EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

EsoReflex MUSE Tutorial

VLT-MAN-ESO-19540-6195

Issue 10.0

Date 2017-03-01

	Name	Date	Signature
Prepared:	L. Coccato, R. Palsa	2017-03-01	
Approved:	W. Freudling		
Released:	M. Sterzik		

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	05-12-2014	All	First official release
2.0	01-02-2015	1-3, 6	Improved text and more detailed explanations on the use of the `muse_exp_combine.xml` workflow.
3.0	01-04-2015	All	Inclusion of exposure alignment in `muse.xml`; Replacement of , `muse_exp_combine.xml`, now dedicated only to alignment and combination of pre-reduced exposures (no raw or master calibration frames needed). Installation instructions compatible with the new `install_esoreflex` installation script.
4.0	15-04-2015	All	Change labels from 1.0.2 to 1.0.3. Updated instructions for installation (Linux and Mac).
5.0	28-04-2015	All	Change software version from 1.0.3 to 1.0.4.
6.0	01-08-2015	All	In sync with MUSE pipeline version 1.0.5 and `Reflex` 2.8.
7.0	01-10-2015	5-7	In sync with MUSE pipeline version 1.2.
		All	Few typos corrected.
8.0	01-10-2015	5-7	In sync with MUSE pipeline version 1.4.
9.0	18-04-2016	2, 5, 9-10	In sync with MUSE pipeline version 1.6. Updated instructions on how to use static response curve and provide user-defined sky masks.
10	01-03-2017	1, 6, 8	In line with pipeline version 2.0 and with the new layout: (creation of sky residual cubes and telluric correction strategy)

This page was intentionally left blank

Contents

1 Introduction And Scope

`Reflex` is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers. The basic philosophy and concepts of Reflex have been discussed by Freudling et al. (2013A&A...559A..96F). Please reference this article if you use Reflex in a scientific publication.

`Reflex` and the data reduction workflows have been developed at ESO and they are fully supported. If you have any issue, please contact *usd-help@eso.org* for further support.

This document is a tutorial designed to enable the user to employ the MUSE workflow to reduce his/her data in a user-friendly way, concentrating on high-level issues such as data reduction quality.

A workflow accepts science and calibration data, as delivered to PIs in the form of PI-Packs (until October 2011) or downloaded from the archive using the CalSelector tool[1] and organises them into datasets. Each dataset contains one or more science object observation (each observation possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction. The data organisation process is fully automatic, which is a major time-saving feature provided by the software. The datasets selected by the user for reduction are fed to the workflow which executes the relevant pipeline recipes (or stages) in the correct order. Full control of the various recipe parameters is available within the workflow, and the workflow deals automatically with optional recipe inputs via built-in conditional branches. Additionally, the workflow stores the reduced final data products in a logically organised directory structure employing user-configurable file names.

The MUSE `Reflex` workflow is designed to process all the single target scientific exposures independently. It therefore produces reconstructed datacube, images, and reduced pixel table for all of them. It is also to design to combine together exposures of the same object and the same instrument set-up, even from different Observing Blocks. The exposures will be automatically aligned before combination, using reference bright objects in the field of view.

The MUSE `Reflex` workflow handles both cases where sky exposures are present or not in the same OB of the target. In the latter case, the sky is evaluated in regions in the field of view where the target contribution is negligible. The most relevant parameters for the sky subtraction strategy can be specified directly in the `Reflex` canvas.

The current MUSE `Reflex` distribution contains also two additional workflows: `muse_exp_combine.xml`, which is dedicated only to the alignment and combination of already-processed reduced pixel tables (it does not process raw frames), and `muse_zap`, which is dedicated to the additional removal of sky residual lines.

In this document, we assume the user is already familiar with the recipes of the MUSE pipeline and their parameters. For more information, we refer the reader to the MUSE pipeline manual available at: *http://www.eso.org/sci/software/pipelines/*.

[1] *http://www.eso.org/sci/archive/calselectorInfo.html*

2 Software Installation

Reflex and the workflows can be installed in different ways: via package repositories, via `install_esoreflex` script or manually installing the software tar files.

The recommended way is to use the package repositories if you operating system is supported. The `macports` repositories support OS X, while the `rpm/yum` repositories support Fedora 20/21/22/23. For any other operating system it is recommended to use the `install_esoreflex` script.

2.1 Installing Reflex workflows via `macports`

This method is supported for OS X operating system. It is assumed that macports (*http://www.macports.org*) and java are installed. If you have any problem with this installation method, please read the full documentation at
http://www.eso.org/sci/software/pipelines/installation/macports.html.

For a quick installation, the following steps will install the ESO pipeline macports repository, the MUSE pipeline, including the Reflex workflow support and Reflex itself:

- Set up the repository:

```
# curl ftp://ftp.eso.org/pub/dfs/pipelines/repositories/macports/setup/Portfile -o Portfile
# sudo port install
# sudo port sync
```

- Install the MUSE pipeline:
  `# sudo port install esopipe-muse-all`

2.2 Installing Reflex workflows via `rpm/yum`

This method is supported for Fedora 20/21/22/23 operating systems. If you have any problem with this installation method, please read the full documentation at
http://www.eso.org/sci/software/pipelines/installation/rpm.html.

For a quick installation, the following steps will install the ESO pipeline rpm repository, the MUSE pipeline, including the Reflex workflow support and Reflex itself:

- Set up the repository for Fedora 20/21:

```
# sudo yum install yum-utils
# sudo yum-config-manager \
  --add-repo=ftp://ftp.eso.org/pub/dfs/pipelines/repositories/fedora/esorepo.repo
```

- Set up the repository for Fedora 22/23:

```
# sudo dnf install dnf-plugins-core
# sudo dnf config-manager \
  --add-repo=ftp://ftp.eso.org/pub/dfs/pipelines/repositories/fedora/esorepo.repo
```

- Install the MUSE pipeline (Fedora 20/21):
  `# sudo yum install esopipe-muse-all`
- Install the MUSE pipeline (Fedora 22/23):
  `# sudo dnf install esopipe-muse-all`

2.3 Installing Reflex workflows via `install_esoreflex`

The software pre-requisites for `Reflex 2.8.2` may be found at:
http://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.8.2` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

   ```
   wget ftp://ftp.eso.org/pub/dfs/reflex/install_esoreflex
   ```

2. Make the installation script executable:

   ```
   chmod u+x install_esoreflex
   ```

3. Execute the installation script:

   ```
   ./install_esoreflex
   ```

   and the script will ask you to specify three directories: the download directory `<download_dir>`, the software installation directory `<install_dir>`, and the directory to be used to store the demo data `<data_dir>`. If you do not specify these directories, then the installation script will create them in the current directory with default names.

4. You will be asked whether you want to use your Internet connection. Unless you want to reuse already downloaded packages (only advanced users), use the default Yes.

5. You will be given a choice of pipelines (with the corresponding workflows) to install. Please specify the numbers for the pipelines you require, separated by a space, or type "A" for all pipelines.

6. For the pipelines to be installed you will be prompted for the demo data sets to be installed. Type "A" for all demo datasets. Take into account that if you are installing in a directory that already contains data, it won't be removed.

7. The script will also detect whether previous versions of the workflows or Reflex were installed and in this case you have the option to update links or remove obsolete cache directories. It is advised to use the defaults.

8. To start `Reflex`, issue the command:

```
<install_dir>/bin/esoreflex
```

It may also be desirable to set up an alias command for starting the `Reflex` software, using the shell command `alias`. Alternatively, the `PATH` variable can be updated to contain the `<install_dir>/bin` directory.

3 System requirements

3.1 Hardware

The processing of MUSE data is very demanding in terms of computing resources. In particular, it requires a machine with sufficient memory installed, and it is available only for 64-bit machines. The recommended platform is a powerful workstation with a recent 64-bit Linux system.

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

- 64 GB of memory
- 24 CPU cores (physical cores)[2]
- 4 TB of free disk space
- GCC 4.8.2 (or newer)

Scientific programs usually foreseen the creation of a datacube by merging multiple exposures taken at the same position. On average, the memory consumption grows linearly with the number of observations.

In the case of creation of mosaic, the size of the data cube may become really huge, and the required memory grows accordingly.

By default, the workflow is set to use all the available cores (e.g. 24, in the configuration suggested above) and processes the data of the 24 MUSE IFUs *in parallel*. The serial or parallel execution of each recipe is set by the `nifu` recipe parameter[3]: a value of -1 will process the IFUs in parallel (fast, but memory demanding), a value of 0 will process the IFUs in series (24 times slower, but less memory demanding). A value of $1 \leq N \leq 24$ will process only the selected $N-$th IFU.

The execution of the recipes in parallel and the use of all the available cores can led to memory issues even for a 64 GB machine, if many calibration files are to be combined together. For example, the combination of 45 flats or 45 arcs to generate the `LSF_PROFILE` requires more than 64 GB. A solution could be to instruct the workflow to use only 12 cores. In this case, the workflow computes 2 groups of 12 IFUs in series, and the 12 IFU in each group are processed in parallel; the execution times doubles, but memory demands are halved (at least for those recipes that accept the `nifu` parameter).

This can be done by setting the following environmental variable before the execution of `Reflex`:

```
export OMP_NUM_THREADS=12
```

For more information, please refer to the MUSE pipeline manual available at
http://www.eso.org/sci/software/pipelines/.

[2]Using 24 CPUs is on average from 10% to 30% faster than using 12 CPUs (although the number of CPUs is doubled). Please evaluate the costs/benefits of a 24 CPU system over a 12 CPU system.

[3]This parameter is available only in the **muse_bias**, **muse_flat**, **muse_wave**, and **muse_lsf**, and **muse_scibasic** recipes. Other recipes, which are designed to combine all the available IFUs cannot be run in series.

3.2 JVM Memory set-up

The MUSE workflow need a sufficient amount of memory allocated to `Reflex`. The best way to set the memory allocation of `Reflex` is to run the `reflex_set_memory` script that is distributed with `Reflex` *before* starting `Reflex`. The recommended setting for MUSE is to leave the "Minimum amount of memory" unchanged, and set the "Maximum amount of memory" to 2000. Alternatively, the memory setting can be done after starting `Reflex` by clicking on "Tools – JVM Memory Settings" in the menu bar. `Reflex` needs to be restarted for this change to be applied.

3.3 Execution on machines with less than 64 GB of memory

The MUSE pipeline and the `Reflex` workflow can be still executed in less powerful machines, such as laptops with 8GB of RAM, provided that the user restricts the wavelength range to short interval (e.g. 100 Å). This set-up, although still demanding in terms of computational time, allows the user to test the data reduction strategy before having access to a more powerful machine and reduce the data on the full wavelength range. For example, it can be used to create sky masks, to find the best method and parameters for the sky subtraction in critical wavelength ranges, to calculate the coordinate offsets between different exposures, and much more. More information are in Section 8.5.

4 Demo Data

A demo dataset is distributed together with the MUSE `Reflex` workflow. It consists of three target exposures, one off-set sky exposure, on-sky calibration frames (sky flats, standard star), and instrument calibration frames (biases, flats, arcs). In addition, the set of static calibrations included in the pipeline distribution is needed for the reduction of the demo dataset.

- **Target exposures.** Frame Tags: `PRO.CATG=OBJECT`; `DPR.TYPE=OBJECT`
  - `MUSE_WFM-NOAO_OBS173_0069.fits`
  - `MUSE_WFM-NOAO_OBS173_0071.fits`
  - `MUSE_WFM-NOAO_OBS039_0037.fits`
- **Offset-sky exposures.** Frame Tags: `PRO.CATG=SKY`; `DPR.TYPE=SKY`
  - `MUSE_WFM-NOAO_OBS173_0070.fits`
- **On-Sky calibration exposures.**
  - **Sky flats.** Frame Tags: `PRO.CATG=SKYFLAT`; `DPR.TYPE=FLAT,SKY`
    - `MUSE_WFM_SKYFLAT172_0001.fits`
    - `MUSE_WFM_SKYFLAT172_0002.fits`
    - `MUSE_WFM_SKYFLAT172_0003.fits`
    - `MUSE_WFM_SKYFLAT172_0004.fits`
    - `MUSE_WFM_SKYFLAT172_0005.fits`
  - **Standard star.** Frame Tags: `PRO.CATG=STD`; `DPR.TYPE=STD`
    - `MUSE_WFM_STD172_0002.fits`
- **Instrument calibration exposures.**
  - **Biases.** Frame Tags:`PRO.CATG=BIAS`; `DPR.TYPE=BIAS`
    - `MUSE_CAL_BIAS173_0004.fits`
    - `MUSE_CAL_BIAS173_0005.fits`
    - `MUSE_CAL_BIAS173_0006.fits`
    - `MUSE_CAL_BIAS173_0007.fits`
    - `MUSE_CAL_BIAS173_0008.fits`
  - **Flat fields.** Frame Tags:`PRO.CATG:FLAT`; `DPR.CATG:FLAT,SKY`
    - `MUSE_WFM_FLAT172_0049.fits`
    - `MUSE_WFM_FLAT172_0050.fits`
    - `MUSE_WFM_FLAT172_0051.fits`
    - `MUSE_WFM_FLAT172_0052.fits`
    - `MUSE_WFM_FLAT172_0053.fits`
  - **Wavelength calibrations.** Frame Tags:`PRO.CATG=ARC`; `DPR.TYPE=WAVE`

    * MUSE_WFM_WAVE173_0001.fits
    * MUSE_WFM_WAVE173_0002.fits
    * MUSE_WFM_WAVE173_0003.fits
    * MUSE_WFM_WAVE173_0004.fits
    * MUSE_WFM_WAVE173_0005.fits
    * MUSE_WFM_WAVE173_0006.fits
    * MUSE_WFM_WAVE173_0007.fits
    * MUSE_WFM_WAVE173_0008.fits
    * MUSE_WFM_WAVE173_0009.fits

- **Static calibrations.**
  - Files included in the pipeline distribution:
    * astrometry_reference.fits Frame Tags: PRO.CATG=ASTROMETRY_REFERENCE
    * astrometry_wcs_wfm.fits Frame Tags: PRO.CATG=ASTROMETRY_WCS
    * badpix_table.fits Frame Tags: PRO.CATG=BADPIX_TABLE
    * extinct_table.fits Frame Tags: PRO.CATG=EXTINCT_TABLE
    * filter_list.fits Frame Tags: PRO.CATG=FILTER_LIST
    * geometry_table_wfm.fits Frame Tags: PRO.CATG=GEOMETRY_TABLE
    * line_catalog.fits Frame Tags: PRO.CATG=LINE_CATALOG
    * sky_lines.fits Frame Tags: PRO.CATG=SKY_LINES
    * std_flux_table.fits Frame Tags: PRO.CATG=STD_FLUX_TABLE
    * std_response_wfm-e.fits Frame Tags: PRO.CATG=STD_RESPONSE
    * std_response_wfm-n.fits Frame Tags: PRO.CATG=STD_RESPONSE
    * vignetting_mask.fits Frame Tags: PRO.CATG=VIGNETTING_MASK
    * lsf_profile_slow_wfm-n.fits Frame Tags:PRO.CATG=LSF_PROFILE
    * lsf_profile_slow_wfm-e.fits Frame Tags:PRO.CATG=LSF_PROFILE
  - Files included in the workflow demo data-set:
    * TRACE_TABLE-06_NomMode_Temp8p49.fits Frame Tags:PRO.CATG=TRACE_TABLE
    * TRACE_TABLE-06_ExtMode_Temp8p53.fits Frame Tags:PRO.CATG=TRACE_TABLE

5 Quick start: reducing the demo data

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the MUSE demo data set supplied with the `Reflex 2.8.2` release. By following these steps, the user should have enough information to attempt a reduction of his/her own data without any further reading:

1. Start the `Reflex` application by typing

   ```
   <install_dir>/bin/esoreflex &
   ```

   at the terminal command line.

   The empty `Reflex` canvas will appear (Figure 5.1). We refer to Section 9 for a description of the main elements of the canvas. Note: for other datasets, please consider to reduce the number of cores to be used by setting the environmental variable `OMP_NUM_THREADS=12` before starting `Reflex` (see Section 3.1).

2. If you have not set the Memory allocation before (see section 3.2), then set the JVM Memory settings, and restart `Reflex`.

3. Now open the MUSE workflow by clicking on `File -> Open File`, selecting first `MUSE-1.6` and then the file `muse.xml` in the file browser. You will be presented with the workflow canvas shown in Figure 5.2. Note that the workflow will appear as a canvas in a new window.

   Note: there is a second workflow in the directory, `muse_exp_combine.xml`, which will be described in Section 7.2.

4. To aid in the visual tracking of the reduction cascade, it is advisable to use component (or actor) highlighting. Click on `Tools -> Animate at Runtime`, enter the number of milliseconds representing the animation interval (100 ms is recommended), and click `OK`.

5. Under "Setup Directories" in the workflow canvas there are seven parameters that specify important directories (green dots). Setting the value of `ROOT_DATA_DIR` is the only necessary modification if you want to process data other than the demo data[4], since the value of this parameter specifies the working directory within which the other directories are organised. Double-click on the parameter `ROOT_DATA_DIR` and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the `Browse` button to select the directory from a file browser. When you have finished, click `OK` to save your changes. Further information on the set-up directories are given in Section 10.1.

   Note: for data acquired prior to August 2014, the tracing performed on the flat fields can fail due low detector temperature effects. This is automatically taken into account in the workflow by using static `TRACE_TABLE-06` files, which will replace those created by the pipeline in the case the failure is detected. The `TRACE_TABLE-06` files can be retrieved from:
   http://ftp://ftp.eso.org/pub/dfs/pipelines/muse/muse-calib-tracetables-ifu06.tar.gz and can be stored in the static calibration directory. The Demo dataset already includes these files.

[4] If you used the install script `install_esoreflex`, then the value of the parameter `ROOT_DATA_DIR` will already be set correctly to the directory where the demo data was downloaded.

6. Specify the desired data reduction strategy by setting the relevant parameters in the workflow canvas, under the "Data reduction strategy parameter". The description of these parameters will be done in Section 6.3.

7. Press  to start the workflow. The following steps will be executed:

   - If `EraseDirs=true` was set, the workflow will erase the content of the log, bookkeeping, and temporary directories.
   - The workflow will highlight the `Data Organiser` actor which recursively scans the raw data directory (specified by the parameter `RAW_DATA_DIR` under "Setup Directories" in the workflow canvas) and constructs the DataSets. Note that the raw and static calibration data must be present either in `RAW_DATA_DIR` or in `CALIB_DATA_DIR`, otherwise DataSets may be incomplete and cannot be processed. However, if the same reference file was downloaded twice in different places this creates a problem as `Reflex` cannot decide which one to use.

     Datasets are created on the basis of header information and a set of `OCA`[5].
   - The `Data Set Chooser` actor will be highlighted next and will display a "Select Datasets" window (see Figure 5.3) that lists the DataSets along with the values of a selection of useful header keywords[6]. The first column consists of a set of tick boxes which allow the user to select the DataSets to be processed. By default all complete DataSets which have not yet been reduced will be selected. Incomplete datasets are greyed out in the window. If you put the mouse cursor on the top of an incomlete dataset, you will get information on the missing file categories.

     There are two types of datasets. The first type contains only one science target exposure (and calibrations), the second type contains multiple science exposures belonging to the same target name and the same instrument configuration (and calibrations). This latter type of datasets have the suffix `_combined_cubes` appended to its name. The target exposures in each of these latter datasets are meant to be combined together into one unique final datacube.

8. Click the `Continue` button and watch the progress of the workflow by following the red highlighting of the actors. A window will show which DataSet is currently being processed.

   Tip: it is recommended to start processing the single exposures first, and then re-run the workflow to process the `_combined_cubes` datasets afterwards. The main reason is that the data process can take up to several hours. If starting with the individual exposures, the user can inspect the products as soon as each exposure has been processed (if the ProductExplorerMode is set to "Enabled" in the main `Reflex` canvas), without waiting for all of them to be processed. Then, when processing the `_combined_cubes`, the previous reduction of the individual exposures will be re-used in virtue of the lazy mode (if the recipe parameters and input files have not changed in the meantime).

   Warning: When reducing individual datasets, the reduced pixeltables are copied in the final directory. This require more disks space.

[5] `OCA` stands for Organization, Classification, Association and refers to rules, which allow to classify the raw data according to the contents of the header keywords, organize them in appropriate groups for processing, and associate the required calibration data for processing. They can be found in the directory `<install_dir> /share/esopipes/muse-2.0/reflex/` , carrying the extension `.oca`

[6] The keywords listed can be changed by right-clicking on the `DataOrganiser` Actor, selecting `Configure Actor`, and then changing the list of keywords in the second line of the pop-up window.

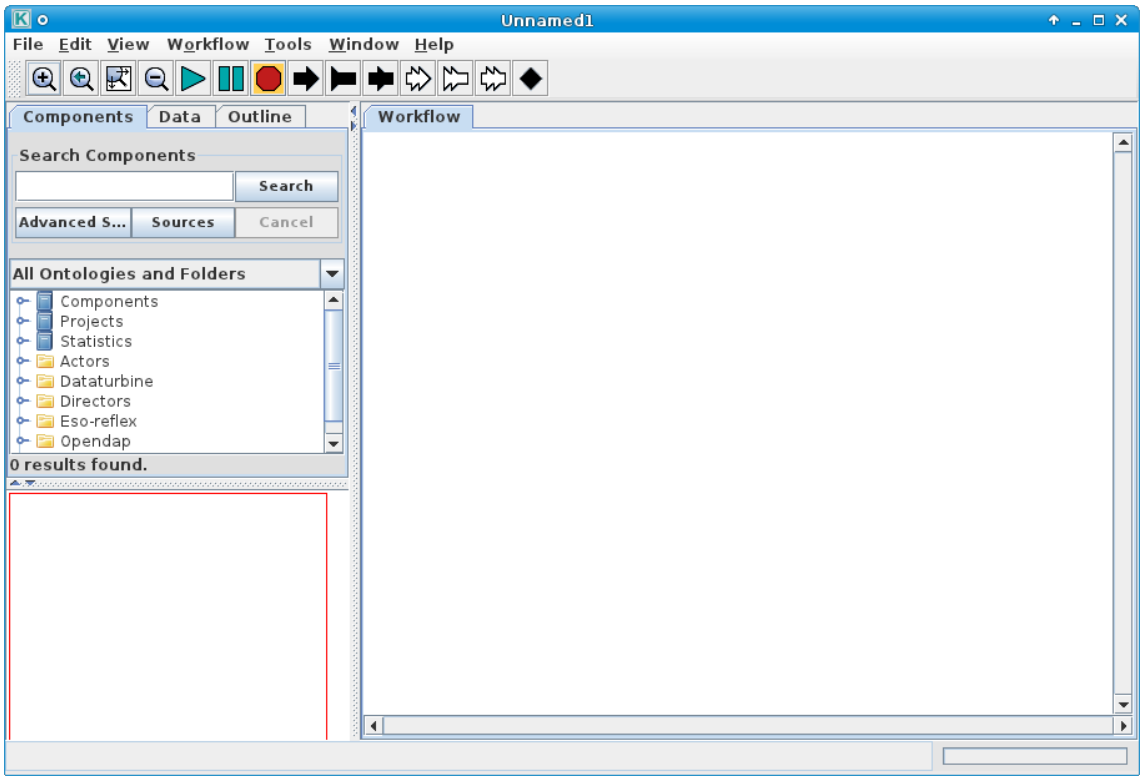


Figure 5.1: The empty `Reflex` canvas.

9. The workflow processes the selected datasets separately. Only when the reduction of one dataset is completed, the reduction of the next dataset starts.

10. The workflow will continue with the remaining DataSets following the same steps described above.

11. After the workflow has finished, all the products from all the DataSets can be found in a directory under `END_PRODUCTS_DIR` with the named with the workflow start timestamp. Further subdirectories will be found with the name of each DataSet.

12. When the reduction of the last DataSet is finishes, a pop-up window called *Product Explorer* will appear showing the datasets which have been so far reduced together with the list of final products. This actor allows the user to inspect the final data products, as well as to search and inspect the input data used to create any of the products of the workflow. Figure 5.4 shows the Product Explorer window.

    Note: if the global parameter ProductExplorerMode is set to "Enabled", the `Reflex` Product Explorer pups up at the end of each reduced data set. The default value is "Triggered" , which enables the *Product Explorer* at the end of the last dataset.

Well done! You have successfully completed the quick start section and you should be able to use this knowledge to reduce your own data. However, there are many interesting features of `Reflex` and the MUSE workflow that merit a look at the rest of this tutorial.

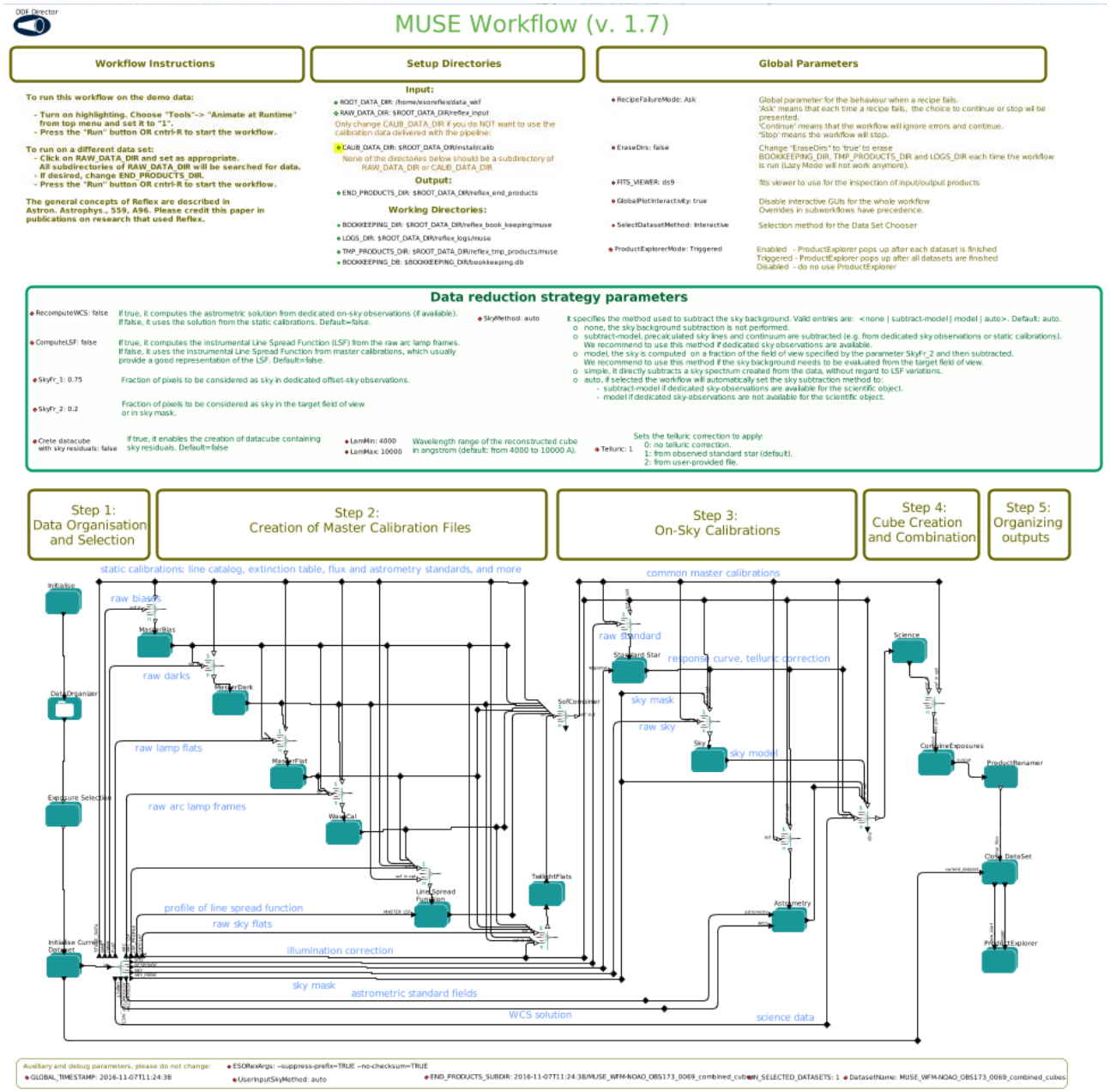


Figure 5.2: The MUSE `Reflex muse.wkf` workflow.

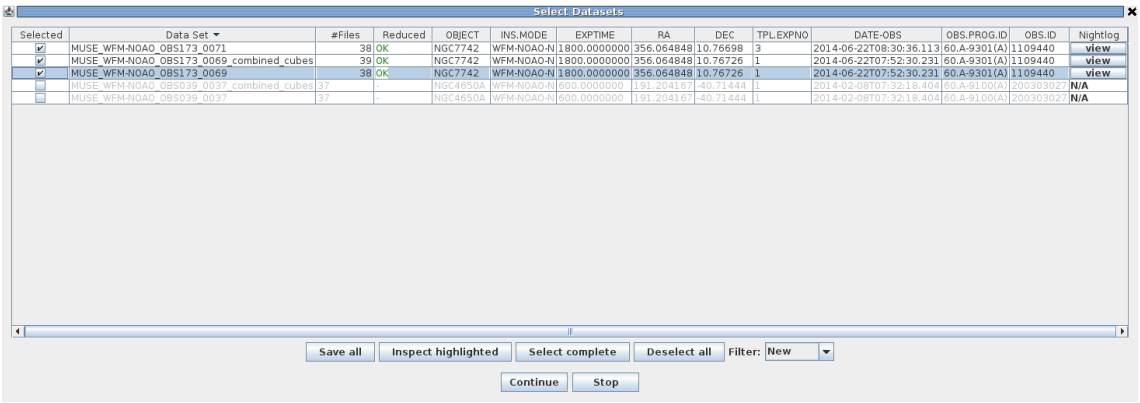


Figure 5.3: The Select Dataset window.

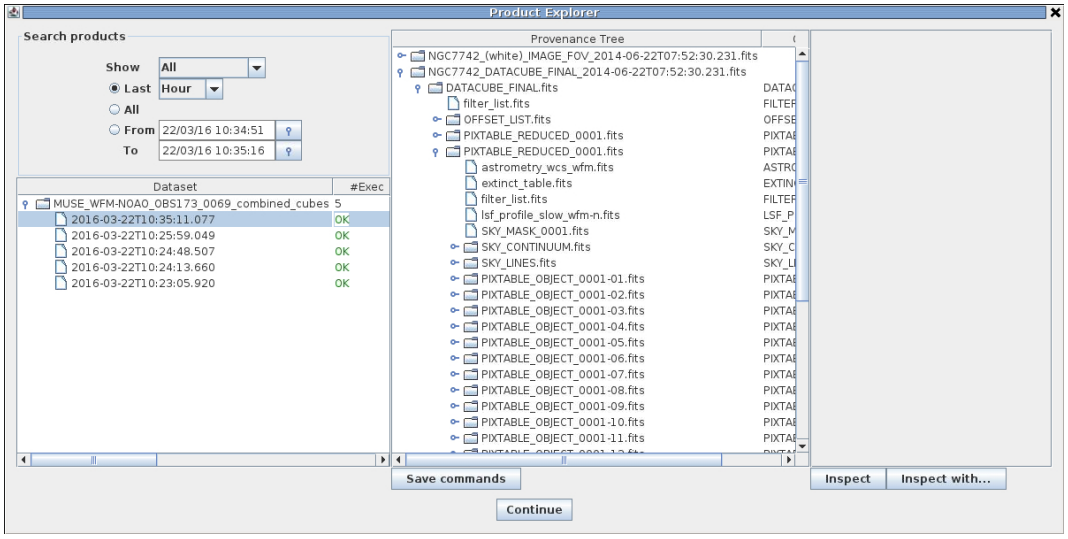


Figure 5.4: The Product Explorer window.

6 Reducing MUSE data

In order to reduce a desired dataset, please set the `ROOT_DATA_DIR` and `RAW_DATA_DIR` to the desired paths.

6.1 The data reduction cascade and the workflow composite actors

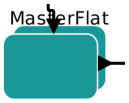
The MUSE workflow is designed to execute a well defined data reduction cascade. It triggers a number of "composite actors" that are associated to specific pipeline recipes. The exact execution sequence of these actors and recipes depends on the content of the dataset itself and on the data reduction strategy set up by the user. In particular, the user can specify whether to compute a new parametrization of the line spread function, a new astrometric calibration or to use the ones provided with the downloaded dataset. Also the strategy for sky reduction can be decided, whether to use dedicated sky observations (if available) or compute the background sky contribution on empty regions in the scientific target field of view. All these strategies can be configured by setting the appropriate strategy parameters in the main workflow canvas (Section 6.3).

The workflow triggers the following composite actors:

Calibration Recipes:

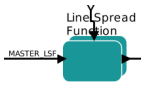
- 

`MasterBias`: it executes the recipe **muse_bias** . It processes the raw bias frames and creates a master bias.
- 

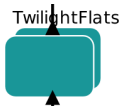
`MasterDark`: it executes the recipe **muse_dark**. If raw dark frames are present, the actor processes them and creates a master dark. It requires the products of `MasterBias` as inputs. *Important:*: the use of dark frames is not recommended for the scientific reduction; dark frames are taken on a monthly base (therefore they do not represent in detail the dark current at the time of the observations) and they add noise to the final products. Currently, the recipe is disabled by default. To enable it, open the MasterDark actor, double click on the master dark recipe to configure it, and set the recipe mode from "Disabled" to "Run".
- 

`MasterFlat`: it executes the recipe **muse_flat**. It processes the raw flat-fields exposures, producing a master flat and a trace table. It requires the products of `MasterBias` and `MasterDark` (if executed) as inputs.
- 

`WaveCal`: it executes the recipe **muse_wavecal**. It processes the raw arc frames, producing a table with the wavelength solution. It requires the products of `MasterBias`, `MasterDark` (if executed), and `MasterFlat` as inputs.

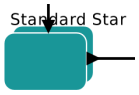
- 

`Line Spread Function`: it executes the recipe **muse_lsf**. If available, and if specified by the **ComputeLSF** parameter (Section 6.3), the recipe processes dedicated raw arc frames and produces a table with the parametrization of the line spread function. It requires the products of `MasterBias`, `MasterDark` (if executed), `MasterFlat`, and `WaveCal` as inputs.

- 

`TwilightFlats`: it executes the recipe **muse_twilight**. If available, the recipe processes twilight sky flats and produces illumination corrections. It requires the products of `MasterBias`, `MasterDark` (if executed), `MasterFlat`, and `WaveCal` as inputs.

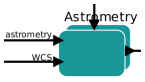
Science Recipes:

- 

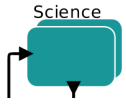
`Standard Star`: it executes the recipes **muse_scibasic** and **muse_standard**. It processes the frames of the standard star and returns a response curve and a telluric correction. If standard stars observations are not present in the dataset, it uses the response curve from the static calibration. It requires the products of the calibration recipes as input. It is possible to use user-supplied version of the response curve and telluric correction rather than those produced by the Standard Star actor (see Section 8.4).

- 

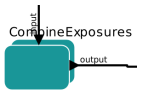
`Sky`: it executes the recipes **muse_scibasic** and **muse_create_sky**. If dedicated sky observations are present in the dataset, it creates a list of sky emission lines and a sky continuum to be used in the sky subtraction of the science observations. It requires the products of the calibration recipes and of `Standard Star` as input. The recipe has an automatic procedure to determine the sky regions in the frame. It is hower possible to provide an optimized sky mask for this purpose (see Section 8.2).

- 

`Astrometry`: it executes the recipe **muse_scibasic** and **muse_astrometry**. If available, and if specified by the **ComputeWCS** parameter (Section 6.3), the recipe processes the raw astrometric calibrations frames and produces the astrometric solution. It requires the products of the calibration recipes and of `Standard Star` as input.

- 

`Science`: it executes the recipe **muse_scibasic** and **muse_scipost**. It processes the raw science frames producing fully reduced datacubes (`DATACUBE_FINAL`), pixel table (`PIXTABLE_REDUCED`), and reconstructed images (`IMAGE_FOV`). The strategy for sky subtraction depends on the user set up of the **SkyMethod** strategic parameter (Section 6.3). It requires the the products of the calibration recipes, of `Standard Star`, `Astrometry` (if executed), and `Sky` (according to the parameter set-up) as input.

- 

`Combine Exposures`: if the selected dataset contains multiple exposures to be aligned

and combined together, the actor executes the recipes **muse_exp_align** and **muse_exp_combine**. It requires the products of the `Science` actor.

For more information about the inputs and outputs of individual recipes and their configuration parameters, please consult the MUSE pipeline manual.

6.2 Workflow products

All the products of the individual recipes are saved into the temporary products directory (`TMP_PRODUCTS_DIR` that can be set up in the main workflow canvas). The most important final products will be also copied into the `END_PRODUCTS_DIR` final product directory; the exact category of final products that is copied there depends on the nature of the dataset.

If the processed dataset contains multiple exposures, the following final products obtained from the combination of the individual exposures are saved in the final product directory:

- `DATACUBE_FINAL`: Output datacube. Its a 3 extensions fits file. The first extension contains the primary header. The second extension contains the fully reduced datacube, obtained by combining the reconstructed cubes of the IFUs in each exposure that belong to the same object. The datacube is a three-dimensional array (x, y, λ), where the first two dimensions represent the spatial coordinates on the sky (RA and DEC, respectively). The third dimension is wavelength (in units of Å). Therefore for a given (x, y), the datacube shows the spectrum obtained at those RA and DEC coordinates on the sky. Units of the second extension are 10^{-20} ergs cm^{-2} Å^{-1} s^{-1}. The third extension contains the error cube associated to the second extension. Units of the third extension are $(10^{-20}\mathrm{ergs\ cm^{-2}\ Å^{-1}\ s^{-1}})^2$.
- `IMAGE_FOV`: Field-of-view images corresponding to the –filter parameter (default: `-filter=white`). It is obtained by integrating the datacube along the wavelength direction using the filter transmission curve specified by the –filter recipe parameter.

The products of the individual exposures are saved in the temporary directory `TMP_ PRODUCTS_DIR`.

If the dataset contains only single exposures, the following final products are saved in the `END_PRODUCTS_DIR` final product directory:

- `DATACUBE_FINAL`: Output datacube, as described above.
- `IMAGE_FOV`: Field-of-view images, as described above.
- `PIXTABLE_REDUCED`: Fully reduced pixel tables for each exposure. They contain the information of coordinates on the sky, flux, wavelength, data quality, error for each individual pixel in the detectors. The datacubes are obtained by resampling the reduced pixel tables onto a regular 3D grid.

The exact name of the final product depends on the header keywords of the input dataset, as specified in the configuration of the ProductRenamer.

At the end of the reduction of each dataset, a message will pop-up indicating the location of the final products.

Lazy mode. By default, the lazy mode in `Reflex` is enabled: each recipe can re-use the products of a previous execution, if parameters and the input files are unchanged. This means that the products of one recipe can be re-used for the reduction of another dataset that processes the same files with the same parameters. One important application is the following: let's suppose that we have already reduced the dataset with combination of multiple exposures. Then, the reduction of each single dataset used in the combination will be extremely fast, because the products of the previous reduction will be re-used.

6.3 Setting the data reduction strategy

All the recipe parameters can be changed by configuring the associated `RecipeExecuter` actor. This can be done by opening the various composite actors until the `RecipeExecuter` associated to the desired pipeline recipe is visible. To open a composite actor, click on it with the mouse right button, and select "Open Actor". To configure the `RecipeExecuter` click on it with the mouse right button, and select "Configure Actor". A list of all recipe parameters will be available for editing. Press "Commit" to apply the changes.

The main `Reflex` canvas offers to the user a quick selection of some key parameters and options, which are relevant to select the appropriate strategy for the data reduction. They are located in the section "Data reduction strategy parameters" (Figure 6.1).

- **Calibrations parameters.**
  - **RecomputeWCS.** Valid entries are *true* or *false*. Default = *false*. It sets the astrometric solution to be used in the workflow.

    If set to *false*, the astrometric solution are selected from the static calibration directory (recommended option).

    If set to *true*, the astrometric solution will be recomputed from on-sky observations of astrometric standard fields (if available in the dataset).

    The frame tags for the astrometric solution and on-sky raw calibrations are `ASTROMETRIC_WCS` and `ASTROMETRY`, respectively.

    It is advisable to set **RecomputeWCS** = *true* only if the on-sky astrometric exposures are observed in very good observing conditions.
  - **ComputeLSF.** Valid entries are *true* or *false*. Default = *false*. It sets which parametrization of the instrumental Line Spread Function (LSF) to use during data reduction. This parametrization is fundamental to construct reliable models of the sky emission lines. The frame tag for the LSF parametrization is `LSF_PROFILE`. A reliable parametrization requires 45 dedicated arc lines frames (`DPR.TYPE = WAVE,LSF`).

    If set to *false*, the workflow uses the parametrization associated to the dataset (either downloaded with the dataset, or in the static calibration directory). With this option, the **muse_lsf** recipe within the Line Spread Function actor is not triggered.

    If set to *true*, the LSF is parametrized by the **muse_lsf** recipe from the dedicated raw arc calibrations available in the dataset.

    Typically, the `LSF_PROFILE` calibration downloaded with the dataset contains a very good parametrization of the instrumental line spread function profile.
  - **Telluric**. It specifies which telluric correction strategy to adopt. Possible values are:

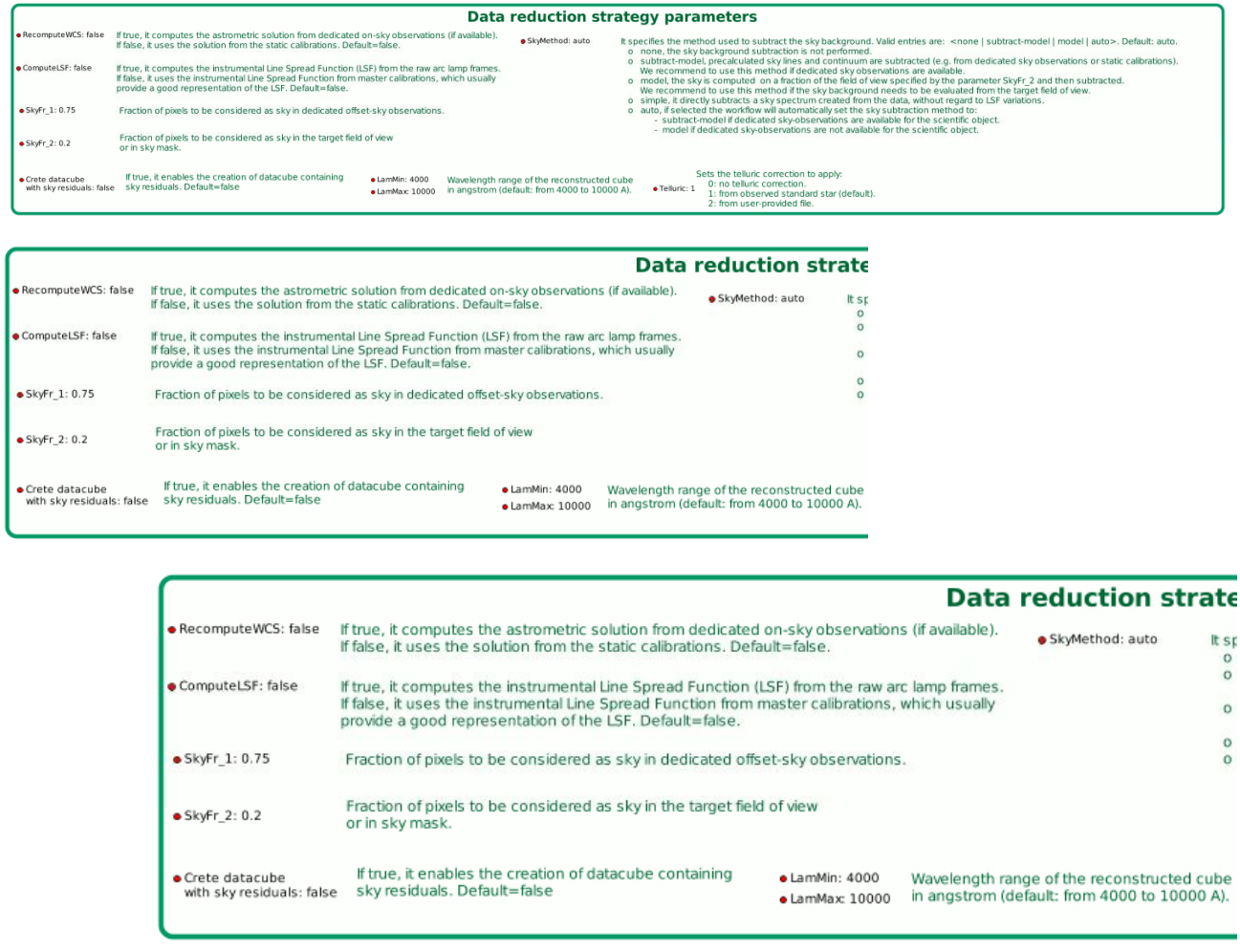


Figure 6.1: The Data reduction strategic parameters section in the `Reflex` canvas. Top panel contains the full view of the section. Central and Bottom panels contain the left and right zooms of the section, respectively.

  - 0 does not perform any telluric correction.
  - 1 (default) performs the telluric correction using the standard star observed at twilight.
  - 2 Uses an user-provided file (same structure of the STD_TELLURIC, as produced by the muse_standard recipe).

  Please consult Section Section 8.4 for more information .

- **Wavelength range parameters.** If you are interested in a restricted wavelength range it is possible to create the final datacube accordingly. The following parameters have an affect on the **muse_create_sky**, **muse_astrometry**, **muse_scipost**, and **muse_exp_combine** recipes.

  - **LamMin**. Sets the minimum wavelength (in Å) to consider when reconstructing the datacube. It corresponds to the recipe parameter `-lambdamin`. Default: 4000.
  - **LamMax.** Sets the maximum wavelength (in Å) to consider when reconstructing the datacube. It corresponds to the recipe parameter `-lambdamax`. Default: 10000.

  The recipe **muse_standard** is not affected by **LamMin** and **LamMax**, because the change of the corresponding `-lambdamin` and `-lambdamax` parameters can cause the recipe to fail. If you are using on a computer with limited memory capabilities (see Section for hardware specifications 3.1) the **muse_standard** will fail in reconstructing the datacube and the workflow will crash. To avoid that, you can remove the standard star observations from the dataset and use the master calibrations downloaded from the ESO archive via the CalSelector tool.

- **Strategy for sky subtraction.** The MUSE pipeline (and therefore the workflow) evaluates the sky to be subtracted in two ways. If dedicated sky observations are present, the sky can be evaluated using a specified fraction of spaxels in the reconstructed sky image. This fraction is specified by the **SkyFr_1** workflow parameter. Alternatively, the sky can be evaluated directly on the scientific frames, in regions where the contribution of the target is negligible. A specified fraction of spaxels in the reconstructed image (the faintest spaxels) will be selected to create the sky spectrum. This fraction is specified by the **SkyFr_2** workflow parameter.

  The following parameters are relevant for the sky subtraction. Each dataset might require different values.

  - **Create Datacube with sky residual**. If set to true, the workflow will create datacubes, reconstructed image, and pixeltables of the dedicated sky exposures (if present in the dataset). These product are saved in a directory nemed after the dataset they refer to, but with the suffix `-SkyResidualCubes` in the name, which is located in the reflex end product directory. It is advisable to set it to true if one intend to remove residual sky lines with the ZAP tool (see Section 8.3).
  - **SkyFr_1.** It corresponds to the recipe parameter `-fraction` in the **muse_ create_sky** recipe. This is relevant only if dedicated sky exposures are present in the dataset.
  - **SkyFr_2.** It corresponds to the recipe parameter `-skymodel_fraction` in the **muse_scipost** recipe. This is relevant if the sky has to be evaluated directly from the target exposure.
  - **SkyMethod.** Method for sky subtraction. Valid entries are: “auto”, “model”, “subtract-model”, “simple”, and “none”. Default: auto. These values define the `-skymethod` parameter in the **muse_scipost** recipe, except for the “auto” mode (which is not recognized by **muse_scipost**).
    - “none”, the sky background subtraction is not performed.

- "subtract-model", precalculated sky lines and continuum are subtracted (e.g. from dedicated sky observations or static calibrations). We recommend to use this method if dedicated sky observations are available.
- "model", the sky is computed on a fraction of the field of view specified by the parameter **SkyFr_2** and then subtracted. We recommend to use this method if the sky background needs to be evaluated from the target field of view.
- "simple", the sky to be subtracted is created directly from the data, without regard to LSF variations.
- "auto" If selected, the workflow will automatically set the **SkyMethod** variable to:
  - "subtract-model" if dedicated sky-observations are available for the scientific object in the dataset.
  - "model" if dedicated sky-observations are not available for the scientific object in the dataset.

If dedicated sky observations are present, good values could be **SkyFr_1** = 0.75, **SkyFr_2** = 0.2, and **SkyMethod** = subtract-model (the latter is automatically set if the "auto" mode is selected). If offset sky observations are not present, the sky will be evaluated from a specified fraction of pixels in the target field of view; good values could be **SkyFr_2** = 0.2 and **SkyMethod** = model (the latter is automatically set if the "auto" mode is selected); the parameter **SkyFr_1** has no effect in this case.

Warning: if **SkyMethod** = *model*, **SkyFr_2** cannot be 0, otherwise the recipe fails.

Warning: If dedicated sky observations are present, and if the SkyMethod is set to model, then the sky continuum is evaluated from the dedicated sky observations, and the sky emission lines are computed from the science target. This strategy reduces systematic due to to variations of the intensity of the emission lines between the sky and the target observations. It is recommended if there is a portion of the field of view in the target exposure where the continuum of the targets is small (even if not zero), and the emission lines from the target are negligible. Usually, this strategy gives better results than computing also the sky continuum in the target exposure because the time variation of the sky continuum are smaller than that of the emission lines.

However, If an user wants to compute also the sky continuum in the target exposure (i.e., not using at all the dedicated sky observations), then the suggested strategy is to remove the dedicated sky observations from the dataset. In the Select Dataset window (Figure 5.3), highlight the desired dataset, click on Inspect Dataset, and deselect the sky exposures.

7 Combination of multiple exposures

The `muse.xml` workflow processes individual exposures independently and, depending of the content of the dataset, it combines the exposures that belong to the same object and the same instrument configuration to generates a final combined datacube. The combination of individual exposures onto a final datacube is performed on datasets with the suffix `combined_cubes` in their names.

In some cases, the individual exposures might need to be aligned to each other to get a proper combination. This can be done using, for example, foreground stars as reference. After the alignment process is done, the individual exposures will be combined into the final cube.

The alignment and combination of exposures can be done in three ways:

- Automatic offsets computation and image combination with the `muse.xml` workflow. This is the recommended way; see Section 7.1 for further details of the executed steps.
- Automatic offsets computation and image combination with a dedicated MUSE `EsoReflex` workflow, named `muse_exp_combine.xml`. It processes only pre-reduced `PIXTABLE_REDUCED` and `IMAGE_FOV`, without the need of any raw or master calibration file. See Section 7.2 for this option.
- The automatic offsets computation and exposure combination can be run separately outside the workflow via the `esorex` command line. The MUSE recipes that perform these two tasks are **muse_exp_align** and **muse_exp_combine**, respectively. If needed, the combination of exposures can be done using offsets computed by other tools than the **muse_exp_align** recipe.

  This solution is advisable only when the automatic alignment procedure proposed above fails, or does not compute offsets that are accurate enough. This can happen, for example, if there are not enough bright sources in the field of view (or in the overlapping regions between adjacent pointings) or if the sources are too smooth and featureless to be used as reference.

  Please consult the MUSE pipeline manual http://www.eso.org/sci/software/pipelines/ for this option.

7.1 Automatic combination within the `muse.xml` workflow

The MUSE workflow first processes individual scientific exposures separately, and then it creates one set of products for each one of them (`DATACUBE_FINAL`, `IMAGE_FOV`, and `PIXTABLE_REDUCED`).

For the datasets with name ending in `combined_cubes`, the workflow groups together all the individual exposures and combines them into an unique final datacube.

The combination procedure includes the following steps, which are executed automatically by the workflow:

1. Execution of the **muse_exp_align** recipe for image alignment. This recipe performs the following tasks:
   - Identify sources in each `IMAGE_FOV` frame.
   - Compute the offsets of each frame with respect to a reference position by comparing the common sources of each overlapping exposures. If an exposure has no overlap, its offset is set automatically to zero (i.e., the coordinates in the header will be used for alignment).

- Produces a fits table with the list of the offsets in RA and DEC for each of the exposures to combine (`OFFSET_TABLE`).

2. Combination of the `PIXTABLE_REDUCED` into an unique datacube using the **muse_exp_combine** recipe and the offsets defined in the `OFFSET_TABLE`. A reconstructed image (`IMAGE_FOV`) will also be produced.
3. Copy the final results into the directory: `END_PRODUCTS_DIR/ <timestamp>` , where `<timestamp>` is the time stamp of the latest `EsoReflex` execution.
4. If set by the general parameter `ProductExplorerEnabled`, the product explorer will show up, showing the product association tree and allowing the inspection of each file in the data reduction chain.

To disable the alignment procedure (but still leaving the combination active), open the CombineExposure actor (right mouse button and select Open Actor), locate the muse_exp_align_1 recipe executer and configure it (right mouse button and select Configure Actor). Set the recipe mode from "Run" to "Disable".

7.2 Automatic combination within the `muse_exp_combine.xml` workflow

There might be the case where the user has only the `PIXTABLE_REDUCED` and `IMAGE_FOV` to combine, and no raw or master calibration frames. To this purpose, the `muse_exp_combine.xml` workflow can be used.

To use this workflow:

- Select the MUSE workflow by clicking on `File -> Open...` and then selecting the file `muse-2.0/muse_exp_combine.xml` in the file browser. A new window containing the workflow will appear (see Figure 7.1).
- Edit the `RAW_DIR` and specify the directory containing the frames to be combined (`PIXTABLE_REDUCED` and `IMAGE_FOV`). Files are searched recursively. Unique file names must be given to each file, regardless of their absolute paths.
- Edit the `CALIB_DATA_DIR` to specify the location of static calibration directory (if necessary).
- Edit the bookkeeping, logs, temporary products, and final products directory as needed.
- Select the appropriate wavelength range of the final datacube by editing the **LamMin** and **LamMax** parameters (see Section 6.3).
- Press  to start the workflow.

The workflow will automatically perform the following steps:

1. Exposures are grouped belonging to the same object (defined by the header keyword `HIERARCH ESO OBS NAME`) and the same instrument setup (defined by the header keyword `HIERARCH ESO INS MODE`). A Dataset chooser will pop-up, showing the datasets to be aligned and combined together. If one desires to combine all the exposures in the input data directory regardless of the object name, this can be done by editing the OCA rules in this way (before starting the workflow).

- Open the muse_wkf.oca file with an editor. To see its location it is sufficient to double click on the Data Organizer actor (inside the CombineExposures composite actor) and look at the "OCA file" field.
- Change the command (approx at line 397 of the OCA file):

```
select execute(muse_exp_combine) from inputFiles
  where REFLEX.CATG == "OBJECT" or REFLEX.CATG ==
  "PIXTABLE_REDUCED" group by OBS.TARG.NAME,
        INS.MODE as (TPL_A, combined_cubes);
```

with the command:

```
select execute(muse_exp_combine) from inputFiles
  where REFLEX.CATG == "OBJECT" or REFLEX.CATG ==
  "PIXTABLE_REDUCED" group by INS.MODE as
      (TPL_A, combined_cubes);
```

Note that the grouping by instrument mode is still active.

- Change the section (approx at line 1308 of the OCA file):

```
action muse_exp_combine
{
 minRet = 2;  maxRet = 2000;
  select file as PIXTABLE_REDUCED from calibFiles where
  REFLEX.CATG == "PIXTABLE_REDUCED"
  and inputFile.INS.MODE==INS.MODE
  and inputFile.OBS.TARG.NAME==OBS.TARG.NAME;

 minRet = 2;  maxRet = 2000;
  select file as IMAGE_FOV from calibFiles where
  REFLEX.CATG == "IMAGE_FOV"
  and inputFile.INS.MODE==INS.MODE
  and inputFile.OBS.TARG.NAME==OBS.TARG.NAME;
```

With the section:

```
action muse_exp_combine
{
 minRet = 2;  maxRet = 2000;
  select file as PIXTABLE_REDUCED from calibFiles where
  REFLEX.CATG == "PIXTABLE_REDUCED"
  and inputFile.INS.MODE==INS.MODE;

 minRet = 2;  maxRet = 2000;
```

```
select file as IMAGE_FOV from calibFiles where
REFLEX.CATG == "IMAGE_FOV"
and inputFile.INS.MODE==INS.MODE;
```

Note that the grouping by instrument mode is still active.

- Save the OCA rule file. It is recommended to use a different name for the new OCA rule file.
- Specify the new name of the OCA rule file inside the Data Organizer and press “Commit”.
- Press  to start the workflow.

2. Execution of the **muse_exp_align** recipe for image alignment, as detailed in Section 7.1.
3. Combination of the `PIXTABLE_REDUCED` into an unique datacube using the **muse_exp_combine** recipe and the offsets defined in the `OFFSET_TABLE`. A reconstructed image (`IMAGE_FOV`) will also be produced.
4. Final results are copied into the directory: `END_PRODUCTS_DIR/ <timestamp>` , where `<timestamp>` is the time stamp of the latest `EsoReflex` execution.
5. If set by the general parameter `ProductExplorerEnabled`, the product explorer will show up, showing the product association tree and allowing the inspection of each file in the data reduction chain.

To disable the alignment procedure (but still leaving the combination active), open the CombineExposure actor (right mouse button and select Open Actor), locate the muse_exp_align_1 recipe executer and configure it (right mouse button and select Configure Actor). Set the recipe mode from “Run” to “Disable”.

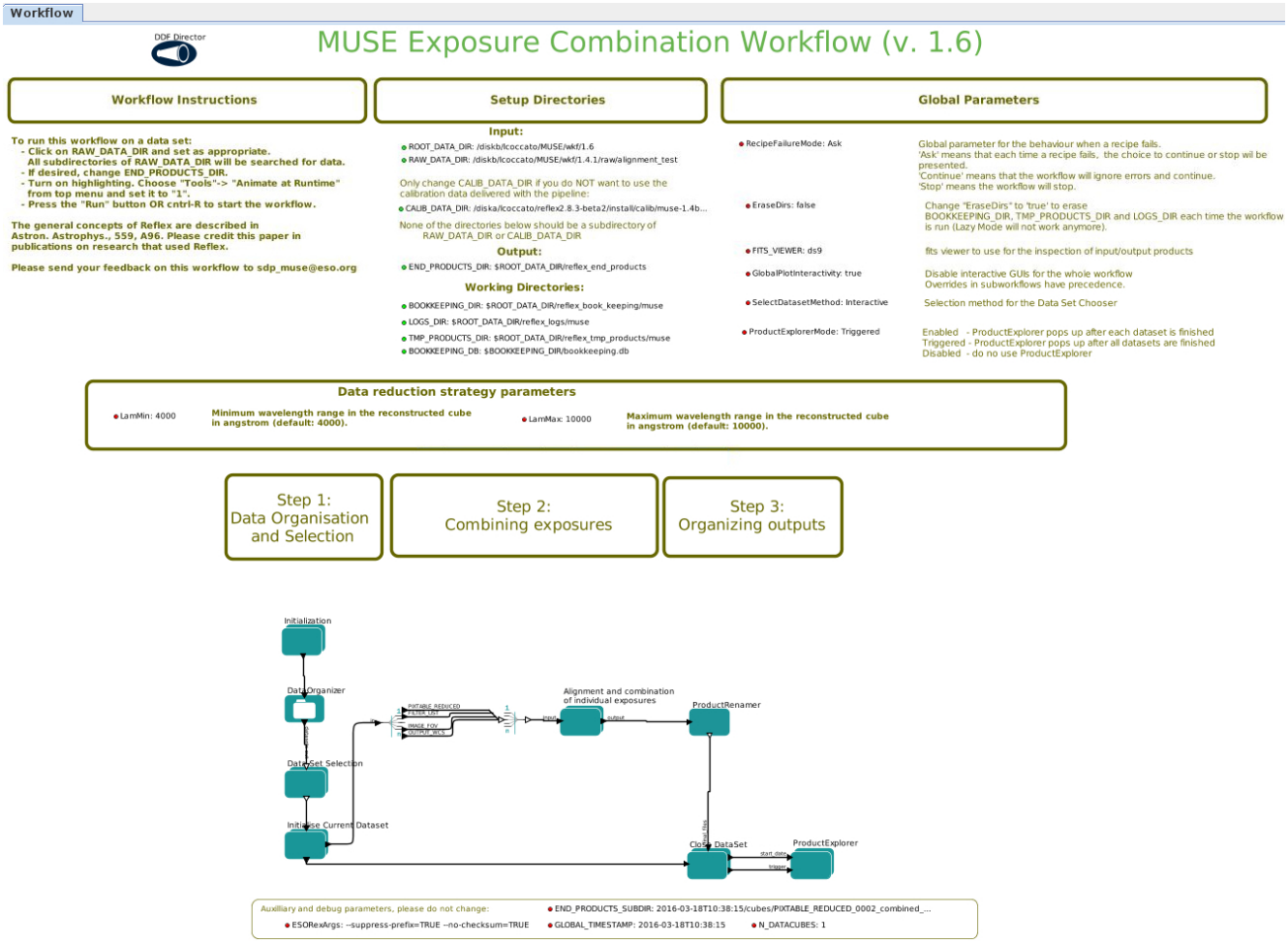


Figure 7.1: The MUSE `Reflex` `muse_exp_combine.xml` workflow.

8 Tips and tricks

8.1 Optimization of the automatic alignment

A full description of the MUSE recipe `muse_exp_align` and its parameters is provided in the MUSE pipeline manual. Here we provide a brief description and some tips. Recipe parameters can be edited by configuring the RecipeExecuter associated to the **muse_exp_align** recipe, which can be found by opening the CombineExposures composite actor.

8.1.1 The algorithm in a nutshell

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`) expressed in in sigma above background RMS. If the number of sources is larger than the maximum allowed (`-srcmax`) the threshold is increased by the amount specified by the parameter `-step`. In 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 till the number of sources is within the allowed 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 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 by the source detection iterations.

8.1.2 Tips

Here we provide some tips that can help the user to optimize the alignment of the exposures to be combined, in the case the default recipe set-up fails.

- Too many (or too few) sources were detected in one image. If the case, an error message pops up and the user has the option to stop the workflow or to continue using header information for alignment. This situation could be avoided by increasing/decreasing the threshold (recipe parameter `-threshold`), or the threshold step (recipe parameter `-step`), or the number of iterations (`-iterations`). Alternatively, one can modify the permitted minimum and maximum number of detected sources (recipe parameters, `-srcmin` and `-srcmax`, respectively). In the case the field of view is populated by extended sources (e.g. group of galaxies) it might be helpful to increase the FWHM of the convolution kernel for detection sources (recipe parameter `-fwhm`).

- The maximum number of allowed stars is detected, but the alignment is not correct. This can happen in crowded fields, and the alignment algorithm does not find the correct matching between the sources detected in different frames. There are two tricks to overcome this issue. First, one could either decrease the number of maximum sources allowed in the detection. However, this solution is not advisable for

mosaicing, because the number of reference sources in the overlapping regions could be too small. Second, if the offsets are known to be small [7], the user can decrease value of the first search radius and set the corresponding recipe parameter (`-rsearch`) to, for example, 10.,4.,2.,0.8. The first entry (10 in this example) forces the matching algorithm to find offsets of at most 10 arcseconds. If the first entry of `-rsearch` is modified, it might be required to change the `-nbins` parameter; a value twice the first `-rsearch` value is usually a good choice (`-nbins=20` in the example above).

- Crowded field, dithered exposures. If the field is very crowded, but the exposures are targeting the same field, it might be advisable to decrease the maximum number of maximum sources (recipe parameter `-srcmax`) to avoid problems in the matching algorithm. Alternatively, if there are brigth sources in the field (e.g., if targeting globular clusters), it is advisable to increase the `-threshold`) and the `-step`) parameters.

- Crowded field, mosaicing exposures. Although a large number of sources are found, it might be necessary to increase the minimum or maximum number of sources (recipe parameters `-srcmin` and `-srcmax`, respectively) to ensure enough stars in the overlapping region between the exposures. If the large number of sources can create confusion, decrease the first entry of the `-rsearch` recipe parameter.

- Sparse field, the minumum number of sources is detected, but the alignment is not correct. This can be solved by increasing the number number of sources to detect (with the risk to introduce too many spurious detections), or by decreasing the first entry of the `-rsearch` recipe parameter.

8.2 Optimization of the empty sky regions for sky background evaluation

The pipelines recipes **muse_create_sky** and **muse_scipost** (which are executed within the composite actors `Sky` and `Science`, respectively) produce a file with category `SKY_MASK`. This file is a mask that specifies the spaxels in the field of view that were used to compute the sky by the recipes. However, these recipes can accept a `SKY_MASK` as input as well. In this case, the recipe will use the input mask and will not calculate one from scratch.

If the user wants to use his/her own masks in a dataset, (s)he can include them within the raw data directory. The Data Organizer in Reflex associates for each file (target or sky exposure) the mask that has its same MJD-OBS, if present.

If you use custom sky massk, please remember to set the SkyFr_1 and SkyFr_2 parameters accordingly to the needs in the workflow main canvas.

The user can prepare the masks with any tool, as long as their structures are compatible with the muse pipeline. The most convenient way of doing it is to edit the masks previously created by the workflow, which are stored in the temporary directory of the corresponding recipe: muse_create_sky_1/latest/products_dir/ contains the masks created by **muse_create_sky** in its latest execution, whereas muse_scipost_1/latest/products_dir/ contains the masks created by **muse_scipost** in its latest execution.

[7] The user can estimate of the offsets by looking at the combined image obtained without the alignment. To skip the alignment, open the CombineExposure actor, locate the RecipeExecuter `muse_exp_align_1`, click with the right mouse button on it and set the recipe mode to skip in the RecipeExecuter configuration window, and press commit. Remember to set the recipe mode to run after it.

8.3 Optimization of the sky removal via post-processing: the muse_zap workflow

Typically, the sky removal leave residuals of the order of ∼5% of the sky signal in the reduced datacubes. These residual can be higher in the case the sky has been evaluated on dedicated exposures, because of the time difference between the sky and target observations.

The current MUSE workflow release, contains an additional workflow, named muse_zap, that is designed to run the ZAP code by Soto et al. (2016, MNRAS, 458, 3210) to better remove the sky residuals on the final datacubes (either from single exposures or combined).

The workflow is visible via the command:

```
esoreflex -l
```

A dedicated tutorial on how to use the workflow is available at http://www.eso.org/sci/software/pipelines/

Note. The cleaning of residual sky lines via the ZAP algorithm might require the characterization of sky residuals on dedicated sky exposures, if available for a dataset. By default the MUSE workflow does not produce a datacube with sky residuals. To enable it, please set the configuration parameter `Create Datacube with sky residuals` to true. This will create the sky-subtracted pixeltables, datacubes, and reconstructed images of each individual sky exposures. These files are saved in the end product directory (as specified by `END_PRODUCTS_DIR` in the main reflex canvas), under the current time stamp, inside a directory that have the same name of the dataset they refer to, with the additional suffix `-SkyResidualCubes`.

If your strategy foresees the combination of several sky residual cubes, you can run the `muse_exp_combine.xml` workflow to combine the desired sky residual pixeltables.

8.4 Using user-supplied response curves and telluric correction

Following the same principle of Section 8.2 it is possible to use a response curve `STD_RESPONSE` and a telluric correction `STD_TELLURIC` different from those produced by the workflow, as long as the structure of these files is compatible with the one produced by the pipeline. The most convenient way of doing it is to edit the files previously created by the workflow, which are stored in the temporary directory of the muse_standard recipe. For example, muse_create_sky_1/latest/products_dir/ contains the products of the latest execution of muse_standard.

In order to use the edited files, they need to be copied into the raw data directory. The workflow will associate the `STD_RESPONSE` and `STD_TELLURIC` which are closer in time to the observations as well as the raw observations of the standard stars (if available). By default, the workflow processes the raw standard star observations (if available).

To use the user-provided response curve and telluric correction, deselect the standard stars observations from the dataset and set **Telluric = 2** in the main reflex canvas.

To use the response curve from the observed star, but but the user-provided telluric correction, just set the **Telluric** parameter to 2 in the main reflex canvas (without deselecting the standard star from the dataset).

To disable the telluric correction set **Telluric = 0**. Please note that in regions where the contribution of the object is low, the telluric absorption features can be mis-interpreted as sky features, and partially removed by

the sky subtraction algorithm. The correction on telluric absorption features directly in the muse datacubes is recommended only for high signal to noise regions. To apply the correction derived from high S/N regions to the entire datacube it is recommended to convert the telluric correction into the same format as `STD_TELLURIC` and feed the pipeline with this file.

8.5 Execution of the workflow on computers with limited Memory

As discussed in Section 3.2 it is possible to analyze a short wavelength range and reduce a dataset on a computer with limited ram (e.g. 8 Gb of memory).

This can be done by changing the **LamMin**, **LamMax** parameters, as illustrated in Section 6.3. However, the recipe **muse_standard** is not affected by these parameters, and any change of the `-lambdamin` and `-lambdamax` parameters in the corresponding `RecipeExecuter` actor will cause the recipe to fail in allocating the needed memory and the workflow to crash.

To avoid that, one can exclude the raw frames of the standard star from the dataset (inspect the dataset via the Select Dataset window and deselect the muse_standard group of files). In this way the the response function provided in the static calibration directory (or a user-provided response curve, see Section 8.4) is used and the **muse_standard** recipe is not executed.

It might also be needed to deselect the twilight flats from your dataset, and set the parameters **RecomputeWCS** and **ComputeLSF** to false in the main workflow canvas.

8.6 Verification tools

The current MUSE pipeline distribution includes some verification and data-handling tools that are useful to verify the quality of the calibrations and inspect pixel tables.

These tools are not executed within the current workflow version, but they are available via command line. Depending on the configuration of your bash file, each verification tool `<tool_name>` can be launched either by typing its name on the terminal, or by typing the full path, i.e., `<install_dir>/bin/<tool_name>`.

The visual tools use gnuplot[8] for plotting[9]. All tools mentioned here give a usage hint when called without parameters.

For more information on these tools, we refer the reader to the MUSE pipeline manual.

8.6.1 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).

[8] Available from `http://www.gnuplot.info/`.

[9] 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.

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

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

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

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

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

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

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

Here, one has to be careful to select files that belong to the same IFU! Then one can visually verify that the polynomial solution matches the individual traced points.

Finally, one can also use the master flat-field product as background of the plot, so that one can actually check that the tracing points were correctly computed:

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

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

The widths of the slices on the CCD should be around 77 pixels, but their actual widths may slowly vary between top and bottom of the CCD, and between the slices near the edges and in the 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 8.2. A color gradient (from green on the left of the CCD to red on the right) plus different symbols are used to make the slices distinguishable. It is apparent that the slices on the edges of the CCD are the widest (above 78 pix) while those near the center of the CCD are narrow (below 76 pixels).

8.6.2 Verification of the wavelength solution

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

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

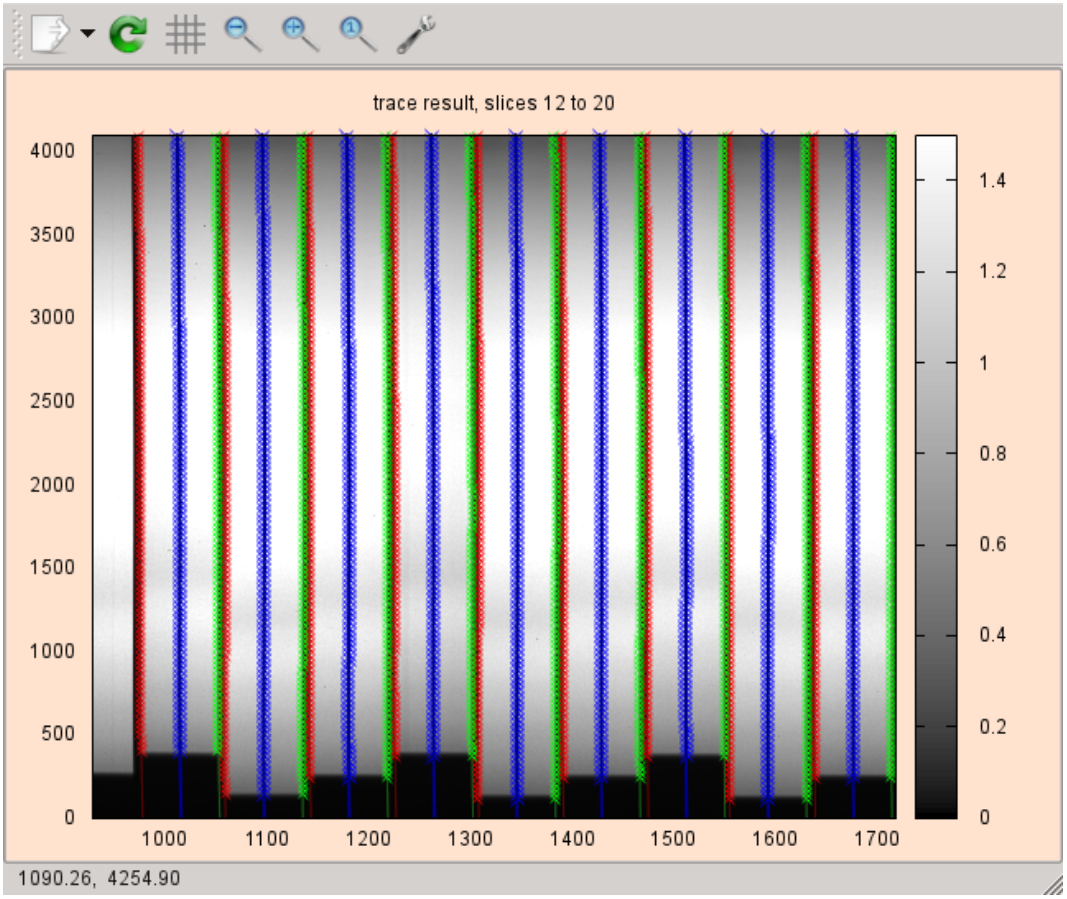


Figure 8.1: The graphical window showing the output of the `muse_trace_plot_samples` tool, then plotting slices 12 to 20 in IFU 6, using the trace samples table, the trace table, and the master flat-field image (see text for details).

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

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

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

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

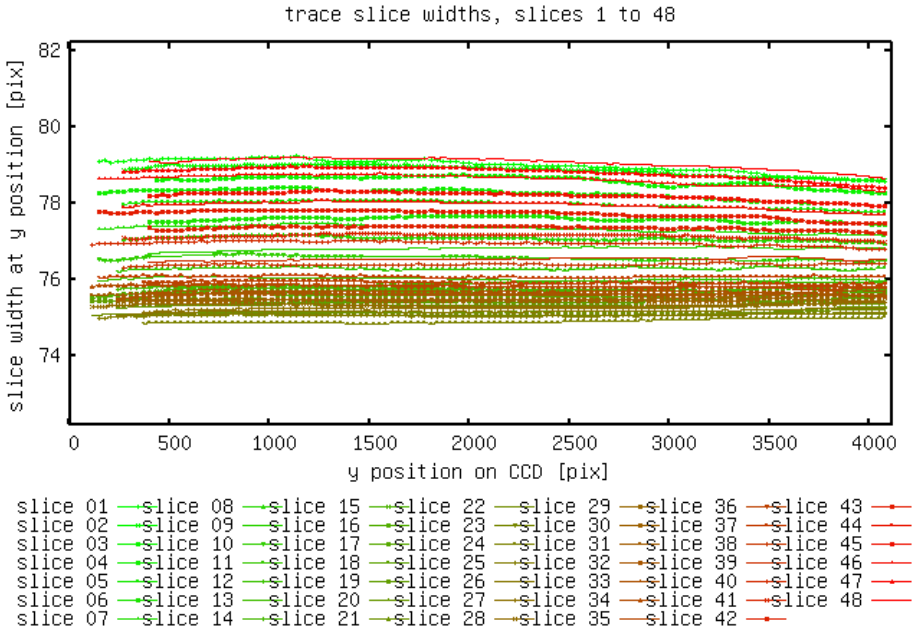


Figure 8.2: The graphical window showing the output of the `muse_trace_plot_widths` tool, plotting slices 1 to 48 of IFU 6, using the trace samples table (see text for details).

CCD column (`-c`). It is most useful when displaying the vertical axis as residuals, using `-r`. Figure 8.4 shows the output of the command

```
muse_wave_plot_column -s 12 -r WAVECAL_TABLE-10.fits \
     WAVECAL_RESIDUALS-10.fits
```

This is an example of a good calibration with low residuals (the final RMS for the solution in this slice was 0.030 Å). The tool has automatically selected all columns belonging to this slice and colored them according to their horizontal position on the CCD (green is left, red is right), and used different symbols. As one can see, the fainter arc lines (like the Ne I line at 5400.6 Å) have typically a much larger spread of residuals than the bright lines (e.g. 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.

8.7 Step by step product inspection

The current MUSE workflow does not include interactive actors that allow to inspect intermediate products or re-execute a certain recipe in order to fine tune the reduction. The interactivity features will be included in future workflow releases.

However, the `Reflex` workflow allows to “pause” the workflow in order inspect some intermediate products, such as the response curve or the mask automatically produced for sky-subtraction.

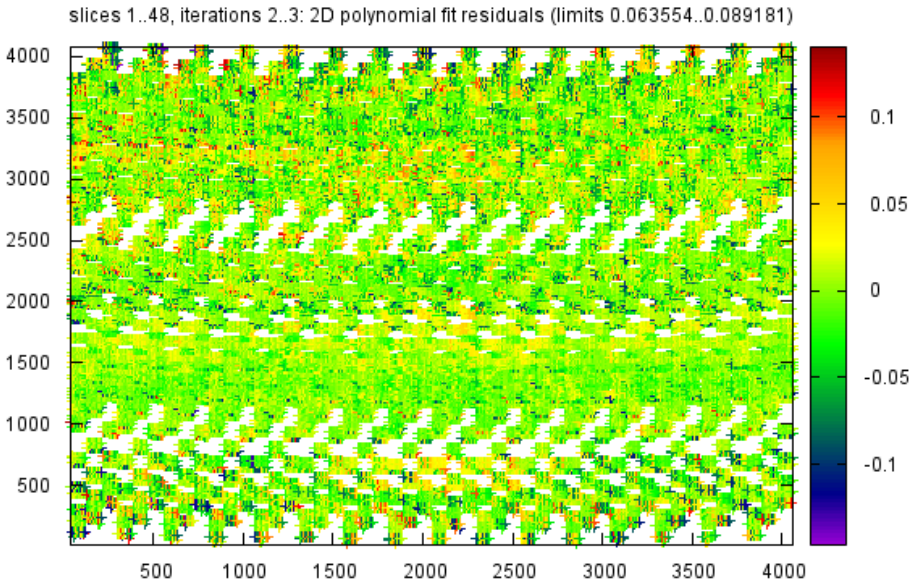


Figure 8.3: The graphical window showing the output of the `muse_wave_plot_residuals` tool, plotting all slices of IFU 10, using the wavelength calibration residuals table (see text for details).

To inspect or edit the products of the Standard Star and the Sky composite actors, follow these steps:

- Open the Standard Star (or the Sky) actor, by right-clicking with the mouse and selecting “Open Actor”. A subworkflow will appear.
- Locate and configure the DataFilter Actor in the subworkflow window, by double clicking with the mouse on it.
- Set the mode of the DataFilter actor from “Skip” to “Select”, and press the “Commit” button.
- Execute the workflow.

When the workflow encounters the DataFilter, it prompts a window with all the files that are traveling along that path. It is possible to inspect each single file, and, by dragging the mouse on the file name, to get the full path of the file.

The user can modify the `SKY_MASK` (produced either within the Sky or Science actor) or the `STD_RESPONSE` and/or `STD_TELLURIC` (produced by the Standard star actor). In order to be able to use the edited files, the user needs to stop the workflow, put these edited files into the raw data directory, and restart the workflow. Consult Sections 8.2 and 8.4 for further details.

It is also possible to insert a DataFilter anywere in workflow, and inspect the files that are broadcasted to it. Please, consult the Reflex Development manual for how to drag and connect different actors in the workflow: *http://ftp://ftp.eso.org/pub/dfs/reflex/reflex_dev_guide-1.0.pdf* .

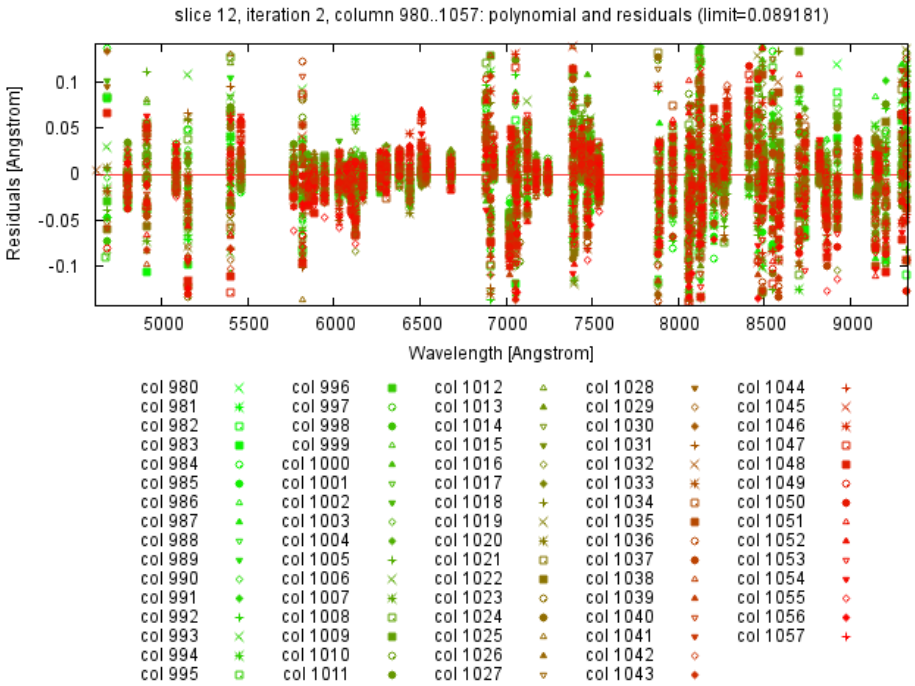


Figure 8.4: The graphical window showing the output of the `muse_wave_plot_column` tool, plotting slice 12 of IFU 10, using the wavelength calibration residuals and wavelength calibration tables (see text for details).

9 About The Reflex Canvas

9.1 Saving And Loading Workflows

In the course of your data reductions, it is likely that you will customise the workflow for various data sets, even if this simply consists of editing the `ROOT_DATA_DIR` to a different value for each data set. Whenever you modify a workflow in any way, you have the option of saving the modified version to an `XML` file using `File -> Export As` (which will also open a new workflow canvas corresponding to the saved file). The saved workflow may be opened in subsequent `Reflex` sessions using `File -> Open`. Saving the workflow in the default Kepler format (.kar) is only advised if you do not plan to use the workflow in another computer.

9.2 Buttons

At the top of the `Reflex` canvas are a set of buttons which have the following useful functions:

-  - Zoom in.
-  - Reset the zoom to 100%.
-  - Zoom the workflow to fit the current window size (Recommended).
-  - Zoom out.
-  - Run (or resume) the workflow.
-  - Pause the workflow execution.
-  - Stop the workflow execution.

The remainder of the buttons (not shown here) are not relevant to the workflow execution.

9.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the  ,  , and  buttons, respectively. A workflow is executed by clicking the  button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the  button, or the workflow may be paused by clicking the  button which will allow the current actor/recipe to finish execution before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the  button again.

10 The MUSE Workflow

The MUSE workflow canvas is organised into a number of areas. From top-left to top-right you will find general workflow instructions, directory parameters, and global parameters. In the middle row you will find five boxes describing the workflow general processing steps in order from left to right, and below this the workflow actors themselves are organised following the workflow general steps.

10.1 Workflow Canvas Parameters

The workflow canvas displays a number of parameters that may be set by the user. Under "Setup Directories" the user is only required to set the `RAWDATA_DIR` to the working directory for the DataSet(s) to be reduced, which, by default, is set to the directory containing the demo data. The `RAWDATA_DIR` is recursively scanned by the `Data Organiser` actor for input raw data. The directory `CALIB_DATA_DIR`, which is by default within the pipeline installation directory, is also scanned by the `Data Organiser` actor to find any static calibrations that may be missing in your DataSet(s). If required, the user may edit the directories `BOOKKEEPING_DIR`, `LOGS_DIR`, `TMP_PRODUCTS_DIR`, and `END_PRODUCTS_DIR`, which correspond to the directories where book-keeping files, logs, temporary products and end products are stored, respectively (see the Reflex User Manual for further details; [1]).

Under the "Global Parameters" area of the workflow canvas, the user may set the `FITS_VIEWER` parameter to the command used for running his/her favourite application for inspecting FITS files. Currently this is set by default to `fv`, but other applications, such as `ds9`, `skycat` and `gaia` for example, may be useful for inspecting image data.

By default the `EraseDirs` parameter is set to `false`, which means that no directories are cleaned before executing the workflow, and the recipe actors will work in Lazy mode (see Section 10.2.2), reusing the previous pipeline recipe outputs where input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the `EraseDirs` parameter to `true`, which forces the workflow to recursively delete the contents of the directories specified by `BOOKKEEPING_DIR`, `LOGS_DIR`, and `TMP_PRODUCTS_DIR`. This is useful for keeping disk space usage to a minimum and will force the workflow to fully rereduce the data each time the workflow is run.

The parameter `RecipeFailureMode` controls the behaviour in case that a recipe fails. If set to `Continue`, the workflow will trigger the next recipes as usual, but without the output of the failing recipe, which in most of the cases will lead to further fails of other recipes without the user actually being aware of it. This mode might be useful for unattended processing of large number of datasets. If set to `Ask`, a pop-up window will ask whether the workflow should stop or continue. This is the default. Additionally, the `Stop` mode will stop the workflow execution immediately.

The parameter `GlobalInteractivity` controls whether the interactive windows will appear for those windows which are *enabled* by default. The possible values are `true, false`. Take into account that some windows are disabled in the default configuration and therefore are not affected by this parameter.

The parameter `ProductExplorerMode` controls whether the `ProductExplorer` actor will show its window or not. The possible values are `Enabled` (the `ProductExplorer` displays at the end of the reduction of every selected dataet), `Triggered` (the `ProductExplorer` displays at the end of the reduction of the last selected dataet), `false` (the `ProductExplorer` does not shows up).

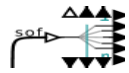
10.2 Workflow Actors

10.2.1 Simple Actors

Simple actors have workflow symbols that consist of a single (rather than multiple) green-blue rectangle. They may also have a logo within the rectangle to aid in their identification. The following actors are simple actors:

- 

- The `Data Organiser` actor.
- 

- The `Data Set Chooser` actor.
- 

- The `Fits Router` actor
- 

- The `Product Renamer` actor.
- 

- The `ProductExplorer` actor. This actor allows the user to inspect the product tree, to display the files, and to show the list of recipe calls used during data reduction.

Access to the parameters for a simple actor is achieved by right-clicking on the actor and selecting `Configure Actor`. This will open an "Edit parameters" window. Note that the `Product Renamer` actor is a jython script (Java implementation of the Python interpreter) meant to be customised by the user (by double-clicking on it).

10.2.2 Lazy Mode

By default, all recipe executer actors in a pipeline workflow are "Lazy Mode" enabled. This means that when the workflow attempts to execute such an actor, the actor will check whether the relevant pipeline recipe has already been executed with the same input files and with the same recipe parameters. If this is the case, then the actor will not execute the pipeline recipe, and instead it will simply broadcast the previously generated products to the output port. The purpose of the Lazy mode is therefore to minimise any reprocessing of data by avoiding data rereduction where it is not necessary.

One should note that the actor Lazy mode depends on the contents of the directory specified by `BOOKKEEPING_DIR` and the relevant FITS file checksums. Any modification to the directory contents and/or the file checksums will cause the corresponding actor when executed to run the pipeline recipe again, thereby rereducing the input data.

The forced re-reduction of data may of course be desirable.

To force a re-reduction of data for any single `RecipeExecuter` actor in the workflow (which will be inside the relevant composite actor), right-click the `RecipeExecuter` actor, select `Configure Actor`, and uncheck the Lazy mode parameter tick-box in the "Edit parameters" window that is displayed. This will force the re-execution of other downstream recipes that uses the products of the recipe for which the lazy mode has been disabled.

To disable entirely the lazy mode in `Reflex`, uncheck the Lazy mode parameter in "all" the `RecipeExecuters`.

To force a re-reduction a given dataset(s), one can change the name of the bookkeeping directory. A new reduction will start (with the lazy mode still enabled), but the results of previous reduction will not be re-used. Alternatively, one can set the `EraseDirs` parameter under the "Global Parameters" area of the workflow canvas to `true`. This will then remove all previous data that are stored in the bookkeeping, temporary, and log directories as well.

11 Frequently Asked Questions

- **The error window fills the whole screen - how can I get to the `Continue`/`Stop` buttons?**

  Press the Alt key together with your left mouse button to move the window upwards and to the left. At the bottom the `Continue`/`Stop` buttons will be visible. This bug is known but a fix has not yet been found.

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas, under Setup Directories) and organised further in directories by pipeline recipe.

- **Can I use different sets of bias frames to calibrate my flat frames and science data?** Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it ([1]). It is this purpose that is used by the workflow to send the correct set of bias frames to the recipes for flat frame combination and science frame reduction, which may or may not be the same set of bias frames in each case.

- **Can I run `Reflex` from the command line?** Yes, use the command:

  ```
  esoreflex -n <workflow_path>/<workflow>.xml
  ```

  The -n option will set all the different options to Kepler and the workflows that avoid any graphical display (including pipeline interactive windows).

  It is possible to specify workflow variables (those that appear in the workflow canvas) in the command line. For instance, to set the raw data directory can be done with this command:

  ```
  esoreflex -n -RAW_DATA_DIR <raw_data_path> \
               <workflow_path>/<workflow>.xml
  ```

  You can see all the command line options with the command `./esoreflex` -h.

  Note that this mode is not fully supported, and the user should be aware that the path to the workflow must be absolute and even if no GUI elements are shown, it still requires a connection to the window manager.

- **How can I add new actors to an existing workflow?** You can drag and drop the actors in the menu on the left of the `Reflex` canvas. Under `Eso-reflex -> Workflow` you may find all the actors relevant for pipeline workflows, with the exception of the recipe executer. This actor must be manually instantiated using `Tools -> Instantiate Component`. Fill in the "Class name" field with `org.eso.RecipeExecuter` and in the pop-up window choose the required recipe from the pull-down menu. To connect the ports of the actor, click on the source port, holding down the left mouse button, and release the mouse button over the destination port. Please consult the Reflex User Manual ([1]) for more information.

- **How can I broadcast a result to different subsequent actors?** If the output port is a multi-port (filled in white), then you may have several relations from the port. However, if the port is a single port (filled in black), then you may use the black diamond from the toolbar. Make a relation from the output port to the diamond. Then make relations from the input ports to the diamond. Please note that you cannot click to start a relation from the diamond itself. Please consult the Reflex User Manual ([1]) for more information.

- **How can I run manually the recipes executed by Reflex?** If a user wants to re-run a recipe on the command line he/she has to go to the appropriate reflex_book_keeping directory, which is generally reflex_book_keeping/<workflow>/<recipe_name>_<number> There, subdirectories exist with the time stamp of the recipe execution (e.g. 2013-01-25T12:33:53.926/). If the user wants to re-execute the most recent processing he/she should go to the `latest` directory and then execute the script `cmdline.sh`. Alternatively, to use some customized `esorex` command the user can execute

  ```
  ESOREX_CONFIG="INSTALL_DIR/etc/esorex.rc"
  PATH_TO/esorex --recipe-config=<recipe>.rc <recipe> data.sof
  ```

  where INSTALL_DIR is the directory where Reflex and the pipelines were installed.

  If a user wants to re-execute in the command line a recipe that used a given raw frame, the way to find the proper data.sof in the bookkeeping directory is via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as before.

  If a recipe is re-executed with the command explained before, the products will appear in the directory from which the recipe is called, and not in the reflex_tmp_products or reflex_end_products directory, and they will not be renamed. This doesn't happen if you use the `cmdline.sh` script.

- **Can I reuse the bookkeeping directory created by previous versions of the pipeline?**

  In general no. In principle, it could be reused if no major changes are in the pipeline. However there are situations in which a previously created bookkeeping directory will cause problems due to pipeline versions incompatibility. This is specially true if the parameters of the pipeline recipes have changed. In that case, please remove completely the bookkeeping directory.

- **I want to combine several exposures in a unique datacube, and I have only the `PIXTABLE_REDUCED` and `IMAGE_FOV`, but no raw nor master frames**

  Exposures can be combined in a single datacube even if raw and master calibration frames are not present, via the dedicated `muse_exp_combine.xml` workflow. Consult Section 7.2. Note that it is possible to force the output WCS by providing the desired `OUTPUT_WCS` file in the same directory of the `PIXTABLE_REDUCED`. In the case multiple `OUTPUT_WCS` are present, the closest in time will be selected. If you do not want to use the selected `OUTPUT_WCS` for a specific dataset, just unselect it from the Select Dataset window (see also the FAQ: "How can avoid a file for being used in a specific task?").

- **I am not interested in flux calibrating my data, can I deselect the standard stars from the dataset to speed the data reduction?**

  The response function is needed in the MUSE pipeline. However, one can use the response stored in the static calibration directory, without processing the standard star observations. This is automatically selected by the data organizer, however the response curve from the reduction of standard star will be used by default. If you unselect the `muse_standard` folder in the Select Dataset select window, the workflow will use the response curve from the static calibration.

- **I have my own response function and telluric correction, which I would like to use instead of those produced by the pipeline. How do I do it?** The instructions are detailed in Section 8.4.

- **How can avoid a file for being used in a specific task?**

  To remove files from a dataset (or from a subset), click on Inspect dataset to see the association tree and deselect the files that do not have to be used for that task.

  Note that this operation does not remove the file from your hard-disk.

  The deselected files can still be used in other datasets or other parts of the workflows. This means that a file used for many purposes is still used, except for the purpose that it has been deselected for. For example, a bias frame that has been deselected for the masterbias computation in the standard stars, can still be used in the masterbias computation in the science frame, if was selected by the workflow in first place.

  If one wants some specific files not to be used anywhere, the simplest option is to remove those files from the raw directory.

- **I have my own sky mask regions, can I use them and prevent reflex the creation of new ones?** Yes. Just follow the instruction on Section 8.2.

[1] Forchì V. *Reflex User's Manual*. ESO/SDD/DFS, *http://www.eso.org/gasgano/*, 0.7 edition, 2012. VLT-MAN-ESO-19000-5037. 42, 45