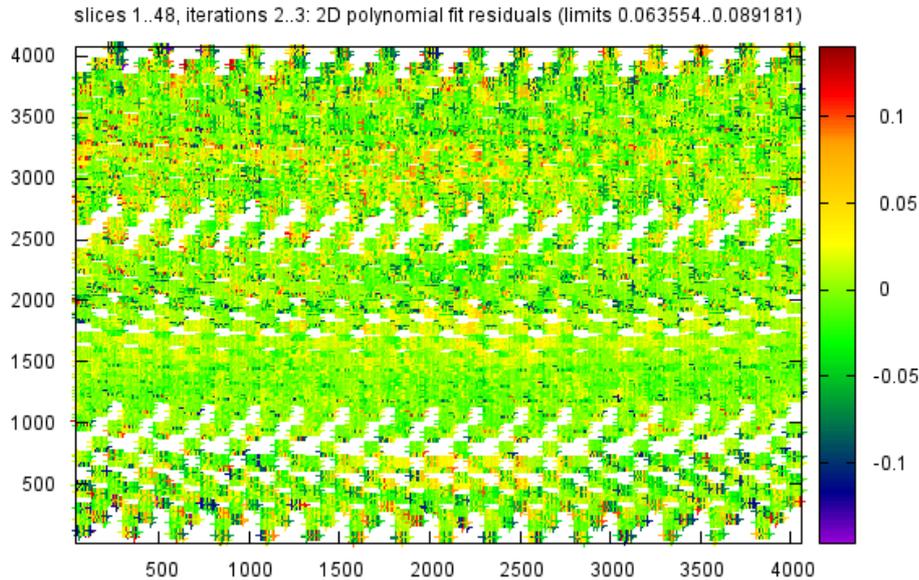
Figure 7.3: The graphical window showing the output of the `muse_wave_plot_residuals` tool, plotting all slices of IFU 10, using the wavelength calibration residuals table (see text for details).

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 coloured 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. Ne I at 6678.3 Å). With the default parameters of **muse_wavecal** (i.e. option `--fitweighting=cerrscatter`) the weak lines are hence weighted much less in the fit of the wavelength solution than the bright lines.
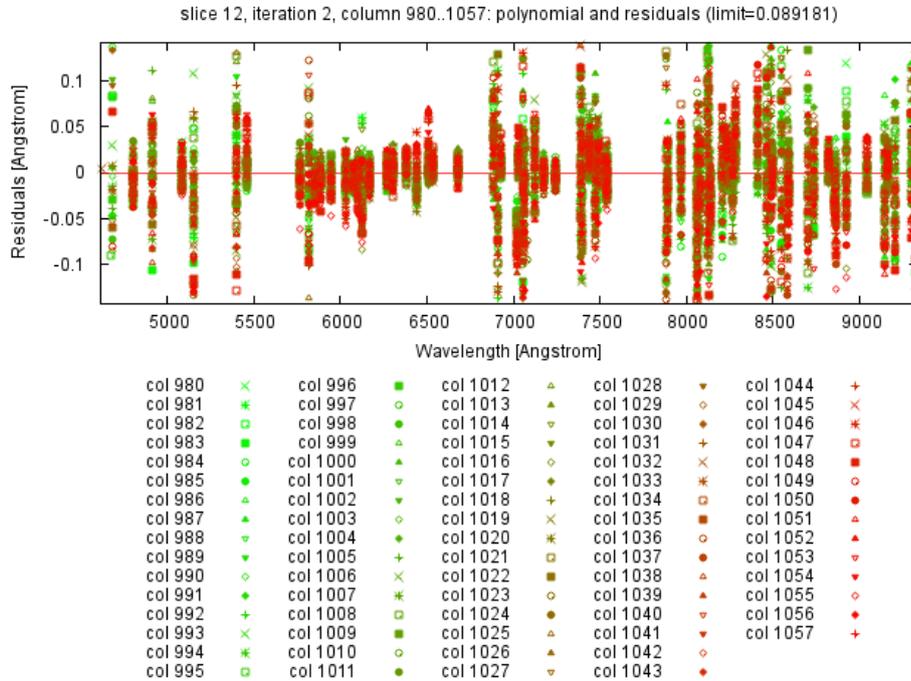
Figure 7.4: The graphical window showing the output of the `muse_wave_plot_column` tool, plotting slice 12 of IFU 10, using the wavelength calibration residuals and wavelength calibration tables (see text for details).

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 echoes the range specified with the units expected for a given pixel table. The command above outputs:

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

7.3.3 Working with Data Cubes

The tool muse_cube_combine is useful, if one has to deal with a large dataset which cannot be fully combined with the **muse_scipost** or **muse_exp_combine** recipes. In that case, one can run **muse_exp_combine** several times, setting the wavelength limits (see Sect. 7.1) such that they overlap in only about 2 wavelength planes in the output cubes. One then has to take care to re-sample all sub-cubes to the same output grid, defined using the OUTPUT_WCS input (see Sect. 9.16 and Sect. A). They will then only be filled at the relevant wavelength ranges, the rest will contain NaN. Then this tool can be used to combine them into a fully populated cube.

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

```
1> esorex muse_exp_combine [...] --lambdamax=6285. ec.sof
2> mv DATACUBE_FINAL.fits CUBE_blue.fits
3> esorex muse_exp_combine [...] --lambdamin=6282.5 --lambdamax=7817.5 ec.sof
4> mv DATACUBE_FINAL.fits CUBE_green.fits
5> esorex muse_exp_combine [...] --lambdamin=7815. ec.sof
6> mv DATACUBE_FINAL.fits CUBE_red.fits
```

and the following *set-of-frames*

```
1> cat ec.sof
PIXTABLE_REDUCED_0001_EXP01.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP02.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP03.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP04.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP05.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP06.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP07.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP08.fits PIXTABLE_REDUCED
PIXTABLE_REDUCED_0001_EXP09.fits PIXTABLE_REDUCED
cube_wcs_header.fits OUTPUT_WCS
```

The FITS header of the reference file `cube_wcs_header.fits` contains the world coordinate system covering the full axis ranges of the final data set[20]. The WCS information must be given using the `CDi_j` notation.

Then, to combine the three partial data sets one can run `muse_cube_combine` as shown in the following:

```
1> muse_cube_combine CUBE_COMBINED.fits CUBE_blue.fits CUBE_green.fits \
   CUBE_red.fits
```

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

7.3.4 Reconstructing cubes from many exposures over big field

In case the cube itself is already big compared to the memory of the machine, one has to use a somewhat different strategy. This case can happen for the case of large mosaics, like e.g. the Orion Nebula (cf. [RD09]).

In this case one needs to create dedicated `OUTPUT_WCS` files for each wavelength range. They should describe grids that are exactly adjacent in wavelength and about three bins smaller than the wavelength range given to **muse_exp_combine** by the lambdamin/max parameters. In a case where the data is four times as big as the available RAM and it has a large field of view, then one would need to create four different `OUTPUT_WCS`. Normally, these files would all have `CD3_3=1.25` and `CRPIX3=1.`, but would differ in the values of `NAXIS3` and

---

[20]As reference header one can use the header of one of the reduced pixel tables from the input *set-of-frames*, whose WCS information is changed such that it covers the full, final data set.

CRVAL3. The following shows the commands to issue and the keywords that change in the `OUTPUT_WCS` for each of the four wavelength ranges:

```
range 1:
    CRVAL3=4600.0
    NAXIS3=1120
    esorex muse_exp_combine [...] --lambdamax=6003.0 ec.sof
    mv DATACUBE_FINAL.fits CUBE_1.fits
range 2:
    CRVAL3=6000.0
    NAXIS3=800
    esorex muse_exp_combine [...] --lambdamin=5997.0 --lambdamax=7003.0 ec.sof
    mv DATACUBE_FINAL.fits CUBE_2.fits
range 3:
    CRVAL3=7000.0
    NAXIS3=800
    esorex muse_exp_combine [...] --lambdamin=6997.0 --lambdamax=8003.0 ec.sof
    mv DATACUBE_FINAL.fits CUBE_3.fits
range 4:
    CRVAL3=8000.0
    NAXIS3=1040
    esorex muse_exp_combine [...] --lambdamin=7997.0 ec.sof
    mv DATACUBE_FINAL.fits CUBE_4.fits
```

To combine these cubes the tool `muse_cube_concatenate` can then be used like this:

```
1> muse_cube_concatenate CUBE_final.fits CUBE_1.fits CUBE_2.fits \
    CUBE_3.fits CUBE_4.fits
```

The output cube `CUBE_final.fits` then contains a contiguous wavelength coverage, from 4600 Åto 9300 Å.

7.3.5 Integrating cubes using filter functions

The utility `muse_cube_filter` can be used to integrate an existing cube in dispersion direction over a filter function. While this is usually done by the recipes **muse_scipost** or **muse_exp_combine**, the tool can be useful if a filter was not considered when the recipes were run in first place, or when additional field-of-view images should be created after partial data cubes were combined using `muse_cube_combine` or `muse_cube_concatenate`.

The typical usage of `muse_cube_combine` is shown in the following:

```
1> muse_cube_filter -f Johnson_V,Cousins_R,Cousins_I DATACUBE.fits \
    filter_list.fits
```

This creates reconstructed field-of-view images for the *V*, *R*, and *I* filters from the given cube. Only the filters which are present in the given filter list file can be used.

In the previous example the images are created as separate files. They are saved as FITS files named according to the filter name, e.g. `DATACUBE_Cousins_R.fits`. Instead of creating new files the images are appended to the input file as FITS image extensions if the option `-x` is used:

```
1> muse_cube_filter -x -f Johnson_V,Cousins_R,Cousins_I DATACUBE.fits \
   filter_list.fits
```

Note that no backup of the original file is created!

The tool uses the same routine as the pipeline recipes that create output cubes (and optional field-of-view images (frame tag `IMAGE_FOV`). The integration over the filter is done using

$$f_{\text{pixel}} = \frac{\sum w_{\text{filter}} \delta\lambda \ f_{\text{voxel}}}{\sum w_{\text{filter}} \delta\lambda}$$

where w_{filter} are the unit-less transmission values of the filter involved, $\delta\lambda$ is the size of each wavelength bin (in Angstrom) and f_{voxel} is the flux of the voxel (in units of $\text{erg s}^{-1}\,\text{cm}^{-2}\,\text{Å}^{-1}$). The output f_{pixel} is then the pixel value of the output field-of-view image, in the same units.

Some of the standard filters provided by the filter list file, which is distributed as part of the MUSE pipeline package (e.g. the Johnson *V* filter), have known photometric zero-points to compute magnitudes in both the Vega and AB systems. These photometric zero-points are propagated in the FITS header of the `IMAGE_FOV` file or image extension of the cube (keywords `DRS.MUSE.FILTER.ZPVEGA` and `DRS.MUSE.FILTER.ZPAB`), together with the keyword `DRS.MUSE.FILTER.FRACTION` indicating how much of the filter band pass was actually covered by the MUSE data that was integrated. Using these zero-points only makes sense, if this filter coverage-fraction is very high (i.e. above 99 %) and if the corresponding cube was flux-calibrated. In addition one also needs to apply the scale factor, which is given as part of the `BUNIT` keyword.

Filters which do not carry the zero-points (like the built-in "white" filter and the special "Kron V" filter needed for instrument-internal calibrations) cannot be used to do photometry.

7.3.6 Reducing sky emission residuals

The sky subtraction built into the MUSE DRS may leave residual sky emission in the form of narrow artefacts in the fully processed spectra stored in the final data cube. These residual features originate from small differences between the subtracted model of the sky emission and the actual sky emission recorded by the instrument. While it is still desirable to improve the built-in sky subtraction, there is also a way to effectively reduce the residual features of the sky emission by post-processing the final data cube.

This post-processing step employs a PCA based sky subtraction (cf. [RD13]) and is available as a stand-alone tool here http://muse-vlt.eu/science/tools.

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 sky-flat handling (**muse_skyflat**). Most critically, the science reduction (**muse_scibasic**) will stop when detecting such a broken trace table.
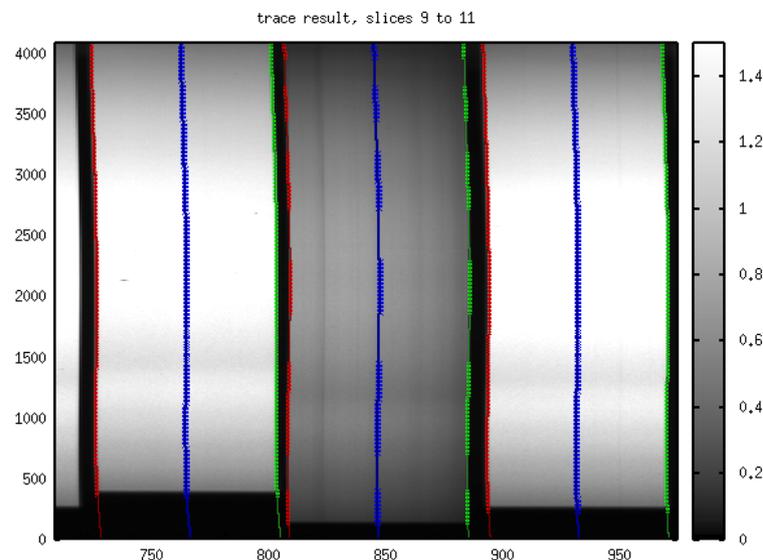
Figure 8.1: The graphical window showing the output of the `muse_trace_plot_samples` tool (see text for details).

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 log-file contains a lot of information that is related to the data reduction. Especially, if you encounter a problem, reading the log-file is likely to give you an idea at which point in the process the problem occurred. The log-file 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 log-file) 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 log-file 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 sections the documentation of the individual 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	min	Description
BIAS	raw	3	Raw bias (more than 3 frames allowed)
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)

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.
			Continued on next 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	**30.**	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 (the level comparison is then done afterwards with $|$100 x stdev$|$ to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section 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
merge	boolean	**false**	Merge output products from different IFUs into a common file.

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

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

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

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

`QC.BIAS.MASTERn.MAX.AVG`   Average of ESO QC BIAS MASTERn MAX in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MAX.RMS`   RMS of ESO QC BIAS MASTERn MAX in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MEAN.AVG`   Average of ESO QC BIAS MASTERn MEAN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MEAN.RMS`   RMS of ESO QC BIAS MASTERn MEAN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MEDIAN.AVG`   Average of ESO QC BIAS MASTERn MEDIAN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MEDIAN.RMS`   RMS of ESO QC BIAS MASTERn MEDIAN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MIN.AVG`   Average of ESO QC BIAS MASTERn MIN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.MIN.RMS`   RMS of ESO QC BIAS MASTERn MIN in quadrant n over all m IFUs

`QC.BIAS.MASTERn.RON.AVG`   Average of ESO QC BIAS MASTERn RON in quadrant n over all m IFUs

`QC.BIAS.MASTERn.RON.RMS`   RMS of ESO QC BIAS MASTERn RON in quadrant n over all m IFUs

`QC.BIAS.MASTERn.RONERR.AVG`   Average of ESO QC BIAS MASTERn RONERR in quadrant n over all m IFUs

`QC.BIAS.MASTERn.RONERR.RMS`   RMS of ESO QC BIAS MASTERn RONERR in quadrant n over all m IFUs

`QC.BIAS.MASTERn.SLOPE.X.AVG`   Average of ESO QC BIAS MASTERn SLOPE X in quadrant n over all m IFUs

`QC.BIAS.MASTERn.SLOPE.X.RMS`   RMS of ESO QC BIAS MASTERn SLOPE X in quadrant n over all m IFUs

`QC.BIAS.MASTERn.SLOPE.Y.AVG`   Average of ESO QC BIAS MASTERn SLOPE Y in quadrant n over all m IFUs

`QC.BIAS.MASTERn.SLOPE.Y.RMS`   RMS of ESO QC BIAS MASTERn SLOPE Y in quadrant n over all m IFUs

`QC.BIAS.MASTERn.STDEV.AVG`   Average of ESO QC BIAS MASTERn STDEV in quadrant n over all m IFUs

`QC.BIAS.MASTERn.STDEV.RMS`   RMS of ESO QC BIAS MASTERn STDEV in quadrant n over all m IFUs

`QC.BIAS.LEVELn.MEAN.AVG`   Average of ESO QC BIAS LEVELn MEAN in quadrant n over all m IFUs

`QC.BIAS.LEVELn.MEAN.RMS`   RMS of ESO QC BIAS LEVELn MEAN in quadrant n over all m IFUs

`QC.BIAS.LEVELn.MEDIAN.AVG`   Average of ESO QC BIAS LEVELn MEDIAN in quadrant n over all m IFUs

`QC.BIAS.LEVELn.MEDIAN.RMS`   RMS of ESO QC BIAS LEVELn MEDIAN in quadrant n over all m IFUs

`QC.BIAS.LEVELn.STDEV.AVG`   Average of ESO QC BIAS LEVELn STDEV in quadrant n over all m IFUs

`QC.BIAS.LEVELn.STDEV.RMS`   RMS of ESO QC BIAS LEVELn STDEV in quadrant n over all m IFUs

`QC.BIAS.MASTER.NBADPIX.AVG`   Average of ESO QC BIAS MASTER NBADPIX over all m IFUs

`QC.BIAS.MASTER.NBADPIX.RMS`   RMS of ESO QC BIAS MASTER NBADPIX over all m IFUs

`QC.BIAS.MASTER.NSATURATED.AVG`   Average of ESO QC BIAS MASTER NSATURATED over all m IFUs

`QC.BIAS.MASTER.NSATURATED.RMS`   RMS of ESO QC BIAS MASTER NSATURATED over all m IFUs

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 1 hour exposure time. QC statistics are computed on the output master dark.

If --model=true, a smooth polynomial model of the combined master dark is computed, created from several individual 2D polynomials to describe different features visible in MUSE dark frames. It is only advisable to use this, if the master dark is the result of at least 50 individual long dark exposures.

9.2.2 Input frames

Category	Type	min	Description
DARK	raw	3	Raw dark (more than 3 frames allowed)
MASTER_BIAS	calib	1	Master bias
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)

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.
			Continued on next page

Parameter	Type	Values **default**, other	Description
– continued from previous page			
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	**30.**	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 (the level comparison is then done afterwards with $|$100 x stdev$|$ to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section 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.
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'.
model	boolean	**false**	Model the master dark using a set of polynomials.
merge	boolean	**false**	Merge output products from different IFUs into a common file.

9.2.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_DARK	Master dark
MODEL_DARK	Model of the master dark (if --model=true).

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

`QC.DARK.INPUT1.NSATURATED.AVG`   Average of ESO QC DARK INPUT1 NSATURATED over all m IFUs

`QC.DARK.INPUT1.NSATURATED.RMS`   RMS of ESO QC DARK INPUT1 NSATURATED over all m IFUs

`QC.DARK.INPUT2.NSATURATED.AVG`   Average of ESO QC DARK INPUT2 NSATURATED over all m IFUs

`QC.DARK.INPUT2.NSATURATED.RMS`   RMS of ESO QC DARK INPUT2 NSATURATED over all m IFUs

`QC.DARK.INPUT3.NSATURATED.AVG`   Average of ESO QC DARK INPUT3 NSATURATED over all m IFUs

`QC.DARK.INPUT3.NSATURATED.RMS`   RMS of ESO QC DARK INPUT3 NSATURATED over all m IFUs

`QC.DARK.INPUT4.NSATURATED.AVG`   Average of ESO QC DARK INPUT4 NSATURATED over all m IFUs

`QC.DARK.INPUT4.NSATURATED.RMS`   RMS of ESO QC DARK INPUT4 NSATURATED over all m IFUs

`QC.DARK.INPUT5.NSATURATED.AVG`   Average of ESO QC DARK INPUT5 NSATURATED over all m IFUs

`QC.DARK.INPUT5.NSATURATED.RMS`   RMS of ESO QC DARK INPUT5 NSATURATED over all m IFUs

`QC.DARK.MASTER.DC.AVG`   Average of ESO QC DARK MASTER DC over all m IFUs

`QC.DARK.MASTER.DC.RMS`   RMS of ESO QC DARK MASTER DC over all m IFUs

`QC.DARK.MASTER.DCERR.AVG`   Average of ESO QC DARK MASTER DCERR over all m IFUs

`QC.DARK.MASTER.DCERR.RMS`   RMS of ESO QC DARK MASTER DCERR over all m IFUs

`QC.DARK.MASTER.MAX.AVG`   Average of ESO QC DARK MASTER MAX over all m IFUs

`QC.DARK.MASTER.MAX.RMS`   RMS of ESO QC DARK MASTER MAX over all m IFUs

`QC.DARK.MASTER.MIN.AVG`   Average of ESO QC DARK MASTER MIN over all m IFUs

`QC.DARK.MASTER.MIN.RMS`   RMS of ESO QC DARK MASTER MIN over all m IFUs

`QC.DARK.MASTER.MEAN.AVG`   Average of ESO QC DARK MASTER MEAN over all m IFUs

`QC.DARK.MASTER.MEAN.RMS`   RMS of ESO QC DARK MASTER MEAN over all m IFUs

`QC.DARK.MASTER.MEDIAN.AVG`   Average of ESO QC DARK MASTER MEDIAN over all m IFUs

`QC.DARK.MASTER.MEDIAN.RMS`   RMS of ESO QC DARK MASTER MEDIAN over all m IFUs

`QC.DARK.MASTER.NBADPIX.AVG`   Average of ESO QC DARK MASTER NBADPIX over all m IFUs

`QC.DARK.MASTER.NBADPIX.RMS`   RMS of ESO QC DARK MASTER NBADPIX over all m IFUs

`QC.DARK.MASTER.NSATURATED.AVG`   Average of ESO QC DARK MASTER NSATURATED over all m IFUs

`QC.DARK.MASTER.NSATURATED.RMS`   RMS of ESO QC DARK MASTER NSATURATED over all m IFUs

`QC.DARK.MASTER.STDEV.AVG`   Average of ESO QC DARK MASTER STDEV over all m IFUs

`QC.DARK.MASTER.STDEV.RMS`   RMS of ESO QC DARK MASTER STDEV over all m IFUs

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	min	Description
FLAT	raw	3	Raw flat (more than 3 frames allowed)
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)

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

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	**30.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than $\vert$ovscsigma x stdev$\vert$, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit (the level comparison is then done afterwards with $\vert$100 x stdev$\vert$ to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section 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
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	**0.4**	Fractional change required to identify edge when tracing
losigmabadpix	double	**5.**	Low sigma to find dark pixels in the master flat
hisigmabadpix	double	**5.**	High sigma to find bright pixels in the master flat
samples	boolean	**false**	Create a table containing all tracing sample points.
merge	boolean	**false**	Merge output products from different IFUs into a common file.

9.3.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
MASTER_FLAT	Master flat
TRACE_TABLE	Trace table
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.FLAT.MASTER.SLICEj.MEAN.AVG`   Average of ESO QC FLAT MASTER SLICEj MEAN in slice j over all m IFUs

`QC.FLAT.MASTER.SLICEj.MEAN.RMS`   RMS of ESO QC FLAT MASTER SLICEj MEAN in slice j over all m IFUs

`QC.FLAT.MASTER.SLICEj.STDEV.AVG`   Average of ESO QC FLAT MASTER SLICEj STDEV in slice j over all m IFUs

`QC.FLAT.MASTER.SLICEj.STDEV.RMS`   RMS of ESO QC FLAT MASTER SLICEj STDEV in slice j over all m IFUs

`QC.FLAT.MASTER.INTFLUX.AVG`   Average of ESO QC FLAT MASTER INTFLUX over all m IFUs

`QC.FLAT.MASTER.INTFLUX.RMS`   RMS of ESO QC FLAT MASTER INTFLUX over all m IFUs

`QC.FLAT.MASTER.MAX.AVG`   Average of ESO QC FLAT MASTER MAX over all m IFUs

`QC.FLAT.MASTER.MAX.RMS`   RMS of ESO QC FLAT MASTER MAX over all m IFUs

`QC.FLAT.MASTER.MIN.AVG`   Average of ESO QC FLAT MASTER MIN over all m IFUs

`QC.FLAT.MASTER.MIN.RMS`   RMS of ESO QC FLAT MASTER MIN over all m IFUs

`QC.FLAT.MASTER.MEAN.AVG`   Average of ESO QC FLAT MASTER MEAN over all m IFUs

`QC.FLAT.MASTER.MEAN.RMS`   RMS of ESO QC FLAT MASTER MEAN over all m IFUs

`QC.FLAT.MASTER.MEDIAN.AVG`   Average of ESO QC FLAT MASTER MEDIAN over all m IFUs

`QC.FLAT.MASTER.MEDIAN.RMS`   RMS of ESO QC FLAT MASTER MEDIAN over all m IFUs

`QC.FLAT.MASTER.STDEV.AVG`   Average of ESO QC FLAT MASTER STDEV over all m IFUs

`QC.FLAT.MASTER.STDEV.RMS`   RMS of ESO QC FLAT MASTER STDEV over all m IFUs

`QC.FLAT.MASTER.NSATURATED.AVG`   Average of ESO QC FLAT MASTER NSATURATED over all m IFUs

`QC.FLAT.MASTER.NSATURATED.RMS`   RMS of ESO QC FLAT MASTER NSATURATED over all m IFUs

`QC.NTRACETABLE`   Number of TRACE_TABLE outputs created by the recipe

`QC.TOPSLICE.FLUX`   Average flux of the 4 top row slices (10,22,34,46) normalized by the average flux of the slices in the channel

`QC.TOPSLICE.FLUX.AVG`   Average of ESO QC TOPSLICE FLUX over all m IFUs

`QC.TOPSLICE.FLUX.RMS`   RMS of ESO QC TOPSLICE FLUX over all m IFUs

`QC.BOTTOMSLICE.FLUX`   Average flux of the 4 bottom row slices (3,15,27,39) normalized by the average flux of the slices in the channel

`QC.BOTTOMSLICE.FLUX.AVG`   Average of ESO QC BOTTOMSLICE FLUX over all m IFUs

`QC.BOTTOMSLICE.FLUX.RMS`   RMS of ESO QC BOTTOMSLICE FLUX over all m IFUs

`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.SLICE10.WIDTH`   Width of top left slice in the IFU (10 on CCD)

`QC.TRACE.SLICE46.WIDTH`   Width of top right slice in the IFU (46 on CCD)

`QC.TRACE.SLICE3.WIDTH`   Width of bottom left slice in the IFU (3 on CCD)

`QC.TRACE.SLICE39.WIDTH`   Width of bottom right slice in the IFU (39 on 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

`QC.TRACE.GAPS.MAX.AVG`   Average of ESO QC TRACE GAPS MAX over all m IFUs

`QC.TRACE.GAPS.MAX.RMS`   RMS of ESO QC TRACE GAPS MAX over all m IFUs

`QC.TRACE.GAPS.MIN.AVG`   Average of ESO QC TRACE GAPS MIN over all m IFUs

`QC.TRACE.GAPS.MIN.RMS`   RMS of ESO QC TRACE GAPS MIN over all m IFUs

`QC.TRACE.GAPS.MEAN.AVG`   Average of ESO QC TRACE GAPS MEAN over all m IFUs

`QC.TRACE.GAPS.MEAN.RMS`   RMS of ESO QC TRACE GAPS MEAN over all m IFUs

`QC.TRACE.GAPS.MEDIAN.AVG`   Average of ESO QC TRACE GAPS MEDIAN over all m IFUs

`QC.TRACE.GAPS.MEDIAN.RMS`   RMS of ESO QC TRACE GAPS MEDIAN over all m IFUs

`QC.TRACE.GAPS.STDEV.AVG`   Average of ESO QC TRACE GAPS STDEV over all m IFUs

`QC.TRACE.GAPS.STDEV.RMS`   RMS of ESO QC TRACE GAPS STDEV over all m IFUs

`QC.TRACE.SLICE_L.TILT.AVG`	Average of ESO QC TRACE SLICE_L TILT over all m IFUs
`QC.TRACE.SLICE_L.TILT.RMS`	RMS of ESO QC TRACE SLICE_L TILT over all m IFUs
`QC.TRACE.SLICE_L.XPOS.AVG`	Average of ESO QC TRACE SLICE_L XPOS over all m IFUs
`QC.TRACE.SLICE_L.XPOS.RMS`	RMS of ESO QC TRACE SLICE_L XPOS over all m IFUs
`QC.TRACE.SLICE_R.TILT.AVG`	Average of ESO QC TRACE SLICE_R TILT over all m IFUs
`QC.TRACE.SLICE_R.TILT.RMS`	RMS of ESO QC TRACE SLICE_R TILT over all m IFUs
`QC.TRACE.SLICE_R.XPOS.AVG`	Average of ESO QC TRACE SLICE_R XPOS over all m IFUs
`QC.TRACE.SLICE_R.XPOS.RMS`	RMS of ESO QC TRACE SLICE_R XPOS over all m IFUs
`QC.TRACE.WIDTHS.MAX.AVG`	Average of ESO QC TRACE WIDTHS MAX over all m IFUs
`QC.TRACE.WIDTHS.MAX.RMS`	RMS of ESO QC TRACE WIDTHS MAX over all m IFUs
`QC.TRACE.WIDTHS.MIN.AVG`	Average of ESO QC TRACE WIDTHS MIN over all m IFUs
`QC.TRACE.WIDTHS.MIN.RMS`	RMS of ESO QC TRACE WIDTHS MIN over all m IFUs
`QC.TRACE.WIDTHS.MEAN.AVG`	Average of ESO QC TRACE WIDTHS MEAN over all m IFUs
`QC.TRACE.WIDTHS.MEAN.RMS`	RMS of ESO QC TRACE WIDTHS MEAN over all m IFUs
`QC.TRACE.WIDTHS.MEDIAN.AVG`	Average of ESO QC TRACE WIDTHS MEDIAN over all m IFUs
`QC.TRACE.WIDTHS.MEDIAN.RMS`	RMS of ESO QC TRACE WIDTHS MEDIAN over all m IFUs
`QC.TRACE.WIDTHS.STDEV.AVG`	Average of ESO QC TRACE WIDTHS STDEV over all m IFUs
`QC.TRACE.WIDTHS.STDEV.RMS`	RMS of ESO QC TRACE WIDTHS STDEV over all m IFUs
`QC.TRACE.SLICEj.WIDTH.AVG`	Average of ESO QC TRACE SLICEj WIDTH in slice j over all m IFUs
`QC.TRACE.SLICEj.WIDTH.RMS`	RMS of ESO QC TRACE SLICEj WIDTH in slice j over all m IFUs
`QC.TOPSLICE.WIDTH`	The value of ESO QC TRACE SLICE10 WIDTH minus ESO QC TRACE SLICE3 WIDTH
`QC.TOPSLICE.WIDTH.AVG`	Average of ESO QC TOPSLICE WIDTH over all m IFUs
`QC.TOPSLICE.WIDTH.RMS`	RMS of ESO QC TOPSLICE WIDTH over all m IFUs
`QC.BOTTOMSLICE.WIDTH`	The value of ESO QC TRACE SLICE46 WIDTH minus ESO QC TRACE SLICE39 WIDTH
`QC.BOTTOMSLICE.WIDTH.AVG`	Average of ESO QC BOTTOMSLICE WIDTH over all m IFUs
`QC.BOTTOMSLICE.WIDTH.RMS`	RMS of ESO QC BOTTOMSLICE WIDTH over all m IFUs

9.4 muse_wavecal

Detect arc emission lines and determine the wavelength solution 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, into separate images for each lamp.

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. 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).

9.4.2 Input frames

Category	Type	min	Description
ARC	raw	1	Raw arc lamp exposures
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
MASTER_FLAT	calib		Master flat (optional, usually not used)
TRACE_TABLE	calib	1	Trace table
LINE_CATALOG	calib	1	Arc line catalog
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)

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	**30.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than $\lvert$ovscsigma x stdev$\rvert$, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit (the level comparison is then done afterwards with $\lvert 100$ x stdev$\rvert$ to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section 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.
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

Parameter	Type	Values **default**, other	Description
		– continued from previous page	
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 data-points 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.
saveimages	boolean	**false**	Save
merge	boolean	**false**	Merge output products from different IFUs into a common file.

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 (if --saveimages=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.

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.

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

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.AVG   Average of ESO QC WAVECAL SLICEj FIT in slice j over all m IFUs

QC.WAVECAL.SLICEj.FIT.RMS   RMS of ESO QC WAVECAL SLICEj FIT in slice j over all m IFUs

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.NSATURATED.LAMPl   Number of saturated pixels in output data of lamp l

QC.WAVECAL.SLICEj.DWLEN.BOTTOM.AVG   Average of ESO QC WAVECAL SLICEj DWLEN BOTTOM in slice j over all m IFUs

QC.WAVECAL.SLICEj.DWLEN.BOTTOM.RMS   RMS of ESO QC WAVECAL SLICEj DWLEN BOTTOM in slice j over all m IFUs

QC.WAVECAL.SLICEj.DWLEN.TOP.AVG   Average of ESO QC WAVECAL SLICEj DWLEN TOP in slice j over all m IFUs

QC.WAVECAL.SLICEj.DWLEN.TOP.RMS   RMS of ESO QC WAVECAL SLICEj DWLEN TOP in slice j over all m IFUs

QC.WAVECAL.SLICEj.FIT.NLINES.AVG   Average of ESO QC WAVECAL SLICEj FIT NLINES in slice j over all m IFUs

QC.WAVECAL.SLICEj.FIT.NLINES.RMS   RMS of ESO QC WAVECAL SLICEj FIT NLINES in slice j over all m IFUs

QC.WAVECAL.SLICEj.LINES.FWHM.MEAN.AVG   Average of ESO QC WAVECAL SLICEj LINES FWHM MEAN in slice j over all m IFUs

QC.WAVECAL.SLICEj.LINES.FWHM.MEAN.RMS   RMS of ESO QC WAVECAL SLICEj LINES FWHM MEAN in slice j over all m IFUs

QC.WAVECAL.SLICEj.LINES.NID.AVG   Average of ESO QC WAVECAL SLICEj LINES NID in slice j over all m IFUs

QC.WAVECAL.SLICEj.LINES.NID.RMS   RMS of ESO QC WAVECAL SLICEj LINES NID in slice j over all m IFUs

QC.WAVECAL.SLICEj.RESOL.AVG   Average of ESO QC WAVECAL SLICEj RESOL in slice j over all m IFUs

QC.WAVECAL.SLICEj.RESOL.RMS   RMS of ESO QC WAVECAL SLICEj RESOL in slice j over all m IFUs

QC.WAVECAL.SLICEj.WLEN.AVG   Average of ESO QC WAVECAL SLICEj WLEN in slice j over all m IFUs

QC.WAVECAL.SLICEj.WLEN.RMS   RMS of ESO QC WAVECAL SLICEj WLEN in slice j over all m IFUs

QC.WAVECAL.SLICEj.WLPOS.AVG   Average of ESO QC WAVECAL SLICEj WLPOS in slice j over all m IFUs

QC.WAVECAL.SLICEj.WLPOS.RMS   RMS of ESO QC WAVECAL SLICEj WLPOS in slice j over all m IFUs

`QC.WAVECAL.SLICE20.LAMP3.LINES.PEAK.MEAN.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP3 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP3.LINES.PEAK.MEAN.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP3 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP3.LINES.PEAK.MAX.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP3 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.SLICE20.LAMP3.LINES.PEAK.MAX.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP3 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.SLICE20.LAMP4.LINES.PEAK.MEAN.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP4 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP4.LINES.PEAK.MEAN.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP4 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP4.LINES.PEAK.MAX.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP4 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.SLICE20.LAMP4.LINES.PEAK.MAX.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP4 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.SLICE20.LAMP5.LINES.PEAK.MEAN.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP5 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP5.LINES.PEAK.MEAN.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP5 LINES PEAK MEAN over all m IFUs
`QC.WAVECAL.SLICE20.LAMP5.LINES.PEAK.MAX.AVG`	Average of ESO QC WAVECAL SLICE20 LAMP5 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.SLICE20.LAMP5.LINES.PEAK.MAX.RMS`	RMS of ESO QC WAVECAL SLICE20 LAMP5 LINES PEAK MAX over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP3.AVG`	Average of ESO QC WAVECAL NSATURATED LAMP3 over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP3.RMS`	RMS of ESO QC WAVECAL NSATURATED LAMP3 over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP4.AVG`	Average of ESO QC WAVECAL NSATURATED LAMP4 over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP4.RMS`	RMS of ESO QC WAVECAL NSATURATED LAMP4 over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP5.AVG`	Average of ESO QC WAVECAL NSATURATED LAMP5 over all m IFUs
`QC.WAVECAL.NSATURATED.LAMP5.RMS`	RMS of ESO QC WAVECAL NSATURATED LAMP5 over all m IFUs
`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

9.5 muse_lsf

Compute the LSF

9.5.1 Description

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

9.5.2 Input frames

At least one raw frame of the category ARC or ARC_LSF is required.

Category	Type	min	Description
ARC	raw		Raw arc lamp exposures (from a standard arc sequence)
ARC_LSF	raw		Raw arc lamp exposures (from a long LSF arc sequence)
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
MASTER_FLAT	calib		Master flat (optional)
TRACE_TABLE	calib	1	Trace table
WAVECAL_TABLE	calib	1	Wavelength calibration table
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)
LINE_CATALOG	calib	1	Arc line catalog

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.
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	**30.**	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 (the level comparison is then done afterwards with |100 x stdev| to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section of the CCD that are ignored when computing statistics or fits.
save_subtracted	boolean	**false**	Save the pixel table after the LSF subtraction.
line_quality	int	**3**	Minimal quality flag in line catalog for selection
lsf_range	double	**7.5**	Wavelength window (half size) around each line to estimate LSF
lsf_size	int	**150**	Image size in LSF direction
lambda_size	int	**30**	Image size in line wavelength direction
lsf_regression_window	double	**0.7**	Size of the regression window in LSF direction
merge	boolean	**false**	Merge output products from different IFUs into a common file.
combine	string	**sigclip**, average, median, minmax, sigclip	Type of lampwise image combination to use.
method	string	**interpolate**, interpolate, hermit	LSF generation method. Depending on this value, either an interpolated LSF cube is created, or a table with the parameters of a hermitean gaussian.

9.5.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
LSF_PROFILE	Slice specific LSF images, stacked into one data cube per IFU.
	Continued on next page

– continued from previous page	
Default file name	Description
PIXTABLE_SUBTRACTED	Subtracted combined pixel table, if --save_subtracted=true. This file contains only the subtracted arc lines that contributed to the LSF calculation. There are additional columns line_lambda and line_flux with the reference wavelength and the estimated line flux of the corresponding arc line.

9.5.5 Quality control parameters

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

`QC.LSF.SLICEj.FWHM.MEAN`   Mean FWHM of the LSF slice j

`QC.LSF.SLICEj.FWHM.STDEV`   Standard deviation of the LSF in slice j

`QC.LSF.SLICEj.FWHM.MIN`   Minimum FWHM of the LSF in slice j

`QC.LSF.SLICEj.FWHM.MAX`   Maximum FWHM of the LSF in slice j

`QC.LSF.SLICEj.FWHM.MAX.AVG`   Average of ESO QC LSF SLICEj FWHM MAX in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.MAX.RMS`   RMS of ESO QC LSF SLICEj FWHM MAX in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.MIN.AVG`   Average of ESO QC LSF SLICEj FWHM MIN in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.MIN.RMS`   RMS of ESO QC LSF SLICEj FWHM MIN in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.MEAN.AVG`   Average of ESO QC LSF SLICEj FWHM MEAN in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.MEAN.RMS`   RMS of ESO QC LSF SLICEj FWHM MEAN in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.STDEV.AVG`   Average of ESO QC LSF SLICEj FWHM STDEV in slice j over all m IFUs

`QC.LSF.SLICEj.FWHM.STDEV.RMS`   RMS of ESO QC LSF SLICEj FWHM STDEV in slice j over all m IFUs

`QC.NSAT.INP`   Sum of ESO QC WAVECAL INPUTi NSATURATED over all i input files divided by the number of input files

`QC.NSAT.INP.AVG`   Average of ESO QC NSAT INP over all m IFUs

`QC.NSAT.INP.RMS`   RMS of ESO QC NSAT INP over all m IFUs

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 table 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. The table is then cleaned up from intermediate debug data. If the --smooth parameter is set to a positive value, it is used to do a sigma-clipped smoothing within each slicer stack, for a more regular appearance of the output table. The table is then 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	min	Description
MASK	raw	50	The full exposure sequence of raw multi-pinhole mask images (more than 50 frames allowed)
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
MASTER_FLAT	calib		Master flat (optional, usually not used)
TRACE_TABLE	calib	1	Trace table
WAVECAL_TABLE	calib	1	Wavelength calibration table
LINE_CATALOG	calib	1	List of arc lines
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)
MASK_CHECK	calib		Some other optional raw frame, ideally a trace mask exposure or something else with smooth features. (optional, more than one frame allowed)

9.6.3 Recipe parameters

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.
smooth	double	**1.5**	Use this sigma-level cut for smoothing of the output table within each slicer stack. Set to non-positive value to deactivate smoothing.

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_UNSMOOTHED	Relative positions of the slices in the field of view (un-smoothed)
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.FWHM.MEAN`   Average of the average FWHM of all bright spots in all exposures.
`QC.GEO.FWHM.STDEV`   Standard deviation of the average FWHM of all bright spots in all exposures.
`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.WLEN1`   Nominal wavelength of arc line l.

`QC.GEO.IFUm.WLEN1.FLUX.MEAN`   Average integrated flux in all spots at reference wavelength l.

`QC.GEO.IFUm.WLEN1.FLUX.MEDIAN`   Median integrated flux in all spots at reference wavelength l.

`QC.GEO.IFUm.WLEN1.FLUX.STDEV`   Standard deviation of integrated flux in all spots at reference wavelength l.

`QC.GEO.IFUm.GAPPOS.MEAN`   Average position of the central gap between the 12 slices of IFU m.

`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.

`QC.GEO.GAPPOS.MEAN`   Average of all mean central gap positions of all IFUs for which the measurement could be made.

`QC.GEO.GAPPOS.STDEV`   Standard deviation of all mean central gap positions of all IFUs for which the measurement could be made.

`QC.GEO.SMOOTH.NX`   Number of slices that were changed with respect to x position by smoothing. Gets set to -1 if smoothing is inactive.

`QC.GEO.SMOOTH.NY`   Number of slices that were changed with respect to y position by smoothing. Gets set to -1 if smoothing is inactive.

`QC.GEO.SMOOTH.NANGLE`   Number of slices that were changed with respect to angle by smoothing. Gets set to -1 if smoothing is inactive.

`QC.GEO.SMOOTH.NWIDTH`   Number of slices that were changed with respect to width by smoothing. Gets set to -1 if smoothing is inactive.

`QC.GEO.TABLE.NBAD`   Number of invalid entries in the geometry table.

9.7 muse_twilight

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

9.7.1 Description

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

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

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

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

To construct a smooth 3D illumination correction, the cube is post-processed in the following way: the white-light image is used to create a mask of the illuminated area. From this area, the optional vignetting mask is removed. The smoothing is then computed for each plane of the cube: the illuminated area is smoothed (by a 5x7 median filter), normalized, fit with a 2D polynomial (of given polynomial orders), and normalized again. A smooth white image is then created by collapsing the smooth cube.

If a vignetting mask was given or NFM data is processed, an area close to the edge of the MUSE field is used to compute a 2D correction for the vignetted area: the original unsmoothed white-light image is corrected for large scale gradients by dividing it with the smooth white image. The residuals in the edge area (as defined by the input mask or hardcoded for NFM) are then smoothed using input parameters. This smoothed vignetting correction is the multiplied onto each plane of the smooth cube, normalizing each plane again.

This twilight cube is then saved to disk.

9.7.2 Input frames

Category	Type	min	Description
SKYFLAT	raw	3	Raw twilight skyflat (more than 3 frames allowed)
			Continued on next page

– continued from previous page			
Category	Type	min	Description
ILLUM	raw		Single optional raw (attached/illumination) flat-field exposure (optional)
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
MASTER_FLAT	calib	1	Master flat
BADPIX_TABLE	calib		Known bad pixels (optional, usually not used, more than one frame allowed)
TRACE_TABLE	calib	1	Tracing table for all slices
WAVECAL_TABLE	calib	1	Wavelength calibration table
GEOMETRY_TABLE	calib	1	Relative positions of the slices in the field of view
VIGNETTING_MASK	calib		Mask to mark vignetted regions in the MUSE field of view (optional)

9.7.3 Recipe parameters

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	**30.**	If the deviation of mean overscan levels between a raw input image and the reference image is higher than $\lvert$ovscsigma x stdev$\rvert$, stop the processing. If overscan="vpoly", this is used as sigma rejection level for the iterative polynomial fit (the level comparison is then done afterwards with $\lvert$100 x stdev$\rvert$ to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section of the CCD that are ignored when computing statistics or fits.
			Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
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.
resample	string	**drizzle**, nearest, linear, quadratic, renka, drizzle, lanczos	The resampling technique to use for the final output cube.
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 the median median of the absolute median deviation.
crsigma	double	**50.**	Sigma rejection factor to use for cosmic ray rejection during final resampling. A zero or negative value switches cosmic ray rejection off.
lambdamin	double	**5000.**	Minimum wavelength for twilight reconstruction.
lambdamax	double	**9000.**	Maximum wavelength for twilight reconstruction.
dlambda	double	**250.**	Sampling for twilight reconstruction, this should result in planes of equal wavelength coverage.
xorder	int	**2**	Polynomial order to use in x direction to fit the full field of view.
yorder	int	**2**	Polynomial order to use in y direction to fit the full field of view.

9.7.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_SKYFLAT	Cube of combined twilight skyflat exposures
TWILIGHT_CUBE	Smoothed cube of twilight sky

9.7.5 Quality control parameters

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

`QC.TWILIGHTm.INPUTi.MEDIAN`   Median value of raw exposure i of IFU m in input list

`QC.TWILIGHTm.INPUTi.MEAN`   Mean value of raw exposure i of IFU m in input list

`QC.TWILIGHTm.INPUTi.STDEV`   Standard deviation of raw exposure i of IFU m in input list

`QC.TWILIGHTm.INPUTi.MIN`   Minimum value of raw exposure i of IFU m in input list

`QC.TWILIGHTm.INPUTi.MAX`   Maximum value of raw exposure i of IFU m in input list

`QC.TWILIGHTm.INPUTi.NSATURATED`   Number of saturated pixels in raw exposure i of IFU m in input list

`QC.TWILIGHTm.MASTER.MEDIAN`   Median value of the combined exposures in IFU m

`QC.TWILIGHTm.MASTER.MEAN`   Mean value of the combined exposures in IFU m

`QC.TWILIGHTm.MASTER.STDEV`   Standard deviation of the combined exposures in IFU m

`QC.TWILIGHTm.MASTER.MIN`   Minimum value of the combined exposures in IFU m

`QC.TWILIGHTm.MASTER.MAX`   Maximum value of the combined exposures in IFU m

`QC.TWILIGHTm.MASTER.INTFLUX`   Flux integrated over the whole CCD of the combined exposures of IFU m

`QC.TWILIGHTm.INTFLUX`   Flux integrated over all slices of IFU m. Computed using the pixel table of the exposure.

`QC.MASTER.MEDIAN.AVG`   Average of ESO QC TWILIGHTm MASTER MEDIAN over all m IFUs

`QC.MASTER.MEAN.AVG`   Average of ESO QC TWILIGHTm MASTER MEAN over all m IFUs

`QC.MASTER.STDEV.AVG`   Average of ESO QC TWILIGHTm MASTER STDEV over all m IFUs

`QC.MASTER.MIN.AVG`   Average of ESO QC TWILIGHTm MASTER MIN over all m IFUs

`QC.MASTER.MAX.AVG`   Average of ESO QC TWILIGHTm MASTER MAX over all m IFUs

`QC.MASTER.INTFLUX.AVG`   Average of ESO QC TWILIGHTm MASTER INTFLUX over all m IFUs

`QC.MASTER.MEDIAN.RMS`   RMS of ESO QC TWILIGHTm MASTER MEDIAN over all m IFUs

`QC.MASTER.MEAN.RMS`   RMS of ESO QC TWILIGHTm MASTER MEAN over all m IFUs

`QC.MASTER.STDEV.RMS`   RMS of ESO QC TWILIGHTm MASTER STDEV over all m IFUs

`QC.MASTER.MIN.RMS`   RMS of ESO QC TWILIGHTm MASTER MIN over all m IFUs

`QC.MASTER.MAX.RMS`   RMS of ESO QC TWILIGHTm MASTER MAX over all m IFUs

`QC.MASTER.INTFLUX.RMS`   RMS of ESO QC TWILIGHTm MASTER INTFLUX over all m IFUs

9.8 muse_scibasic

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

9.8.1 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 twilight-sky cube.

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

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

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

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

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

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

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

9.8.2 Input frames

At least one raw frame of the category OBJECT, STD, SKY or ASTROMETRY is required.

Category	Type	min	Description
OBJECT	raw		Raw exposure of a science target
STD	raw		Raw exposure of a standard star field
SKY	raw		Raw exposure of an (almost) empty sky field
ASTROMETRY	raw		Raw exposure of an astrometric field
ILLUM	raw		Single optional raw (attached/illumination) flat-field exposure (optional)
MASTER_BIAS	calib	1	Master bias
MASTER_DARK	calib		Master dark (optional, usually not used)
MASTER_FLAT	calib	1	Master flat
			Continued on next page

– continued from previous page			
Category	Type	min	Description
TRACE_TABLE	calib	1	Trace table
WAVECAL_TABLE	calib	1	Wavelength calibration table
GEOMETRY_TABLE	calib	1	Relative positions of the slices in the field of view
TWILIGHT_CUBE	calib		Smoothed cube of twilight sky (optional, more than one frame allowed)
BADPIX_TABLE	calib		Known bad pixels (optional, more than one frame allowed)

9.8.3 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	**30.**	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 (the level comparison is then done afterwards with |100 x stdev| to guard against incompatible settings). Has no effect for overscan="offset".
ovscignore	int	**3**	The number of pixels of the overscan adjacent to the data section of the CCD that are ignored when computing statistics or fits.
			Continued on next page

– continued from previous page

Parameter	Type	Values **default**, other	Description
crop	boolean	**true**	Automatically crop the output pixel tables in wavelength depending on the expected useful wavelength range of the active instrument mode (4750-9350 Angstrom for nominal mode and NFM, 4700-9350 Angstrom for nominal AO mode, and 4600-9350 Angstrom for the extended modes).
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.
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. Set to "none" to deactivate.
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.
skyreject	string	**15.,15.,1**	Sigma clipping parameters for the intermediate spectrum to do the Gaussian fit to each sky emission line. Up to three comma-separated numbers can be given, which are interpreted as high sigma-clipping limit (float), low limit (float), and number of iterations (integer), respectively.
resample	boolean	**false**	Resample the input science data into 2D spectral images using all supplied calibrations 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.

Continued on next page

Parameter	Type	Values **default**, other	Description
– continued from previous page			
merge	boolean	**false**	Merge output products from different IFUs into a common file.

9.8.4 Product frames

The following product frames are created by the recipe:

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

9.8.5 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)

9.9 muse_standard

Create a flux response curve from a standard star exposure.

9.9.1 Description

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

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

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

In MUSE's WFM data (both AO and non-AO), the Moffat profile is a good approximation of the actual PSF. Using the smoothed profile ("smoffat") helps to increase the S/N and in most cases removes systematics. In NFM, however, the profile is a combination of a wide PSF plus the central AO-corrected peak, which cannot be fit well by an analytical profile. In this case the circular aperture is the best way to extract the flux. Using --profile="auto" (the default) selects these options to give the best flux extraction for most cases.

9.9.2 Input frames

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

9.9.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
profile	string	**auto**, gaussian, moffat, smoffat, circle, square, auto	Type of flux integration to use. "gaussian", "moffat", and "smoffat" use 2D profile fitting, "circle" and "square" are non-optimal aperture flux integrators. "smoffat" uses smoothing of the Moffat parameters from an initial fit, to derive physically meaningful wavelength-dependent behavior. "auto" selects the smoothed Moffat profile for WFM data and circular flux integration for NFM.
select	string	**distance**, flux, distance	How to select the star for flux integration, "flux" uses the brightest star in the field, "distance" uses the detection nearest to the approximate coordinates of the reference source.
smooth	string	**ppoly**, none, median, ppoly	How to smooth the response curve before writing it to disk. "none" does not do any kind of smoothing (such a response curve is only useful, if smoothed externally; "median" does a median-filter of 15 Angstrom half-width; "ppoly" fits piecewise cubic polynomials (each one across 2x150 Angstrom width) postprocessed by a sliding average filter of 15 Angstrom half-width.
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.
			Continued on next page

– continued from previous page			
Parameter	Type	Values **default**, other	Description
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. If the zeropoint QC parameters are wanted, make sure to add "Johnson_V,Cousins_R,Cousins_I".

9.9.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_STD	Reduced standard star field exposure
STD_FLUXES	The integrated flux per wavelength of all detected sources
STD_RESPONSE	Response curve as derived from standard star(s)
STD_TELLURIC	Telluric absorption as derived from standard star(s)

9.9.5 Quality control parameters

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

`QC.STANDARD.NDET`   Number of detected sources in output cube.

`QC.STANDARD.LAMBDA`   Wavelength of plane in combined cube that was used for object detection.

`QC.STANDARD.POSk.X`   Position of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.

`QC.STANDARD.POSk.Y`   Position of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.

`QC.STANDARD.FWHMk.X`   FWHM of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.

`QC.STANDARD.FWHMk.Y`   FWHM of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.

`QC.STANDARD.FWHM.NVALID`   Number of detected sources with valid FWHM in output cube.

`QC.STANDARD.FWHM.MEDIAN`   Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.STANDARD.FWHM.MAD`   Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.STANDARD.POS1.X.AVG`   Average of ESO QC STANDARD POS1 X over all m IFUs

`QC.STANDARD.POS1.Y.AVG`   Average of ESO QC STANDARD POS1 Y over all m IFUs

`QC.STANDARD.FWHM1.X.AVG`   Average of ESO QC STANDARD FWHM1 X over all m IFUs

`QC.STANDARD.FWHM1.Y.AVG`   Average of ESO QC STANDARD FWHM1 Y over all m IFUs

`QC.STANDARD.POS1.X.RMS`   RMS of ESO QC STANDARD POS1 X over all m IFUs

`QC.STANDARD.POS1.Y.RMS`   RMS of ESO QC STANDARD POS1 Y over all m IFUs

`QC.STANDARD.FWHM1.X.RMS`   RMS of ESO QC STANDARD FWHM1 X over all m IFUs

`QC.STANDARD.FWHM1.Y.RMS`   RMS of ESO QC STANDARD FWHM1 Y over all m IFUs

`QC.STANDARD.STARNAME`   Name of the standard star used for the throughput / zeropoint calculation.

`QC.STANDARD.THRU5000`   Throughput computed at 5000 +/- 100 Angstrom.

`QC.STANDARD.THRU7000`   Throughput computed at 7000 +/- 100 Angstrom.

`QC.STANDARD.THRU8000`   Throughput computed at 8000 +/- 100 Angstrom.

`QC.STANDARD.THRU9000`   Throughput computed at 9000 +/- 100 Angstrom.

`QC.STANDARD.ZP.V`   Zeropoint in Johnson V filter. $zp = -2.5 \log_{10}(fobs\_V / fref\_V)$, where fobs_V was integrated over the filter curve and converted to f_lambda using the known effective VLT area. (optional) Only computed if FILTER_LIST and corresponding --filter is given.

`QC.STANDARD.ZP.R`   Zeropoint in Cousins R filter. $zp = -2.5 \log_{10}(fobs\_R / fref\_R)$, where fobs_R was integrated over the filter curve and converted to f_lambda using the known effective VLT area. (optional) Only computed if FILTER_LIST and corresponding --filter is given.

`QC.STANDARD.ZP.I`   Zeropoint in Cousins I filter. $zp = -2.5 \log_{10}(fobs\_I / fref\_I)$, where fobs_I was integrated over the filter curve and converted to f_lambda using the known effective VLT area. (optional) Only computed if FILTER_LIST and corresponding --filter is given.

`QC.RESP.ROW`   Value of NAXIS2

9.10 muse_create_sky

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

9.10.1 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.

9.10.2 Input frames

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

9.10.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
fraction	double	**0.75**	Fraction of the image (without the ignored part) to be considered as sky. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 1.
ignore	double	**0.05**	Fraction of the image to be ignored. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 0.
lambdamin	double	**4000.**	Cut off the data below this wavelength after loading the pixel table(s).

Continued on next page

Parameter	Type	Values **default**, other	Description
			– continued from previous page
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.

9.10.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
SKY_MASK	Created sky mask
SKY_IMAGE	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

9.10.5 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 l

QC.SKY.LINEl.FLUX   Flux of the strongest line of group l

QC.SKY.CONT.FLUX   Total flux of the continuum

QC.SKY.CONT.MAXDEV   Maximum (absolute value) of the derivative of the continuum spectrum

9.11 muse_astrometry

Compute an astrometric solution.

9.11.1 Description

Merge pixel tables from all IFUs, apply correction for differential atmospheric refraction (when necessary), 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.

There are two pattern matching algorithm implemented, which can be selected by chosing a positive or zero value of faccuracy.

In the first method (with a positive value of faccuracy), start using the search radius, and iteratively decrease it, until no duplicate detections are identified any more. Similarly, iterate the data accuracy (decrease it downwards from the mean positioning error) until matches are found. Remove the remaining unidentified objects.

The second method (when faccuracy is set to zero), iterates through all quadruples in both the detected objects and the catalogue, calculates the transformation and checks whether more than 80% of the detections match a catalog entry within the radius.

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

9.11.2 Input frames

Category	Type	min	Description
PIXTABLE_ASTROMETRY	raw	1	Pixel table of an astrometric field
ASTROMETRY_REFERENCE	calib	1	Reference table of known objects for astrometry
EXTINCT_TABLE	calib		Atmospheric extinction table (optional)
STD_RESPONSE	calib		Response curve as derived from standard star(s) (optional)
STD_TELLURIC	calib		Telluric absorption as derived from standard star(s) (optional)

9.11.3 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	**1.5**	Source detection sigma level to use. If this is negative, values between its absolute and 1.0 are tested with a stepsize of 0.1, to find an optimal solution.
radius	double	**3.**	Initial radius in pixels for pattern matching identification in the astrometric field.
faccuracy	double	**0.**	Factor of initial accuracy relative to mean positional accuracy of the measured positions to use for pattern matching. If this is set to zero, use the quadruples based method.
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).
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.

9.11.4 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

9.11.5   Quality control parameters

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

QC.ASTRO.NDET   Number of detected sources in output cube.

QC.ASTRO.LAMBDA   Wavelength of plane in combined cube that was used for object detection.

QC.ASTRO.POSk.X   Position of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.

QC.ASTRO.POSk.Y   Position of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.

QC.ASTRO.FWHMk.X   FWHM of source k in x-direction in output cube. If the FWHM measurement fails, this value will be -1.

QC.ASTRO.FWHMk.Y   FWHM of source k in y-direction in output cube. If the FWHM measurement fails, this value will be -1.

QC.ASTRO.FWHM.NVALID   Number of detected sources with valid FWHM in output cube.

QC.ASTRO.FWHM.MEDIAN   Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

QC.ASTRO.FWHM.MAD   Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

QC.SCIBASIC.LAMBDA.SHIFT.AVG   Average of ESO QC SCIBASIC LAMBDA SHIFT over all m IFUs

QC.SCIBASIC.LAMBDA.SHIFT.RMS   RMS of ESO QC SCIBASIC LAMBDA SHIFT over all m IFUs

QC.SCIBASIC.NSATURATED.AVG   Average of ESO QC SCIBASIC NSATURATED over all m IFUs

QC.SCIBASIC.NSATURATED.RMS   RMS of ESO QC SCIBASIC NSATURATED over all m IFUs

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

9.12 muse_scipost

Prepare reduced and combined science products.

9.12.1 Description

Sort input pixel tables into lists of files per exposure, merge pixel tables from all IFUs of each exposure.

Correct each exposure for differential atmospheric refraction (unless --lambdaref is far outside the MUSE wavelength range, or NFM is used which has a built-in corrector). Then the flux calibration is carried out, if a response curve was given in the input; it includes a correction of telluric absorption, if a telluric absorption correction file was given. If observations were done with AO and a RAMAN_LINES file was given, a procedure is run to clean the Raman scattering emission lines from the data. Next, the slice autocalibration is computed and the flux correction factors are applied to the pixel table (if --autocalib="deepfield"). If user-provided autocalibration is requested (--autocalib="user"), then the autocalibration is not computed on the input exposure but the autocalibration factors are read from the AUTOCAL_FACTORS table and applied directly to the data. Then the sky subtraction is carried out (unless --skymethod="none"), either directly subtracting an input sky continuum and an input sky emission lines (for --skymethod="subtract-model"), or (--skymethod="model") create a sky spectrum from the darkest fraction (--skymodel_fraction, after ignoring the lowest --skymodel_ignore as artifacts) of the field of view, then fitting and subtracting sky emission lines using an initial estimate of the input sky lines; then the continuum (residuals after subtracting the sky lines from the sky spectrum) is subtracted as well. If --save contains "skymodel", all sky-related products are saved for each exposure. Afterwards the data is corrected for the radial velocity of the observer (--rvcorr), before the input (or a default) astrometric solution is applied. Now each individual exposure is fully reduced; the pixel tables at this stage can be saved by setting "individual" in --save.

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

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

9.12.2 Input frames

Category	Type	min	Description
PIXTABLE_OBJECT	raw	1	Pixel table of a science object
EXTINCT_TABLE	calib	1	Atmospheric extinction table
STD_RESPONSE	calib	1	Response curve as derived from standard star(s)
STD_TELLURIC	calib		Telluric absorption correction as derived from standard star(s) (optional)
			Continued on next page

Category	Type	min	Description
– continued from previous page			
ASTROMETRY_WCS	calib		Astrometric solution derived from astrometric science frame (optional)
OFFSET_LIST	calib		List of coordinate offsets (and optional flux scale factors); can influence how the SKY_MASK is interpreted (optional)
FILTER_LIST	calib		File to be used to create field-of-view images. (optional)
OUTPUT_WCS	calib		WCS to override output cube location / dimensions (see data format chapter for details) (optional)
AUTOCAL_FACTORS	calib		Table with multiplicative factors applied to all slices (used only if --autocalib=user) (optional)
RAMAN_LINES	calib		List of Raman line transitions of O2 and N2 (optional)
SKY_LINES	calib		List of OH transitions and other sky lines (optional)
SKY_CONTINUUM	calib		Sky continuum to use (optional)
LSF_PROFILE	calib		Slice specific LSF parameters. (optional, more than one frame allowed)
SKY_MASK	calib		Sky mask to use; if it contains a WCS, it is aligned to the data before masking (optional)

9.12.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
save	string	**cube,skymodel**	Select output product(s) to save. Can contain one or more of "cube", "autocal", "skymodel", "individual", "positioned", "combined", and "stacked". If several options are given, they have to be comma-separated. ("cube": output cube and associated images, if this is not given, no final resampling is done at all -- "autocal": up to two additional output products related to the slice autocalibration -- "raman": up to four additional output products about the Raman light distribution for AO observations -- "skymodel": up to four additional output products about the effectively used sky that was subtracted with the "model" method -- "individual": fully reduced pixel table for each individual exposure -- "positioned": fully reduced and positioned pixel table for each individual exposure, the difference to "individual" is that here, the output pixel tables have coordinates in RA and DEC, and the optional offsets were applied; this is only useful, if both the relative exposure weighting and the final resampling are to be done externally -- "combined": fully reduced and combined pixel table for the full set of exposures, the difference to "positioned" is that all pixel tables are combined into one, with an added weight column; this is useful, if only the final resampling step is to be done separately -- "stacked": an additional output file in form of a 2D column-stacked image, i.e. x direction is pseudo-spatial, y direction is wavelength.)
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, 0.025" for NFM, 1.0 if data is in pixel units.

Continued on next page

Parameter	Type	Values **default**, other	Description
– continued from previous page			
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, 0.025" 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 the median median of the absolute median deviation.
crsigma	double	**15.**	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	string	**0.8,0.8**	Pixel down-scaling factor for the "drizzle" resampling method. Up to three, comma-separated, floating-point values can be given. If only one value is given, it applies to all dimensions, two values are interpreted as spatial and spectral direction, respectively, while three are taken as horizontal, vertical, and spectral.
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, sdpCube	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. "sdpCube" does some extra calculations to create FITS keywords for the ESO Science Data Products.
weight	string	**exptime**, exptime, fwhm, none	Type of weighting scheme to use when combining multiple exposures. "exptime" just uses the exposure time to weight the exposures, "fwhm" uses the best available seeing information from the headers as well, "none" preserves an existing weight column in the input pixel tables without changes.
Continued on next page			

Parameter	Type	Values **default**, other	Description
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.
autocalib	string	**none**, none, deepfield, user	The type of autocalibration to use. "none" switches it off, "deepfield" uses the revised MPDAF method that can be used for the reduction of mostly empty "Deep Fields", "user" searches for a user-provided table with autocalibration factors.
skymethod	string	**model**, none, subtract-model, model, simple	The method used to subtract the sky background (spectrum). Option "model" should work in all kinds of science fields: it uses a global sky spectrum model with a local LSF. "model" uses fluxes indicated in the SKY_LINES file as starting estimates, but re-fits them on the global sky spectrum created from the science exposure. If SKY_CONTINUUM is given, it is directly subtracted, otherwise it is created from the sky region of the science exposure. Option "subtract-model" uses the input SKY_LINES and SKY_CONTINUUM, subtracting them directly without re-fitting the fluxes, but still makes use of the local LSF, hence LSF_PROFILE is required. The inputs LSF_PROFILE and SKY_LINES are necessary for these two model-based methods; SKY_CONTINUUM is required for "subtract-model" and optional for "model"; SKY_MASK is optional for "model". Finally, option "simple" creates a sky spectrum from the science data, and directly subtracts it, without taking the LSF into account (LSF_PROFILE and input SKY files are ignored). It works on data that was not flux calibrated.
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).

<div align="center">Continued on next page</div>

– 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.
skymodel_fraction	double	**0.10**	Fraction of the image (without the ignored part) to be considered as sky. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 1.
skymodel_ignore	double	**0.05**	Fraction of the image to be ignored. If an input sky mask is provided, the fraction is applied to the regions within the mask. If the whole sky mask should be used, set this parameter to 0.
rvcorr	string	**bary**, bary, helio, geo, none	Correct the radial velocity of the telescope with reference to either the barycenter of the Solar System (bary), the center of the Sun (helio), or to the center of the Earth (geo).
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.

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.
OBJECT_RESAMPLED	Stacked image (if --save contains "stacked")
	Continued on next page

– continued from previous page	
Default file name	Description
PIXTABLE_REDUCED	Fully reduced pixel tables for each exposure (if --save contains "individual")
PIXTABLE_POSITIONED	Fully reduced and positioned pixel table for each individual exposure (if --save contains "positioned")
PIXTABLE_COMBINED	Fully reduced and combined pixel table for the full set of exposures (if --save contains "combined")
AUTOCAL_MASK	Created sky mask for autocalibration (if --autocalib=deepfield and --save contains "autocal" but no input SKY_MASK was given)
AUTOCAL_FACTORS	Table with factors applied during autocalibration (if --autocalib=deepfield and --save contains "autocal")
RAMAN_IMAGES	Images for Raman correction diagnostics (if an input RAMAN_LINES was given, the instrument used AO and --save contains "raman"). Extensions are: DATA: model of the Raman light distribution in the field of view (arbitrary units), RAMAN_IMAGE_O2: reconstructed image in the O2 band, SKY_MASK_O2: sky mask used for the O2 band, RAMAN_IMAGE_N2: reconstructed image in the N2 band, SKY_MASK_N2: sky mask used for the N2 band, RAMAN_FIT_O2: model of Raman flux distribution in the O2 band, RAMAN_FIT_N2: model of Raman flux distribution in the N2 band.
SKY_IMAGE	Reconstructed sky image which is then used to create the SKY_MASK (if --skymethod=model and --save contains "skymodel")
SKY_MASK	Created sky mask (if --skymethod=model and --save contains "skymodel")
SKY_SPECTRUM	Sky spectrum within the sky mask (if --skymethod=model and --save contains "skymodel")
SKY_LINES	Estimated sky line flux table (if --skymethod=model and --save contains "skymodel")
SKY_CONTINUUM	Estimated continuum flux spectrum (if --skymethod=model and --save contains "skymodel")

9.12.5 Quality control parameters

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

`QC.SCIPOST.NDET`   Number of detected sources in output cube.
`QC.SCIPOST.LAMBDA`   Wavelength of plane in combined cube that was used for object detection.
`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.SCIPOST.FWHM.NVALID`   Number of detected sources with valid FWHM in output cube.

`QC.SCIPOST.FWHM.MEDIAN`   Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.SCIPOST.FWHM.MAD`   Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in output cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.AIRM.MEAN`   Average of ESO TEL AIRM START and ESO TEL AIRM END

`QC.AMBI_REQ`   Value of ESO QC AMBI_REQ if present. Otherwise set to 1.0.

`QC.NUMBER_SRC`   Number of identified sources

`QC.LSHIFT_MAX`   Value of ESO WDEL_MAX if present. Otherwise set to 0.0.

`QC.NUM_SAT`   Value of ESO QC NUM_SAT if present. Otherwise set to 0.

`QC.SKY_YN`   Set to Y if SKY_CONTINUUM is present and skymethod is subtract-model or model. Otherwise set to N.

`QC.LSF_YN`   Set to Y if LSF_PROFILE input frames are present. Otherwise set to N.

`QC.SINGLE_YN`   Set to Y if ESO TPL NEXP has the value 1. Otherwise set to N.

`QC.SKY_FILE`   The filename of SKY_CONTINUUM if present.

`QC.SKY_CONT_FLUX`   Value of ESO QC SCIPOST CONT FLUX in SKY_CONTINUUM. Otherwise set to -999.0.

`QC.SKY_NLINES`   Count of number of ESO QC SCIPOST LINE[0-9]+ FLUX keywords in SKY_LINES product.

`QC.DELTA.TIME`   Difference between ESO DRS MUSE SCIBASIC LAMBDA MJD from wavelength calibration (muse_scibasic) and MJD-OBS in cube header.

`QC.HISTO_MAX`   Maximum of histogram of flux distribution in the IMAGE_FOV.

`QC.HISTO_17`   Value of ESO QC HISTO_MAX divided by 50.0 corresponding to lower cut of -1.7 dex.

`QC.SCIPOST.LOWLIMIT`   Low limit in the white light considered as sky, used to create this mask, everything lower are likely artifacts.

`QC.SCIPOST.THRESHOLD`   Threshold in the white light considered as sky, used to create this mask, higher values are likely objects in the field.

`QC.SCIPOST.RAMAN.SPATIAL.XX`   2D Polynomial x^2 coefficient

`QC.SCIPOST.RAMAN.SPATIAL.XY`   2D Polynomial xy coefficient

`QC.SCIPOST.RAMAN.SPATIAL.YY`   2D Polynomial y^2 coefficient

`QC.SCIPOST.RAMAN.SPATIAL.X`   2D Polynomial x coefficient

`QC.SCIPOST.RAMAN.SPATIAL.Y`   2D Polynomial y coefficient

`QC.SCIPOST.RAMAN.FLUX.O2`   Computed average Raman scattered flux in the O2 band (around 6484 Angstrom)

`QC.SCIPOST.RAMAN.FLUX.N2`   Computed average Raman scattered flux in the N2 band (around 6827 Angstrom)

`QC.SCIPOST.LINEl.NAME`   Name of the strongest line in group l

`QC.SCIPOST.LINEl.AWAV`   Wavelength (air) of the strongest line of group l

`QC.SCIPOST.LINEl.FLUX`   Flux of the strongest line of group l

`QC.SCIPOST.CONT.FLUX`   Total flux of the continuum

`QC.SCIPOST.CONT.MAXDEV`   Maximum (absolute value) of the derivative of the continuum spectrum

9.13 muse_exp_align

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

9.13.1 Description

Compute the coordinate offset for each input field-of-view image with respect to a reference. The created list of coordinate offsets can then be used in muse_exp_combine as the field coordinate offsets to properly align the exposures during their combination. The source positions used to compute the field offsets are obtained by detecting point sources in each of the input images, unless the source detection is overridden and an input source list is available for each input field-of-view image. In this latter case the input source positions are used to calculate the field offsets.

Note this recipe has been superseeded by muse_alignment and muse_detection, it is being kept for backward compatibility.

9.13.2 Input frames

Category	Type	min	Description
IMAGE_FOV	raw	2	Input field-of-view images (more than 2 frames allowed)
SOURCE_LIST	calib		Input list of sources for a field-of-view image (optional, more than one frame allowed)

9.13.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
rsearch	string	**30.,4.,2.,0.8**	Search radius (in arcsec) for each iteration of the offset computation.
nbins	int	**60**	Number of bins to use for 2D histogram on the first iteration of the offset computation.
weight	boolean	**true**	Use weighting.
fwhm	double	**5.**	FWHM in pixels of the convolution filter.
threshold	double	**15.**	Initial intensity threshold for detecting point sources. If the value is negative or zero the threshold is taken as sigma above median background MAD. If it is larger than zero the threshold is taken as absolute background level.
bkgignore	double	**0.05**	Fraction of the image to be ignored.
			Continued on next page

Parameter	Type	Values **default**, other	Description
			– continued from previous page
bkgfraction	double	0.10	Fraction of the image (without the ignored part) to be considered as background.
step	double	0.5	Increment/decrement of the threshold value in subsequent iterations.
iterations	int	100000	Maximum number of iterations used for detecting sources.
srcmin	int	5	Minimum number of sources which must be found.
srcmax	int	80	Maximum number of sources which may be found.
roundmin	double	-1.	Lower limit of the allowed point-source roundness.
roundmax	double	1.	Upper limit of the allowed point-source roundness.
sharpmin	double	0.2	Lower limit of the allowed point-source sharpness.
sharpmax	double	1.	Upper limit of the allowed point-source sharpness.
bpixdistance	double	5.	Minimum allowed distance of a source to the closest bad pixel in pixel. Detected sources which are closer to a bad pixel are not taken into account when computing the field offsets. This option has no effect if the source positions are taken from input catalogs.

9.13.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
EXPOSURE_MAP	Map of the total exposure time of the combined field-of-view (only if enabled).
PREVIEW_FOV	Preview image of the combined field-of-view.
SOURCE_LIST	List of parameters of the detected point sources.
OFFSET_LIST	List of computed coordinate offsets.

9.13.5 Quality control parameters

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

`QC.EXPALIGN.EXPTIME.MIN`   Minimum exposure time of the combined field-of-view.
`QC.EXPALIGN.EXPTIME.MAX`   Maximum exposure time of the combined field-of-view.
`QC.EXPALIGN.EXPTIME.AVG`   Average exposure time of the combined field-of-view.
`QC.EXPALIGN.EXPTIME.MED`   Median exposure time of the combined field-of-view.
`QC.EXPALIGN.SRC.POSITIONS`   Origin of the source positions.
`QC.EXPALIGN.NDET`   Number of detected sources.
`QC.EXPALIGN.BKG.MEDIAN`   Median value of background pixels.

`QC.EXPALIGN.BKG.MAD`   Median absolute deviation of the background pixels.

`QC.EXPALIGN.THRESHOLD`   Detection threshold used for detecting sources.

`QC.EXPALIGN.NDETi`   Number of detected sources for input image i

`QC.EXPALIGN.NMATCHi`   Median number of matches of input image i with other images

`QC.EXPALIGN.NMATCH.MIN`   Minimum of the median number of matches for all input images

`QC.EXPALIGN.NOMATCH`   Number of input images that do not have any matches with other images

`QC.EXPALIGN.OFFSET.RA.MIN`   [arcsec] RA minimum offset.

`QC.EXPALIGN.OFFSET.RA.MAX`   [arcsec] RA maximum offset.

`QC.EXPALIGN.OFFSET.RA.MEAN`   [arcsec] RA mean offset.

`QC.EXPALIGN.OFFSET.RA.STDEV`   [arcsec] Standard deviation of RA offsets.

`QC.EXPALIGN.OFFSET.DEC.MIN`   [arcsec] DEC minimum offset.

`QC.EXPALIGN.OFFSET.DEC.MAX`   [arcsec] DEC maximum offset.

`QC.EXPALIGN.OFFSET.DEC.MEAN`   [arcsec] DEC mean offset.

`QC.EXPALIGN.OFFSET.DEC.STDEV`   [arcsec] Standard deviation of DEC offsets.

9.14 muse_detection

Detect sources in field-of-view images, for the purpose of computing coordinate offsets between cubes.

9.14.1 Description

Detect sources in field-of-view images, the output source table is used as input for the muse_alignment recipe.

9.14.2 Input frames

Category	Type	min	Description
IMAGE_FOV	raw	1	Input field-of-view images (more than 1 frame allowed)

9.14.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
border_distance_pxl	int	**3**	Width ignored region around bad pixels, in pixels.
start_threshold_sigma	double	**1.5**	The initial sigma for catalogue inclusion.
threshold_limit	double	**0.5**	The limit sigma for catalogue inclusion.
max_iter	int	**150**	Max number of iterations.
source_min	int	**2**	Min number of sources accepted.
source_max	int	**150**	Max number of sources in catalogue.

9.14.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
EXPOSURE_MAP	Map of the total exposure time of the combined field-of-view (only if enabled).
PREVIEW_FOV	Preview image of the combined field-of-view.
SOURCE_LIST	List of parameters of the detected point sources.
OFFSET_LIST	List of computed coordinate offsets.

9.14.5 Quality control parameters

There are no quality control parameters for the **muse_detection** products.

9.15 muse_alignment

Compute coordinate offsets between cubes.

9.15.1 Description

Compute coordinate offsets between cubes, using source list from muse_detection.

9.15.2 Input frames

Category	Type	min	Description
SOURCE_LIST	calib	2	Input list of sources for a field-of-view image

9.15.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
search_radii	string	**30.,4.,2.,0.8**	Search radius (in arcsec) for each iteration of the offset computation.
threshold_on_mag	float	**1.5**	Maximum magnitude difference between frames to be considerd the same object.
keep_n_brightest	int	**10**	Use only the n brightest sources in each source list for comparison.
weight	bool	**TRUE**	Use weights?

9.15.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
EXPOSURE_MAP	Map of the total exposure time of the combined field-of-view (only if enabled).
PREVIEW_FOV	Preview image of the combined field-of-view.
SOURCE_LIST	List of parameters of the detected point sources.
OFFSET_LIST	List of computed coordinate offsets.

9.15.5 Quality control parameters

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

`QC.EXPALIGN.EXPTIME.MIN`   Minimum exposure time of the combined field-of-view.

`QC.EXPALIGN.EXPTIME.MAX`   Maximum exposure time of the combined field-of-view.

`QC.EXPALIGN.EXPTIME.AVG`   Average exposure time of the combined field-of-view.

`QC.EXPALIGN.EXPTIME.MED`   Median exposure time of the combined field-of-view.

`QC.EXPALIGN.SRC.POSITIONS`   Origin of the source positions.

`QC.EXPALIGN.NDET`   Number of detected sources.

`QC.EXPALIGN.BKG.MEDIAN`   Median value of background pixels.

`QC.EXPALIGN.BKG.MAD`   Median absolute deviation of the background pixels.

`QC.EXPALIGN.THRESHOLD`   Detection threshold used for detecting sources.

`QC.EXPALIGN.NDETi`   Number of detected sources for input image i

`QC.NSTAT`   Detection statistics quality flag. Set to 1 if detections across images are uniform, balanced, or variable within acceptable limits; 0 otherwise.

`QC.EXPALIGN.NMATCHi`   Median number of matches of input image i with other images

`QC.EXPALIGN.NMATCH.MIN`   Minimum of the median number of matches for all input images

`QC.EXPALIGN.NOMATCH`   Number of input images that do not have any matches with other images

`QC.EXPALIGN.OFFSET.RA.MIN`   [arcsec] RA minimum offset.

`QC.EXPALIGN.OFFSET.RA.MAX`   [arcsec] RA maximum offset.

`QC.EXPALIGN.OFFSET.RA.MEAN`   [arcsec] RA mean offset.

`QC.EXPALIGN.OFFSET.RA.STDEV`   [arcsec] Standard deviation of RA offsets.

`QC.EXPALIGN.OFFSET.DEC.MIN`   [arcsec] DEC minimum offset.

`QC.EXPALIGN.OFFSET.DEC.MAX`   [arcsec] DEC maximum offset.

`QC.EXPALIGN.OFFSET.DEC.MEAN`   [arcsec] DEC mean offset.

`QC.EXPALIGN.OFFSET.DEC.STDEV`   [arcsec] Standard deviation of DEC offsets.

9.16 muse_exp_combine

Combine several exposures into one datacube.

9.16.1 Description

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

9.16.2 Input frames

Category	Type	min	Description
PIXTABLE_REDUCED	raw	2	Input pixel tables (more than 2 frames allowed)
OFFSET_LIST	calib		List of coordinate offsets (and optional flux scale factors) (optional)
FILTER_LIST	calib		File to be used to create field-of-view images. (optional)
OUTPUT_WCS	calib		WCS to override output cube location / dimensions. See Section A for details. (optional)

9.16.3 Recipe parameters

Parameter	Type	Values **default**, other	Description
save	string	**cube**	Select output product(s) to save. Can contain one or more of "cube" (output cube and associated images; if this is not given, no resampling is done at all) or "combined" (fully reduced and combined pixel table for the full set of exposures; this is useful, if the final resampling step is to be done again separately). If several options are given, they have to be comma-separated.
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, 0.025" for NFM, 1.0 if data is in pixel units.

Continued on next page

Parameter	Type	Values **default**, other	Description
		– continued from previous page	
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, 0.025" 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 the median median of the absolute median deviation.
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	string	**0.6,0.6**	Pixel down-scaling factor for the "drizzle" resampling method. Up to three, comma-separated, floating-point values can be given. If only one value is given, it applies to all dimensions, two values are interpreted as spatial and spectral direction, respectively, while three are taken as horizontal, vertical, and spectral.
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, sdpCube	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. "sdpCube" does some extra calculations to create FITS keywords for the ESO Science Data Products.
			Continued on next page

Parameter	Type	Values **default**, other	Description
– continued from previous page			
weight	string	**exptime**, exptime, fwhm, header, none	Type of weighting scheme to use when combining multiple exposures. "exptime" just uses the exposure time to weight the exposures, "fwhm" uses the best available seeing information from the headers as well, "header" queries ESO.DRS.MUSE.WEIGHT of each input file instead of the FWHM, and "none" preserves an existing weight column in the input pixel tables without changes.
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).

9.16.4 Product frames

The following product frames are created by the recipe:

Default file name	Description
DATACUBE_FINAL	Output datacube (if --save contains "cube")
IMAGE_FOV	Field-of-view images corresponding to the "filter" parameter (if --save contains "cube").
PIXTABLE_COMBINED	Combined pixel table (if --save contains "combined")

9.16.5 Quality control parameters

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

`QC.EXPCOMB.NDET` Number of detected sources in combined cube.

`QC.EXPCOMB.LAMBDA` Wavelength of plane in combined cube that was used for object detection.

`QC.EXPCOMB.POSk.X` Position of source k in x-direction in combined cube. If the FWHM measurement fails, this value will be -1.

`QC.EXPCOMB.POSk.Y` Position of source k in y-direction in combined cube. If the FWHM measurement fails, this value will be -1.

`QC.EXPCOMB.FWHMk.X`   FWHM of source k in x-direction in combined cube. If the FWHM measurement fails, this value will be -1.

`QC.EXPCOMB.FWHMk.Y`   FWHM of source k in y-direction in combined cube. If the FWHM measurement fails, this value will be -1.

`QC.EXPCOMB.FWHM.NVALID`   Number of detected sources with valid FWHM in combined cube.

`QC.EXPCOMB.FWHM.MEDIAN`   Median FWHM of all sources with valid FWHM measurement (in x- and y-direction) in combined cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.EXPCOMB.FWHM.MAD`   Median absolute deviation of the FWHM of all sources with valid FWHM measurement (in x- and y-direction) in combined cube. If less than three sources with valid FWHM are detected, this value is zero.

`QC.AIRM.MEAN`   Average of ESO TEL AIRM START and ESO TEL AIRM END

`QC.AMBI_REQ`   Value of ESO QC AMBI_REQ if present. Otherwise set to 1.0.

`QC.NUMBER_SRC`   Number of identified sources

`QC.NUM_MATCH`   Value of ESO QC EXPALIGN NMATCH MIN if present. Otherwise set to -1.

`QC.NUM_NOMATCH`   Value of ESO QC EXPALIGN NOMATCH if present. Otherwise set to 0.

`QC.OFFSET`   Value of sqrt(dr*dr+dd*dd) where dr (ESO QC EXPALIGN OFFSET RA MEAN) and dd (ESO QC EXPALIGN OFFSET DEC MEAN) were converted to degrees.

`QC.TARGET`   Value of ESO QC TARGET with spaces replaced by underscores.

`QC.LSF_YN`   Set to Y if LSF_PROFILE input frames are present. Otherwise set to N.

`QC.SINGLE_YN`   Set to Y if ESO TPL NEXP has the value 1. Otherwise set to N.

`QC.DELTA.TIME`   Difference between ESO DRS MUSE SCIBASIC LAMBDA MJD from wavelength calibration (muse_scibasic) and MJD-OBS in cube header.

`QC.HISTO_MAX`   Maximum of histogram of flux distribution in the IMAGE_FOV.

`QC.HISTO_17`   Value of ESO QC HISTO_MAX divided by 50.0 corresponding to lower cut of -1.7 dex.

10 Data Processing Details

This section provides an algorithmic view on the MUSE pipeline recipes. It complements the reference–manual–like documentation of the recipes presented in Section 9 with detailed descriptions of various aspects of the pipeline.

10.1 Algorithmic Details

We refer the reader to the MUSE pipeline paper [RD14] (see Section 1.5) for detailed information on the algorithms used by the pipeline. In the following we give further information on some selected topics.

10.1.1 muse_exp_align alignment

The algorithm detects sources in the exposures, and iteratively finds the best coordinate offsets to apply to each exposure in order to match the detections.

The source detection in each frame is done by finding objects above an intensity threshold (threshold) above the background; if negative or zero the threshold is taken as sigma above median background Median Absolute Deviation (MAD); if positive, the threshold is taken as absolute background level. The computation of the background MAD is regulated by other 2 parameters: bkgignore (fraction of the image to ignore) and bkgfraction fraction of the image (without the ignored part) to be considered as background.

If the number of sources is larger than the maximum allowed (srcmax) the threshold is increased by the amount specified by the parameter step . If the number of sources is smaller than the minimum requested (srcmin), the threshold is decreased by the amount specified by the parameter step . This is repeated until the number of sources is within the selected range or until the maximum number of iterations (iterations) is reached.

The match is done on the basis of closest neighbors, the maximum allowed distance in arcsec at each iteration is regulated by the parameter rsearch . The number of search iterations is determined by the number of elements in rsearch . These iterations are not to be confused with the source detection iterations.

First, the algorithm constructs a 2D histogram of matches in a RA/DEC grid defined by the first element of rsearch (amplitude of the grid in each direction) and nbins (elements of the grid, in each direction). Typically, step is the double of the first element in rsearch . The peak of the histogram determines the the zeroth order offsets. Then, the identified sources are are matched on the basis of the closest neighbors. The offsets are computed so that the distance between matched sources is minimized. The minimization is done for all images simultaneously, so that the exposures are simultaneously aligned with respect to each other, and not with respect to a single reference frame. Second, the iterative process starts. At each iteration the next distance in rsearch is used.

MUSE Pipeline User Manual

Doc. Number:	ESO-264503
Doc. Version:	1
Released on:	2022-03-10
Page:	128 of 159

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 up to 7 additional extensions depending on the instrument mode and the active subsystems, e.g. whether the MUSE SGS or AO 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. Note that some of the parameters monitored by the SGS system make it into keywords in the primary headers of the science images, the most important ones are `OCS.SGS.FWHM.*` and `OCS.SGS.FLUX.*`, that can be used to monitor the image quality and transparency during the exposure. However, the image quality monitoring keywords should be taken with caution because the SGS filter is very wide, and it is thus affected by atmospheric dispersion.

- 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 up to 31 (for science exposures) contain information from the SGS, Auto Guider (for WFM), the SPARTA subsystems (AO only), and the Atmospheric Site Monitor, taken in parallel to the science exposures. Note that only the extension of the active subsystems will be present in the raw data file.

  When the SGS is activated it will record images with the TCCD and produce stack median images every approx. 2 min. These median images can be averaged over the entire science exposure to give a deeper image of the region surrounding the MUSE FOV.

  - Extension SGS_IMG (WFM mode):
    an image of size 1024 × 1024 contains the average of all the stacked median images taken during the science exposure.

- **Extension SGS_CUBE (WFM mode):**
  a cube of $1024 \times 1024 \times N$ pixels, containing all N stacked median images taken during the science exposures.

- **Extension SGS_DATA (WFM mode):**
  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.

- **Extension AG_DATA (WFM mode):**
  a FITS table containing information about the guide star (GS) from the telescope Auto Guider (AG). It contains three columns giving the time of the measurement, and the FWHM in x and y, with a typical periodicity of 11 to 45 seconds. If there is no SGS star these data can be used to check the image quality during the exposure.

- **Extension ASM_DATA:**
  a FITS table containing the Atmospheric Site Monitoring (ASM) data.

- **Extension SPARTA_ATM_DATA (AO modes):**
  a FITS table containing the atmospheric turbulence information as determined by the SPARTA Real Time Computer.

- **Extension SPARTA_CN2_DATA (AO modes):**
  a FITS table containing the atmospheric turbulence profile estimated by the SPARTA Real Time Computer.

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	float	Wavelength [Angstrom]
flux	float	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 (optional column)

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: don't 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'`
  Catalogue of OH transitions and other sky lines,
  muse_create_sky: Estimated sky line flux table,
  muse_scipost: Estimated sky line flux table (if --skymethod=model and --save contains "skymodel")

A.2.3 RAMAN_LINES

Description

This type of file contains a binary table with the relative fluxes on the raman emission lines.

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: don't use)

Frame tags

- **RAMAN_LINES**: `PRO.CATG=='RAMAN_LINES'`
  Catalogue of OH transitions and other sky lines

A.2.4 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]
filter	string	Filter name used for column mag
mag	double	Object (Vega) magnitude [mag]

Frame tags

- **ASTROMETRY_REFERENCE**: `PRO.CATG=='ASTROMETRY_REFERENCE'`
  Catalogue of astrometry reference stars

A.2.5 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.2.6 BADPIX_TABLE

Description

This is a FITS table, typically with 24 extensions. It 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
		Continued on next page

– continued from previous page		
Column name	Type	Description
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.2.7 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.2.8 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.2.9 TELLURIC_REGIONS

Description

This FITS table defines wavelength ranges 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 [RD04] for FITS files with data, bad pixel maps, and variance. The units of the extensions are given by the standard BUNIT keyword.

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

FITS extensions

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

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

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

Frame tags

- **MASTER_BIAS**:
  muse_bias: Master bias

- **MASTER_DARK**:
  muse_dark: Master dark

- **MODEL_DARK**:
  muse_dark: Model of the master dark (if --model=true).

- **MASTER_FLAT**:
  muse_flat: Master flat

- **ARC_RED_LAMP**:
  muse_wavecal: Reduced ARC image, per lamp (if --saveimages=true)

- **MASK_REDUCED**:
  muse_geometry: Reduced pinhole mask images

- **MASK_COMBINED**:
  muse_geometry: Combined pinhole mask image

- **SKY_IMAGE**:
  muse_create_sky: White-light image used to create the sky mask,
  muse_scipost: Reconstructed sky image which is then used to create the SKY_MASK (if --skymethod=model and --save contains "skymodel")

- **MASTER_AMPL**:
  muse_ampl: Combined master AMPL image, written if --savemaster=true

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

- **OBJECT_RESAMPLED**:
  muse_scibasic: Re-sampled 2D image for OBJECT input (if --resample=true),
  muse_scipost: Stacked image (if --save contains "stacked"),
  muse_scipost_make_cube: Stacked image (if --stacked=true)

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

- **STD_RESAMPLED**:
  muse_scibasic: Re-sampled 2D image for STD input (if --resample=true)

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

- **SKY_RESAMPLED**:
  muse_scibasic: Re-sampled 2D image for SKY input (if --resample=true)

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

- **ASTROMETRY_RESAMPLED**:
  muse_scibasic: Re-sampled 2D image for ASTROMETRY input (if --resample=true)

- **REDUCED_RESAMPLED**:
  muse_scibasic: Re-sampled 2D image (if --resample=true)

- **IMAGE_FOV**:
  muse_scipost: Field-of-view images corresponding to the "filter" parameter.,
  muse_exp_combine: Field-of-view images corresponding to the "filter" parameter (if --save contains "cube").,
  muse_scipost_make_cube: Field-of-view images corresponding to the "filter" parameter.

- **EXPOSURE_MAP**:
  muse_exp_align: Map of the total exposure time of the combined field-of-view (only if enabled).

- **PREVIEW_FOV**:
  muse_exp_align: Preview image of the combined field-of-view.

A.3.2 PIXEL_TABLE

Description

In the reduction approach of the MUSE pipeline, data need to be kept un-resampled until the very last step. The pixel tables used for this purpose can be saved at each intermediate reduction step and hence contain lists of pixels together with output coordinates and values.

By default, they are saved as multi-extension FITS images, where each extension corresponds to one table column. The name of the column is saved in the EXTNAME keyword, the unit in the standard BUNIT keyword.

The units evolve through the processing. The units of the spatial coordinates are "pix" at the beginning, which signifies pixels of nominal size within the MUSE field of view, relative to an approximate center. The change to "rad" units when projecting to the tangent plane of the gnomonic coordinate representation. These are native spherical coordinates [RD10] and are not directly interpretable within the MUSE field of view. The final stage are in "deg" units, which signify a full astrometric calibration and each pixel is assigned relative RA and DEC in degrees. The data units change from "count" (photo electrons) to physical units during flux calibration.

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

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

FITS extensions

The default pixel table format (i.e. `MUSE_PIXTABLE_SAVE_AS_TABLE` is not set) is:

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

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

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

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

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

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

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

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

If `MUSE_PIXTABLE_SAVE_AS_TABLE=1` each of the image extensions is stored as a table column, where the name of the column corresponds to the extension name of the respective image extension of the default format.

Frame tags

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

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

- **PIXTABLE_STD**:
  muse_scibasic: Output pixel table for STD input

- **PIXTABLE_SKY**:
  muse_scibasic: Output pixel table for SKY input

- **PIXTABLE_ASTROMETRY**:
  muse_scibasic: Output pixel table for ASTROMETRY input

- **PIXTABLE_REDUCED**:
  muse_scibasic: Output pixel table,
  muse_scipost: Fully reduced pixel tables for each exposure (if --save contains "individual"),
  muse_scipost_raman: Output pixel table for raman subtraction.,
  muse_scipost_subtract_sky: Output pixel table(s) for sky subtraction.,
  muse_scipost_subtract_sky_simple: Output pixel table(s) after simple sky subtraction.,
  muse_scipost_correct_rv: RV corrected pixel table

- **PIXTABLE_POSITIONED**:
  muse_scipost: Fully reduced and positioned pixel table for each individual exposure (if --save contains "positioned")

- **PIXTABLE_COMBINED**:
  muse_scipost: Fully reduced and combined pixel table for the full set of exposures (if --save contains "combined"),
  muse_exp_combine: Combined pixel table (if --save contains "combined"),
  muse_scipost_combine_pixtables: Combined pixel table

A.3.3 DATACUBE

Description

Two FITS NAXIS=3 cubes in two extensions for data values and variance. A bad pixel is represented by a NaN value in the data and variance extensions. Such data cubes follow the ESO specification [RD04] for FITS files with data, bad pixel maps, and variance. The units of the extensions are given by the standard BUNIT keyword.

They can have two-dimensional image extensions, of the same type as IMAGE_FOV. For these, the EXTNAME will be called the same as the filter function that was used to create it. (Depending on recipe parameters, additional filtername_STAT extensions may be present to represent the variance of the images. These images then follow the ESO specification [RD04].) If the image was created using a filter-function with known photometric zero-points, these are propagated to the header of the output image, as DRS.MUSE.FILTER.ZPVEGA and DRS.MUSE.FILTER.ZPAB. The filter fraction DRS.MUSE.FILTER.FRACTION describes, how much of the filter area was actually covered by MUSE data (only if this is above 99%, an image should be trusted for photometry).

FITS extensions

- `'DATA'`: 3D FITS image (float)
  Data values

- `'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_SKYFLAT**:
  muse_twilight: Cube of combined twilight sky-flat exposures

- **TWILIGHT_CUBE**:
  muse_twilight: Smoothed cube of twilight sky

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

- **DATACUBE_ASTROMETRY**:
  muse_astrometry: Reduced astrometry field exposure

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

- **RAMAN_IMAGES**:
  muse_scipost: Images for Raman correction diagnostics (if an input RAMAN_LINES was given, the instrument used AO and --save contains "raman"). Extensions are: DATA: model of the Raman light distribution in the field of view (arbitrary units), RAMAN_IMAGE_O2: reconstructed image in the O2 band, SKY_MASK_O2: sky mask used for the O2 band, RAMAN_IMAGE_N2: reconstructed image in the N2 band, SKY_MASK_N2: sky mask used for the N2 band, RAMAN_FIT_O2: model of Raman flux distribution in the O2 band, RAMAN_FIT_N2: model of Raman flux distribution in the N2 band.,
  muse_scipost_raman: Images for Raman correction diagnostics (if an input RAMAN_LINES was given, the instrument used AO and --save contains "raman"). Extensions are: DATA: model of the Raman light distribution in the field of view (arbitrary units), RAMAN_IMAGE_O2: reconstructed image in the O2 band, SKY_MASK_O2: sky mask used for the O2 band, RAMAN_IMAGE_N2: reconstructed image in the N2 band, SKY_MASK_N2: sky mask used for the N2 band, RAMAN_FIT_O2: model of Raman flux distribution in the O2 band, RAMAN_FIT_N2: model of Raman flux distribution in the N2 band.

A.3.4 EURO3DCUBE

Description

Euro3D format. See Format Definition Document [RD05] 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
		Continued on next page

Column name	Type	Description
– continued from previous page		
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
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 data cube,
  muse_exp_combine: Output data cube (if --save contains "cube"),
  muse_scipost_make_cube: Output data cube

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
Continued on next page		

Column name	Type	Description
– continued from previous page		
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: Trace table

A.3.6 TRACE_SAMPLES

Description

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

FITS extensions

- FITS table

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_PROFILE

Description

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

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

The pipeline automatically detects the format and continues processing accordingly.

FITS extensions

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

- FITS table

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
lsf_width	double array	LSF gauss-hermitian width
hermit3	double array	3th order hermitian coefficient
hermit4	double array	4th order hermitian coefficient
hermit5	double array	5th order hermitian coefficient
hermit6	double array	6th order hermitian coefficient

Frame tags

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

A.3.10 GEOMETRY_TABLE

Description

This file provides the relative location of each slice in the MUSE field of view. It contains one table of 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 (IFU / channel) 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]

Frame tags

- **GEOMETRY_UNSMOOTHED**:
  muse_geometry: Relative positions of the slices in the field of view (unsmoothed)

- **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]
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 SKY_MASK

Description

This can be an input and/or an output file of a pipeline recipe. It is a 2D FITS image of type integer where values of 0 are interpreted as locations of objects and a value of 1 are locations of sky.

The header of the image contains a FITS WCS which can come in two flavours:

- It can be tied to the pixel grid defined by the GEOMETRY_TABLE. In this case, the type is "PIXEL" and the unit is "pixel" and the resulting spatial coordinates range from -150 to +150 pixels.

- It can be a fully defined celestial WCS, in gnomonic projection (type "--TAN" and unit "deg"). This can be used as input to give more precise sky locations as defined by external data. In this case, the user has to make sure to give any coordinate offsets using an OFFSET_LIST valid for the exposure(s) in question.

In both cases, the pipeline only recognizes the CDi_j matrix system of the FITS WCS system (PCi_j cannot be used).

Frame tags

- **SKY_MASK**:
  muse_create_sky: Created sky mask,
  muse_scipost: Created sky mask (if --skymethod=model and --save contains "skymodel")

- **AUTOCAL_MASK**:
  muse_scipost: Created sky mask for auto-calibration (if --autocalib=deepfield and --save contains "autocal" but no input SKY_MASK was given)

A.3.13 AUTOCAL_RESULTS

Description
This FITS table contains the results of the slice auto-calibration used for deep fields. In particular, it contains the correction factors for each wavelength range, slice, and IFU. The entries ESO.DRS.MUSE.LAMBDAi.MIN and ESO.DRS.MUSE.LAMBDAi.MAX in the FITS header of the table give the wavelength ranges used during auto-calibration for each bin.

FITS extensions

- FITS table

Column name	Type	Description
ifu	int	IFU number
sli	int	Slice number
quad	int	Wavelength range bin number
npts	int	Number of points used to compute the correction factor
corr	double	Correction factor for the slice in this wavelength range

Frame tags

- **AUTOCAL_FACTORS**:
  muse_scipost: Table with factors applied during auto-calibration (if --autocalib=deepfield and --save contains "autocal")

A.3.14 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 (if --skymethod=model and --save contains "sky-model")

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

A.3.15 STD_RESPONSE

Description

MUSE flux response table.

In addition to the three main columns, this table may contain additional entries related to the throughput computed from the response curve ("throughput") and estimates of the response and its error that were not smoothed ("response_unsmoothed" and "resperr_unsmoothed").

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.3.16   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)

A.3.17   STD_FLUXES

Description
2D Image containing measurements of flux integration of all stars detected in a standard star field. This is mainly thought to be used for debugging. The image contains a spectral axis (axis 1) with corresponding WCS

information. Axis 2 is the arbitrary numbering of stars detected in the field. Several DRS.MUSE.FLUX.* keywords in the output header contain information regarding each object (x and y position in the corresponding data cube, approximate celestial position, and integrated flux); their numbering corresponds to the axis 2 coordinate. Another FITS keyword (DRS.MUSE.FLUX.NSEL) gives the number of the object that was selected as standard star by the pipeline.

FITS extensions

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

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

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

Frame tags

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

A.3.18 AMPL_CONVOLVED

Description

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

FITS extensions

- `'PHOTONS'`: 2D FITS image (float)
  Photon counts [ph]

- `'ENERGY'`: 2D FITS image (float)
  Per-pixel energy [J]

Frame tags

- **AMPL_CONVOLVED**:
  muse_ampl: Combined and convolved master AMPL image

A.3.19 OFFSET_LIST

Description

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

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

RA_OFFSET = RA(measured) - RA(reference)

DEC_OFFSET = DEC(measured) - DEC(reference)

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

FITS extensions

- FITS table

Column name	Type	Description
DATE_OBS	string	Date and time at the start of the exposure
		Continued on next page

Column name	Type	Description
		– continued from previous page
MJD_OBS	double	MJD at the start of the exposure (optional column) [s]
RA_OFFSET	double	Right ascension offset in degrees [deg]
DEC_OFFSET	double	Declination offset in degrees [deg]
FLUX_SCALE	double	(Relative) flux scaling of the given exposure (optional column)

Frame tags

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

A.3.20 SOURCE_LIST

Description

List of source positions detected on a MUSE field-of-view image.

FITS extensions

- FITS table

Column name	Type	Description
Id	int	Source identifier
X	double	X-position of the source in image coordinates [pix]
Y	double	Y-position of the source in image coordinates [pix]
Flux	double	Source flux (optional column)
Sharpness	double	Source sharpness value (optional column)
Roundness	double	Source roundness value (optional column)
RA	double	Source right ascension in degrees [deg]
DEC	double	Source declination in degrees [deg]
RA_CORR	double	Source right ascension in degrees corrected for field offset (optional column) [deg]
DEC_CORR	double	Source declination in degrees corrected for field offset (optional column) [deg]

Frame tags

- **SOURCE_LIST**:
  muse_exp_align: List of parameters of the detected point sources.

A.4 Other Data files

A.4.1 OUTPUT_WCS

Description

Normally, the MUSE pipeline automatically adapts the output cube dimensions and sky location depending on the data. This type of file can be used to override this automatism. The first valid FITS extension with a 2D or 3D FITS WCS (so with either NAXIS or WCSAXES set to 2 or 3) is used set override parameters for the output cube.

There are, however, several restrictions:

- The axes have to be in the order RA (1), DEC (2), wavelength (3)

- Only support gnomonic projection spatially (TAN), and linear or log air or vacuum wavelength sampling are supported (AWAV, AWAV-LOG, WAVE, or WAVE-LOG).

- A tilted 3rd axis is rejected.

- Only the primary WCS description is evaluated.

- The WCS transformation matrix has to be in CDi_j form (PCi_j is not accepted).

- Floating point WCS parameters without dots in the FITS are not recognized.

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

OUTPUT_WCS can also be used to set cube parameters that cannot be set through recipe parameters. E.g. logarithmic (standard FITS natural log) and vacuum output wavelengths can be set by adapting CRVAL3 according to the rules above.

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:	$4 \times$ Intel(R) Xeon(R) CPU E5-4620 0 @ 2.20GHz (8 cores each)
CPU Cores:	64 (32 physical, 32 logical)
System Memory:	128 GB
Storage System:	5 TB, transfer rate $\approx 80\,\mathrm{MB/s}$

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_lsf	45	24	900	75
muse_twilight	8	24	500	45
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 subsystem, 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`[21].

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 architecture, thread configuration, cache sizes, and a lot more.

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

```
1> likwid-topology
```

---

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

MUSE Pipeline User Manual

Doc. Number:	ESO-264503
Doc. Version:	1
Released on:	2022-03-10
Page:	157 of 159

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 Calibrations for Commissioning and Science Verification Data

When processing observations which were taken during one of the MUSE commissioning runs or the Science Verification runs the geometric calibration and the astrometric solution (`geometry_table_wfm.fits` and `astrometry_wcs_wfm.fits` which are included in the MUSE pipeline distribution are not applicable. Instead the calibrations prepared for these runs should be used. These legacy calibration files are available from the MUSE pipeline download page as a tar-archive. It includes the pairs of geometric calibration and astrometric solution suitable for data taken during the three Commissioning runs (COMM1, COMM2a, and COMM2b) and the two Science Verification runs (SV1, and SV2). The following table summarizes which calibrations should be used for which observations. The file name extension `.fits` has been omitted in the table.

The distribution package providing these files is available in the MUSE Additional Dataset section of the ESO pipeline download page http://www.eso.org/sci/software/pipelines as *Legacy Static Calibrations*.

Run	Expiration Date	Geometry Table	Astrometric Solution
COMM1	2014-02-22	`geometry_table_wfm_comm1`	`astrometry_wcs_wfm_comm1`
COMM2a + SV1	2014-07-15	`geometry_table_wfm_comm2a`	`astrometry_wcs_wfm_comm2`
COMM2b + SV2	2014-12-01	`geometry_table_wfm_comm2b`	`astrometry_wcs_wfm_comm2b`

MUSE Pipeline User Manual

Doc. Number:	ESO-264503
Doc. Version:	1
Released on:	2022-03-10
Page:	159 of 159

E   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
Likwid Lightweight Performance Tools	https://github.com/RRZE-HPC/likwid
FITS image compression utilities	https://heasarc.gsfc.nasa.gov/docs/software/fitsio/fpack
Zurich Atmosphere Purge Tool	http://muse-vlt.eu/science/tools