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 5.0

Date 2015-04-28

	Name	Date	Signature
Prepared:	L. Coccato, R. Palsa	2015-04-28	
Approved:	W. Freudling		
Released:	M. Sterzik		

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.

This page was intentionally left blank

Contents

1 Introduction And Scope

EsoReflex is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine (Kepler[1]) 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. (2013, A&A, 559, 96). Please reference this article if you use EsoReflex for your research.

EsoReflex 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.

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[2] and organises them into DataSets. Each DataSet contains one science object 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 EsoReflex 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 also allows the combination of exposures of the same object and the same instrument set-up, even from different Observing Blocks. The exposures can be automatically aligned before combining them, using reference bright objects in the field of view. The alignment of exposures requires the installation of few additional Python modules. If these additional modules are not installed, the exposures will still be combined using header coordinates for alignment. Further details are given in Section 7.1.

The MUSE EsoReflex 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 EsoReflex canvas.

The current MUSE EsoReflex distribution contains also an additional workflow, `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.

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 Data Reduction Cookbook available at: *http://www.eso.org/sci/software/pipelines/*.

[1] *http://kepler-project.org*
[2] *http://www.eso.org/sci/archive/calselectorInfo.html*

2 Software Installation

The software pre-requisites for `EsoReflex 2.6` may be found at:
http://www.eso.org/sci/software/pipelines/reflex_workflows

For Apple Mac OS X 10.9 installation is supported with MacPorts. Refer to the instructions provided here:
http://www.eso.org/sci/software/pipelines/reflex_workflows/macports.html

For other systems, please follow these instructions to install the `EsoReflex 2.6` software and demo data:

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 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.

2.0.1 Installation of additional Python modules and creation of the EsoReflex configuration file `muse.rc`

The MUSE workflow will reduce individual exposures, and then it will group and combine exposures that belong to the same object and same instrument mode. Exposures are aligned using header coordinates as reference.

However, in some case, header coordinates are not accurate enough, and the frames require an extra alignment.

The current MUSE `EsoReflex` distribution contains a Python script dedicated to the automatic alignment of exposures to be combined. The automatic alignment procedure uses bright sources in the field of view and requires the installation of additional Python modules.

The extra modules required to enable the extra alignment feature need to be installe separately. If the extra modules are not installed, the workflow will work normally, and it will use the header coordinates for alignment.

The extra modules that are required for the automatic alignment are:

- astropy v0.4 (http://www.astropy.org/)
- photutils (https://photutils.readthedocs.org)
- scikit-image v0.10 (http://scikit-image.org/)

Cython v0.21 (http://cython.org/) and pip might be required to install some of the above packages.

The recommended way to install these extra packages, without creating any conflict with previous Python package installations or other workflows, is using Miniconda. The procedure to follow is described below:

1. Download Miniconda from the website (python 2.7 version):
   http://http://conda.pydata.org/miniconda.html
2. Execute:
   `bash miniconda.sh`

   we recommend to install it into `$HOME/miniconda`. For example, if you home directory is `/home/username`, the miniconda installation directory will be: `/home/username/miniconda`.

   When asked to prepend the Miniconda install location to your `$PATH` in the .bashfile, select No.
3. Type the following commands:
   (a) `$HOME/miniconda/bin/conda create -n python_muse python=2 astropy=0.4 scikit-image=0.10 scipy matplotlib=1.4 pip cython=0.21`
   (b) `$HOME/miniconda/envs/python_muse/bin/pip install pyfits photutils`

By following the above procedures, we created a self-contained Python environment named `python_muse`, which is independent of other Python modules that might be present in your system.

We can now provide a configuration file to `EsoReflex`, which will contain the location of the above Python modules. To do that:

1. Copy the file

   ```
   <install_dir>/etc/esoreflex.rc
   ```

   into your home directory and rename it `muse.rc`. Any name and location are fine, as long as they will be correctly provided to the esoreflex execution script (see Section 3). Please not that <install_dir> is the `EsoReflex` installation directory (Section 2); for Linux users, this has been specified during the installation, for Mac users it is `/opt/local/`.
2. Modify the variable `esoreflex.python-command` inside `muse.rc` so that it uses the one in the `python_muse` environment.

   For example, if the old value is:

```
esoreflex.python-command=python
```

the new values will be:

```
esoreflex.python-command=<path_miniconda>/envs/<env_name>/bin/python
```

where `<path_miniconda>` is the location where miniconda has been installed (e.g. `/home/username/miniconda`), and `<env_name>` is the name of the Python environment that has been created with Miniconda (e.g. `python_muse`). According to the instructions given above, the `esoreflex.python-command` will be (single line):

```
esoreflex.python-command=/home/username/miniconda/envs/
        python_muse/bin/python
```

Note: The procedures described above have been tested on Apple OS X 10.9 platform via MacPort installation, on Fedora, Ubuntu, and Scientific Linux distributions.

3 Software Execution

To start EsoReflex, issue the command:

```
<install_dir>/bin/esoreflex -config ~/muse.rc
```

In the example above, we assume that the muse.rc configuration file is located in the home directory. If not, the full path to the muse.rc must be provided. Please consult Section 2.0.1 for instructions of how to create it.

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

A couple of notes:

- The provided muse.rc configuration file specifies which Python command to use in order to be able to use the Python modules needed by the alignment procedure (see Section 2.0.1).

  This set-up is optimized the MUSE workflow, but it might fail to execute other workflows. Therefore, to run other workflows, simply start EsoReflex on another terminal window, without providing the muse.rc configuration file.

- If the muse.rc file is not provided, or it is not properly edit, or the needed Python modules are not installed, the MUSE workflow will still run, process, and combine the exposures, using the coordinates in the header for alignment. If the case, the user will be notified during the execution of the workflow by a pop-up window.

4 System requirements

4.1 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)[3]
- 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 pipeline uses all the available cores (e.g. 24, in the configuration suggested above) and processes the data of the 24 MUSE IFUs *in series* (see the MUSE manual for a detailed description of the instrument). The serial or parallel execution of each recipe is set by the `nifu` recipe parameter[4]: 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 $1 < N < 24$ will process only the selected $N-$th IFU.

By default, the workflow is set to run the recipes *in parallel*, i.e. the `nifu` parameter is set to -1, and to use all the available cores. This 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 15 flats or 15 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.

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 Data Reduction Cookbook available at *http://www.eso.org/sci/software/pipelines/*.

[3]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.

[4]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.

4.2 JVM Memory set-up

The MUSE workflow need a sufficient amount of memory allocated to EsoReflex. The best way to set the memory allocation of EsoReflex is to run the reflex_set_memory script that is distributed with EsoReflex *before* starting EsoReflex. 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 EsoReflex by clicking on "Tools – JVM Memory Settings" in the menu bar. EsoReflex needs to be restarted for this change to be applied.

4.3 Execution on machines with less than 64 GB of memory

The MUSE pipeline and the EsoReflex 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 7.2.

5 Demo Data

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

Warning: The MUSE files that are downloaded through the ESO archive, are compressed with `fpack` (they have the `.fz` extension). In general, the files need to be uncompressed before running a workflow. However, for the special case of MUSE data, it is enough to rename the files removing the `.fz` extension.

- **Target exposures.** Frame Tags: `PRO.CATG=OBJECT`; `DPR.TYPE=OBJECT`
  - `MUSE_WFM-NOAO_OBS173_0069.fits`
  - `MUSE_WFM-NOAO_OBS173_0071.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

6 Quick Start: Reducing The Demo Data

In this Section we describe how to reduce the demo data supplied with the `Reflex` 2.6 release. The steps to follow are listed below:

1. Start the `EsoReflex` application by typing

   ```
   <install_dir>/bin/esoreflex -configure ~/muse.rc
   ```

   at the terminal command line. The configuration file `muse.rc` must be given to execute the alignment procedure via dedicated Python modules. If it is not given, the alignment will be done using the header coordinates. See Section 2.0.1 for how to create the configuration file.

   The empty `EsoReflex` canvas will appear (Figure 6.1). We refer to Section 8 for a description of the main elements of the canvas. Note: for other dataset, please consider to reduce the number of cores to be used by setting the environmental variable `OMP_NUM_THREADS=12` before starting `EsoReflex` (see Section 3.1).

2. If you have not set the Memory allocation before (see section 4.2), then set the JVM Memory settings, and restart `EsoReflex`.

3. Select the MUSE workflow by clicking on `File -> Open...` and then selecting the file `muse-1.0.4/muse.xml` in the file browser. *Note:* there is a second workflow in the directory, namely `muse_exp_combine.xml`, which will be described in Section 7.1.2.

   Once the workflow has been loaded, a new `EsoReflex` window will appear, containing the MUSE workflow, as shown in Figures 6.2.

4. To aid 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 (1 ms is recommended), and click OK.

5. Specify the location of the directories containing the data, static calibration, and the desired output directories. This can be done by double clicking with the mouse on the appropriate fields in the workflow canvas, under the "Set-up directories", and selecting the correct path on the file browser that will pop-up. In the current workflow installation, all the set-up directories are configured with your system by default, and they point to the location where the demo data and the required static calibrations are located. Further information on the set-up directories are given in Section 9.1. Be sure that the static calibration directory contains the `TRACE_TABLE-06` files (required by the workflow) and the `lsf_profile` files (required only if the ComputeLSF=false, see Section 6.1).

   `TRACE_TABLE-06` files can also be retrieved from:
   http://ftp://ftp.eso.org/pub/dfs/pipelines/muse/muse-calib-tracetables-ifu06.tar.gz

6. If the files in the input directory are compressed, uncompress them (or remove the `.fz` extension in the filenames). Demo data have already the `.fz` extension removed.

7. 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 6.1.

8. Press to start the workflow. The Data Organizer probes the input directories and group the files on the basis of header information and a set of `OCA`[5] rules to create the datasets to be reduced. The `DataChooser` window pops up (Figure 6.3), allowing the user to inspect the files that compose the datasets that have been created by the Data Organizer. Press “Continue” to continue the data reduction.

9. The workflow first processes the selected datasets separately. Then the reduced exposures that are belonging to the same object and the same instrument configuration are grouped together and combined. If all the additional Python modules are properly installed, exposures are aligned before combination using bright sources in common between adjacent exposures as reference. Header coordinates will be used otherwise, and a display window will notify the user.

   The workflow executes the following pipeline recipes:

   - **muse_bias**.
   - **muse_dark** (if dark frames are available).
   - **muse_flat**.
   - **muse_wavecal**.
   - **muse_lsf** (if ComputeLSF = true).
   - **muse_twilight** (if `SKYFLAT` observations are available).
   - **muse_scibasic**.
   - **muse_standard**.
   - **muse_create_sky** (if offset-sky observations are available).
   - **muse_astrometry** (if RecomputeWCS = true, and if `ASTROMETRY` observations are available).
   - **muse_scipost**.
   - **muse_exp_combine**.

   We refer the reader to the MUSE Cookbook for further information on these recipes and their parameters.

   If the global parameter ProvenanceExplorerEnables is set to true, the `EsoReflex` Provenance Explorer pups up at the end of each reduced data set reduction and the products association tree can be explored. Press “Continue” on the Provenance Explorer window to conclude the reduction.

   A second Provenance Explorer pups up at the end the image combination (if the global parameter ProvenanceExplorerEnables is set to true). Press “Continue” on the Provenance Explorer window to conclude the reduction.

   Warning: Due to the large amount of files produced by the MUSE data reduction pipeline, the current version of the `EsoReflex` Provenance Explorer might have difficulties to handle the entire file association tree. It might be therefore necessary either to click on “Continue” without exploring the files, or to

[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-1.0.4/reflex/, carrying the extension `.oca`

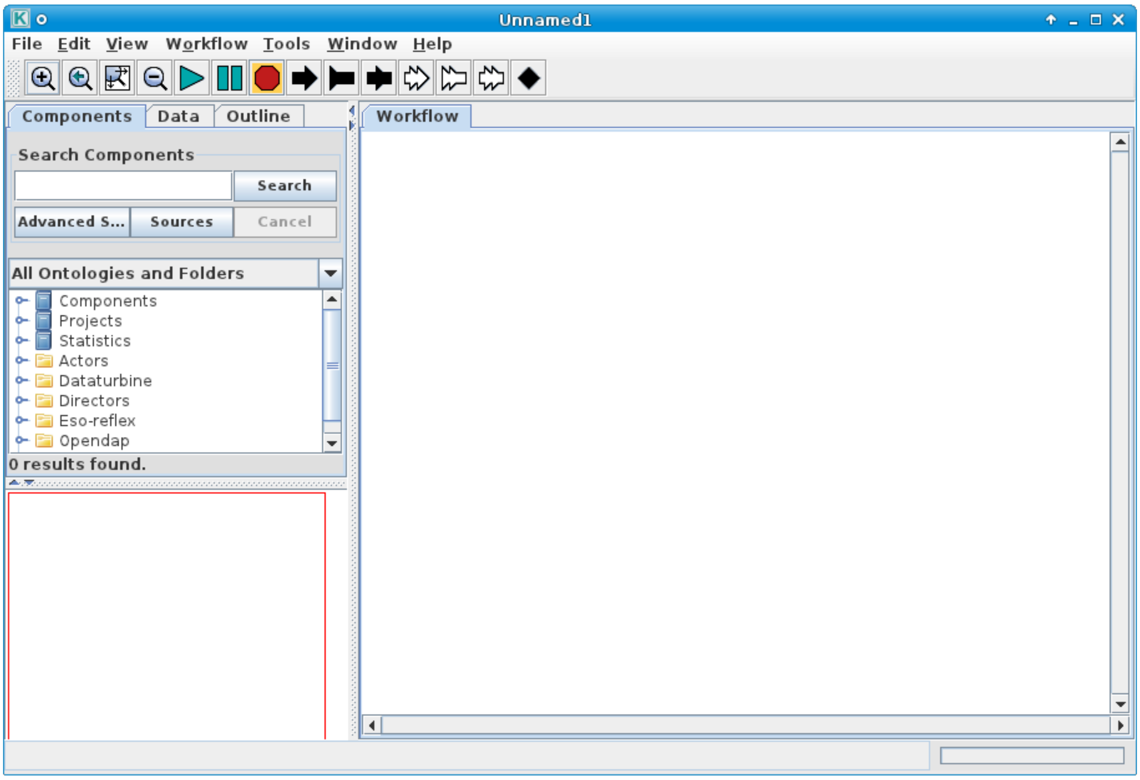


Figure 6.1: The empty Reflex canvas.

disable the Provenance Explorer (by setting to "false" the ProducExploreEnabled parameter in the main reflex canvas). This bug will be fixed in the next EsoReflex release.

The following final products DATACUBE_FINAL, IMAGE_FOV, and PIXTABLE_REDUCED are saved in the reflex_end_product directory. The exact name of the file 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.

6.1 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 desired RecipeExecuter is visible. To open a composite actor, click on it with the mouse right button, and select "Open Actor". To configure the desired 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 EsoReflex 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 6.4).

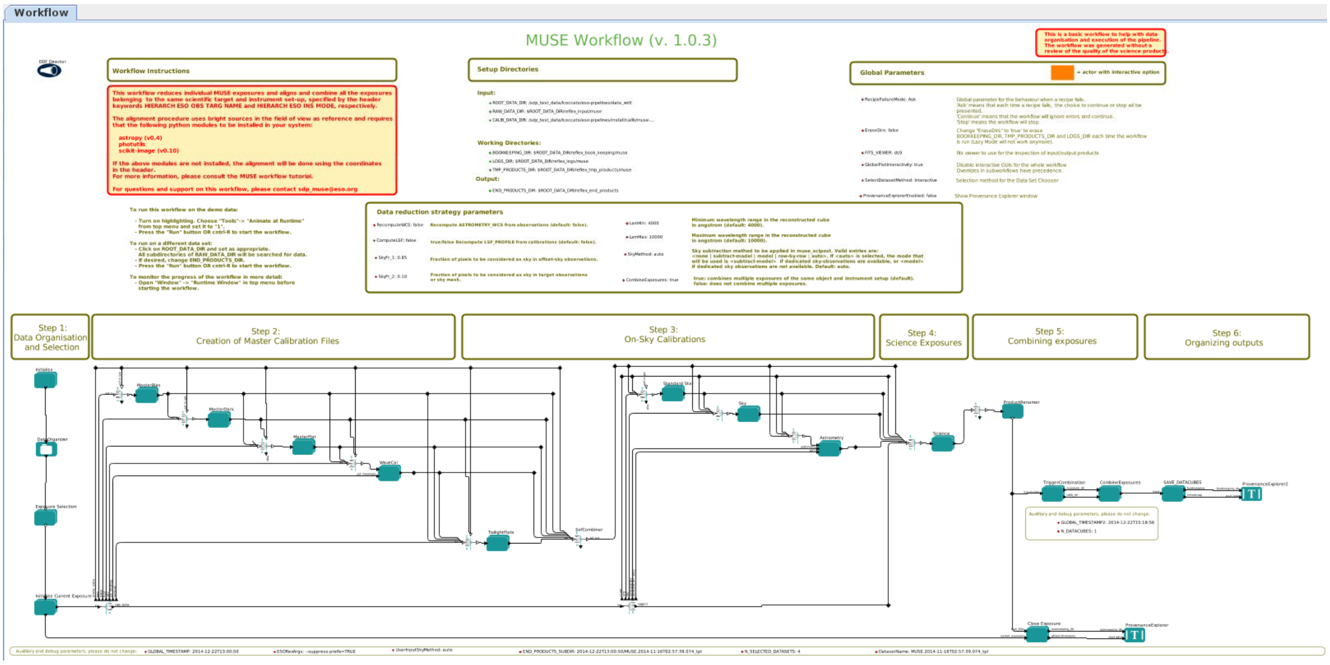


Figure 6.2: The MUSE `Reflex` `muse.wkf` workflow.

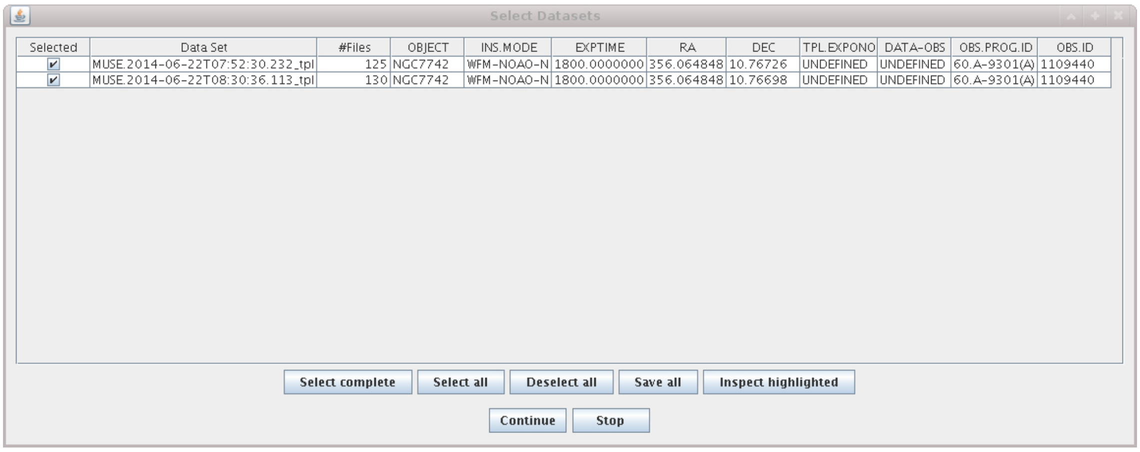


Figure 6.3: The Select Dataset window.

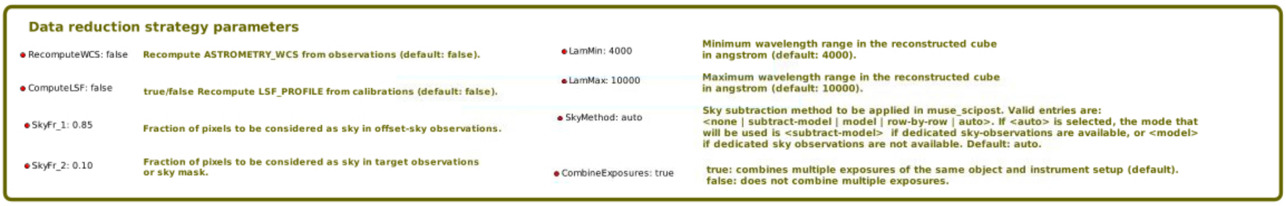


Figure 6.4: The Data reduction strategic parameters section in the `Reflex` canvas.

- **Calibrations parameters.**
  - **RecomputeWCS.** If set to *false*, the `ASTROMETRIC_WCS` are selected from the static calibration directory (recommended option). Otherwise, they will be recomputed from the `ASTROMETRY` calibrations available in the dataset. The dataset downloaded from the ESO archive via the CalSelector tool should already provide the `ASTROMETRIC_WCS`, which are computed with the `ASTROMETRY` observations closest in time to the scientific dataset. Default: false.
  - **ComputeLSF.** If set to *false*, the `LSF_PROFILE` are selected from the static calibration directory. With this option, the **muse_lsf** routine within the WaveCal actor will not be triggered. If the parameter is set to "true", the `LSF_PROFILE` will be recomputed from the `ARC` calibrations available in the dataset. With this option, the **muse_lsf** routine within the WaveCal actor will be triggered. The computation of the `LSF_PROFILE` is time consuming and memory demanding. If necessary, one can reduce the number of cores to use or set the nifu=0 in the **muse_lsf** recipe (See Section 3.1). The static calibration directory already contains precompiled `LSF_PROFILE` for the extended and nominal MUSE modes. However, it is recommended to create your own `LSF_PROFILE`, at least one per observing period. You can stop the workflow execution as soon as the first **muse_lsf** execution is over (or after the first dataset has been reduced), copy the `LSF_PROFILE` that has been produced into the static calibration directory, and then re-run the workflow with `ComputeLSF = false`. Default: false.
- **Wavelength range parameters.** If you are interested in a restricted wavelength range it is possible to create the `DATACUBE_FINAL` accordingly. The following parameters have an affect on the **muse_create_sky**, and **muse_scipost** recipes.
  - **LamMin**. Sets the minimum wavelength (in Å) to consider when reconstructing the datacube in the **muse_create_sky**, **muse_astrometry**, and **muse_scipost** recipes. It corresponds to the recipe parameter `-lambdamin`. Default: 4000.
  - **LamMax.** Sets the maximum wavelength (in Å) to consider when reconstructing the datacube in the **muse_create_sky**, **muse_astrometry**, and **muse_scipost** recipes. 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 will 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, i) manually set `–lambdabin` and `–lambdamax` to a short wavelength range, and ii) set the "RecipeExecute" mode to "continue". This can be achieved by opening the "Standard" composite actor (mouse right-button – Open Actor), and configuring the appropriate recipe parameters in the corresponding `RecipeExecuter` actor (mouse right-button – Configure Actor).

In this way, the workflow does not crash, but the final datacube will not be corrected for instrumental response or telluric absorption. If you want to correct for instrumental response, replace the `STD_RESPONSE` with the one provided in the static calibration directory (see Section 6.3).

- **Strategy for sky subtraction.** The MUSE pipeline (and therefore the workflow) evaluates the sky to be subtracted in two ways. 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. Alternatively, if dedicated sky observations are present, the sky can be evaluates using a specified fraction of spaxels in the reconstructed sky image. This fraction is specified by the **SkyFr_1** workflow parameter.

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

  – **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", "none", and "row-by-row". These values define the `–skymethod` parameter in the **muse_scipost** recipe, except for the "auto" mode (which is not recognized by **muse_scipost**). If dedicated sky frames are present, the auto mode will set `–skymethod` to "subtract-model"; if dedicated sky frames are not present in the dataset, the auto mode will set `–skymethod` to "model". Default: auto.

  If dedicated sky observations are present, good values could be **SkyFr_1** = 0.85, **SkyFr_2** = 0., 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** = 1.0 and **SkyMethod** = model (the latter is automatically set if the "auto" mode is selected); the parameter **SkyFr_1** has no effect.

  Warning: if **SkyMethod** = model, **SkyFr_2** cannot be 0, otherwise the workflow crashes.

- **Data reduction parameters.** The following parameter will determine the strategy for the construction of the final datacube.

  – **CombineExposures.** If it is set to true (default), it allows the combination of multiple exposures 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`). If all the Python modules for automatic image alignment are installed and the `muse.rc` configuration file is provided (see Section 2), then the exposures are also automatically aligned using bright sources in the field of view as reference before combination. Otherwise, coordinates in the header will be used for the alignment and a pop-up window will notify the user during the execution of the workflow.

7 Tips and tricks

7.1 Combination of multiple exposures

The `muse.xml` workflow processes individual exposures independently and, optionally, it combines exposures that belong to the same object and the same instrument configuration and generates a final combined datacube. The combination of exposures is triggered by the Strategic data reduction parameter **CombineExposures** (see Section 6.1).

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 combined cube.

The alignment of exposures can be done in three ways:

- Automatic identification and offset computation, with a Python script included in the `muse.xml` workflow, please consult the software requirements, Section 2. This is the suggested way; see Section 7.1.1 for further details of the executed steps. Obviously, the alignment procedure is not triggered if the combination of exposures is not enabled (i.e. if the `CombineExposures` parameter in the main workflow canvas is set to `false`, see Section 6.1).
- Automatic identification and offset computation 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 7.1.2 for this option.
- Manual identification of reference sources, offsets computation, and exposure combination via external execution of the **muse_exp_combine** recipe with the `esorex` command line; see Section 7.1.3 for this option.

7.1.1 Automatic combination within the `muse.xml` workflow

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

Then, if we have set the strategic parameter `CombineExposures = true` in the main Reflex canvas, the workflow groups together all the exposures that belong to the same object (defined by header keyword: `HIERARCH ESO OBS NAME`) and instrument configuration for further processing and combination (defined by the header keyword: `HIERARCH ESO INS MODE`).

A new dataset chooser window will be prompted, showing the datasets to be aligned and combined together. In the case of the demo dataset, only two exposures of the same dataset are available.

Once the “Continue” button is pressed, the workflow will automatically:

- Execute the Python actor for image alignment. This actor performs the following tasks:
  - Select the reference frame (the one with the oldest MJD-DATE header keyword).

- Identify sources in each IMAGE_FOV frame.
- Identify sources in common between adjacent frames.
- Compute the offsets of each frame with respect the reference frame.
- Update the RA, DEC, CRVAL1, and CRVAL2 header keywords the IMAGE_FOV and PIXTABLE_REDUCED frames, to account for the measured offsets.
- Add some additional header keywords to the frames to specify the number of reference sources used and the applied offsets.

- If the alignment procedure fails or the required Python modules are not installed, or the Python dependencies are not fulfilled, the coordinates in the header will be used for the alignment. If the failure is caused by not having the required Python modules installed, the user will be notified by a pop-up window.
- Combine the aligned PIXTABLE_REDUCED into an unique datacube using the **muse_exp_combine** recipe. A reconstructed IMAGE_FOV will also be produced.
- Link the final results into the directory: END_PRODUCTS_DIR/ <timestamp> cubes, where <timestamp> is the time stamp of the latest EsoReflex execution.
- If the general parameter ProvenaceExplorerEnabled is set to “true”, the provenance explorer will show up, and the final products can be inspected.

7.1.2 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. In this case, the muse_exp_combine.xml workflow can be used instead of the more general muse.xml. As in the case of muse.xml, this workflow requires the additional Python modules to run the automatic alignment procedure. If they are not present, the coordinates in the header will be used for alignment; in this case the user will be notified in a pop-up window.

To use this workflow:

- Select the MUSE workflow by clicking on File -> Open.... and then selecting the file muse-1.0.4/muse_exp_combine.xml in the file browser. A new window containing the workflow will appear (see Figure 7.1).
- Edit the EXPOSURE_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 6.1).
- Press  to start the workflow.

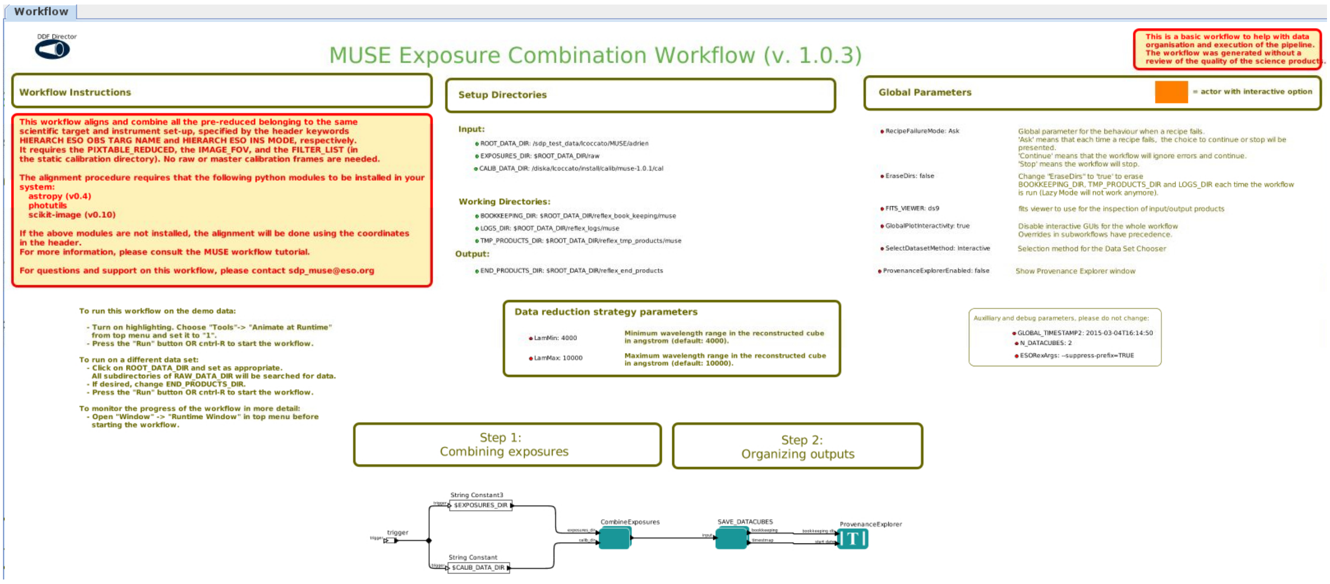


Figure 7.1: The MUSE `Reflex` `muse_exp_combine.xml` workflow.

As in the previous case, the workflow will:

1. Groups the exposures 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`).
2. Aligns the frames by executing the dedicated Python actor. If the automatic alignment fails, the coordinates in the header will be used for alignment.
3. Combine the exposures producing the final datacube.
4. If the general parameter `ProvenaceExplorerEnabled` is set to "true", the provenance explorer will show up at the end of all the exposure combination and the final products can be inspected.

7.1.3 Combination via external call of muse_exp_combine recipe

This solution is advisable 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.

The steps to follow for aligning the individual exposure to a reference frame, computing the offsets, and combining the pixel tables via external call of the **muse_exp_combine** recipe are listed below.

- Locate and display the `IMAGE_FOV` associated to the `PIXTABLE_REDUCED` that need to be combined. They have been produced by the workflow and they are located in a sub-directory inside `reflex_end_` product, which is named after the dataset they belong to. The exposures to be combined can belong to different OBs, but they must be produced with the same wavelength rage and instrument mode.

- Select a reference frame, with respect to all the other offsets must be computed. For example, the image with older DATE-OBS can be selected as reference image.
- Compare the individual field of views (i.e. IMAGE_FOV) to those of the reference image, and calculate the desired offsets in RA and DEC. The offsets are expressed in degrees on the sky plane (i.e. do not apply the $\cos\delta$ correction) and defined as follow:

```
RA_OFFSET = RA_MEASURED - RA_REFERENCE
DEC_OFFSET = DEC_MEASURED - DEC_REFERENCE
```

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

- Merge the PIXTABLE_REDUCED into a final datacube by specifying the offsets using **muse_exp_combine**. An example command to type on the terminal is the following (single command line):

```
MUSE_XCOMBINE_RA_OFFSETS="0.0000000, -5.4920943e-05, -0.00031381379" \
MUSE_XCOMBINE_DEC_OFFSETS="0.0000000, 8.3338105e-05, 7.2223397e-05" \
esorex muse_exp_combine  pixtab_reduced.sof
```

  where pixtab_reduced.sof is an ASCII file containing the names of the PIXTABLE_REDUCED to be combined, other (optional) input files, and their tags.

An example of pixtab_reduced.sof is given below:

```
filter_list.fits                               FILTER_LIST
PIXTABLE_REDUCED_2014-06-22T07:52:30.231.fits  PIXTABLE_REDUCED
PIXTABLE_REDUCED_2014-06-22T08:30:36.113.fits  PIXTABLE_REDUCED
```

7.2 Execution of the workflow on computers with limited Memory

As discussed in Section 4.2 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 6.1. 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. This is due to a bug in the **muse_standard** recipe.

To avoid that:

- open the "Standard" composite actor (mouse right-button – Open Actor), and configure the `RecipeExecuter` associated to the **muse_standard** recipe (mouse right-button – Configure Actor).
- manually set `-lambdabin` and `-lambdamax` to a short wavelength range inside the `RecipeExecuter` actor associated to **muse_standard**. This can be done
- set the `Recipe execution mode` to "Continue".

In this way **muse_standard** will still fail, but the workflow will not crash and the final datacube will not be corrected for instrumental response or telluric absorption. If you want to correct for instrumental response, you can replace the output of **muse_standard** by following the advance editing features described in the next Session.

7.3 Step by step product inspection and editing: expert users

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 `EsoReflex` workflow is not a mere black box: it is indeed possible to "pause" the workflow in order inspect some intermediate products, such as the response curve or the computed sky spectrum.

Editing the intermediate products is also possible, but there is the risk that the edited files are not compatible with the requirements of the subsequent pipeline recipes (e.g. one extension is removed during the editing, or the header is not maintained). Also, the modification via external procedure will not be documented in the bookkeeping directories and in the reflex database: this makes the data reduction hard to reproduce. The user is responsible for the proper editing, integrity of the fits header, and documentation of all the steps.

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 file can be edited or substituted (it must be overwritten); the workflow can be then continued by pressing "Continue". It is also possible to manually re-execute the recipe, as the sof file and the recipe parameters are stored in the corresponding bookkeeping directory, and replace any file with the newly generated one.

If you want to inspect other products (i.e. master flats), you can insert a DataFilter anywhere in the workflow; see the Reflex Developer 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* .

Warning: It is important to realize that, because of the lazy mode, any subsequent execution of the workflow will re-use the edited file version (if all the conditions that triggers the lazy mode on a recipe are fulfilled, see Section 8.2.2). This is because the modifications occurred after the execution of the recipe (e.g. **muse_sky**). If this is not desired, remember to de-activate the lazy mode for that recipe before future workflow executions: open the composite actor (i.e. Sky), configure the recipe (right mouse click), unmark the lazy mode check-box, and press “Commit”.

The trick of editing of the files by pausing the workflow via the DataFilter, is meant to be for expert users, and it is meant to be a temporary solution before the workflow will be upgraded with interactive actors. If the editing is not done properly (e.g the structure of the file and its extensions, or the header are not maintained) the workflow might fail or give wrong results.

8 About The Reflex Canvas

8.1 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 format (.kar) is only advised if you do not plan to use the workflow in another computer.

8.2 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.

8.3 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. Note that after clicking the button, it is possible that more than one actor is executed, since this behaviour depends on the workflow scheduling. For instance, if there are two actors in parallel, and you pause the workflow while one is being executed, then both of them will be executed before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the button again.

9 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.

9.1 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 9.2.2), 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 realising 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 inmediatly.

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 `ProvenanceExplorerEnabled` controls whether the `ProvenanceExplorer` actor will show its window or not. The possible values are `true, false`.

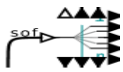
9.2 Workflow Actors

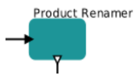
9.2.1 Simple Actors

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 `Provenance Explorer` actor.

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).

9.2.2 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 rereduction of data at each execution may of course be desirable. To force a rereduction of all data for all `RecipeExecuter` actors in the workflow (i.e. to disable Lazy mode for the whole workflow), set the `EraseDirs` parameter under the “Global Parameters” area of the workflow canvas to `true`. This will then remove all previous results as well. To force a rereduction 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.

10 Test Cases

The MUSE EsoReflex workflow 1.0.4 has been tested with the following obserational set-up detailed in Table 10.1:

Table 10.1: Files contained in the datasets on which the workflow has been tested (including both Nominal and Extended modes):

FRAMES	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Offset sky observations	No	No	Yes	Yes
Sky flat observations	No	No	No	No
Astrometric observations	Yes (used)	Yes (not used)	Yes (used)	Yes (not used)
LSF_PROFILE computed from raw	No	No	No	No
FRAMES	Dataset 5	Dataset 6	Dataset 7	Dataset 8
Offset sky observations	Yes	Yes	Yes	Yes
Sky flat observations	Yes	Yes	Yes	Yes
Astrometric observations	Yes(used)	Yes (not used)	No	No
LSF_PROFILE computed from raw	No	No	No	Yes

11 Frequently Asked Questions

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas) 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 launch `Reflex` from the command line?** Yes, use the command:

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

  The -n option will set all the different optiions to Kepler and the workflows that avoid any graphical display (including pipeline interactive windows). 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 `EsoReflex` 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.txt`. Alternatively, to ensure that the path to `esorex` is the correct one, the user can execute

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

where INSTALL_DIR is the directory where Reflex and the pipelines were installed. If the user knows the name of the input raw files for the recipe, the correct directory among the many time stamps can be found via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as 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.

- **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 7.1.2.

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