



European Organisation for Astronomical Research in the Southern Hemisphere

Programme: VLT

Project/WP: Science Data Products Group

EsoReflex ZAP tutorial

Document Number: ESO-287180

Document Version: 6.0

Document Type: Manual (MAN)

Released on: 2023-05-01

Document Classification: Public

Prepared by: L. Coccato

Validated by:

Approved by:

Name

This page was intentionally left blank



EsoReflex ZAP tutorial

Doc. Number: ESO-287180
Doc. Version: 6.0
Released on: 2023-05-01
Page: 3 of 40

Change record

| Issue/Rev. | Date | Section/Parag. affected | Reason/Initiation/Documents/Remarks |
|------------|------------|-------------------------|--|
| 1.0 | 01-07-2016 | All | First official release |
| 2.0 | 01-03-2017 | All | Association of dedicated sky exposures. |
| 3.0 | 15-03-2018 | All | Upgrade to ZAP version 2.1 and esoreflex 2.9 |
| 3.1 | 10-05-2019 | 2.5 | Explicitly install wxPython 3.0 when using miniconda |
| 4.0 | 01-03-2020 | 2.x | Compatible with Python 3. Updated installation instructions. |
| 5.0 | 01-05-2020 | All | In sync with reflex 2.11 |
| 6.0 | 01-05-2023 | 7 | Few tipps added. |

This page was intentionally left blank



Contents

| | | |
|----------|---|-----------|
| 1 | Introduction and scope | 7 |
| 2 | Installation | 8 |
| 2.1 | Installing Reflex workflows via <code>macports</code> | 8 |
| 2.2 | Installing Reflex workflows via <code>rpm/yum</code> | 9 |
| 2.3 | Installation with the <code>install_esoreflex</code> script | 10 |
| 2.4 | Installation of Scikit-learn Python package | 11 |
| 2.5 | Installation of Python packages via <code>miniconda</code> | 12 |
| 2.6 | System requirements | 13 |
| 2.7 | Demo dataset | 13 |
| 3 | Quick Start: Reducing The Demo Data | 14 |
| 4 | About the main <code>esoreflex</code> canvas | 19 |
| 4.1 | Saving And Loading Workflows | 19 |
| 4.2 | Buttons | 19 |
| 4.3 | Workflow States | 19 |
| 4.3.1 | Simple Actors | 20 |
| 4.3.2 | Lazy Mode | 20 |
| 5 | The MUSE ZAP Workflow | 22 |
| 5.1 | Workflow Canvas Parameters | 22 |
| 5.2 | MUSE ZAP specific workflow parameters | 23 |
| 6 | Reducing your own data | 24 |
| 6.1 | The <code>esoreflex</code> command | 24 |
| 6.2 | Launching the workflow | 24 |
| 6.3 | Workflow Steps | 26 |
| 6.3.1 | Data Organisation And Selection | 27 |
| 6.3.2 | DataSetChooser | 28 |
| 6.4 | The processing cascade | 29 |
| 6.4.1 | The ProductExplorer | 29 |



| | | |
|----------|---|-----------|
| 7 | Removing the residuals of the sky background | 32 |
| 7.1 | Creation of Sky Mask | 32 |
| 7.1.1 | Description of the interactive window | 32 |
| 7.1.2 | Description of the parameters | 33 |
| 7.2 | Removing sky residuals via ZAP | 35 |
| 8 | Frequently Asked Questions | 37 |



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.

The current MUSE reflex distribution contains a dedicated workflow, `muse_zap.xml`, which makes use of the Zurich Atmosphere Purge code (ZAP, Soto et al. 2016, MNRAS, 458, 3210). It is a Python code that has been created to remove residual sky contamination from the MUSE datacubes by performing principal components analysis.

The current reflex workflow uses version 2.1 of the ZAP code, which supports MUSE observations taken with and without adaptive optics (AO and NOAO modes).

Typically, the datacubes produced by the MUSE pipeline contain a residual sky contamination of the order of $\approx 5\%$ of the sky signal. This residual contamination can be higher if the sky has been evaluated on dedicated exposures, because of the time difference between the sky and target observations. The ZAP code helps to remove this residual contamination.

The purpose of this document is to instruct the user on the use of the `muse_zap.xml` workflow to efficiently remove residual sky contamination from MUSE datacubes.

In the following, we assume that the user is familiar with:

- the basic concepts of esoreflex;
- the basic steps of the reduction with the MUSE pipeline, in particular with the products category names such as `DATA_CUBE_FINAL` and `IMAGE_FOV`;
- the ZAP code.

Reflex and the data reduction workflows have been developed at ESO and they are fully supported. If you have any issue with this particular workflow, please contact sdp_muse@eso.org for further support.



2 Installation

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

The recommended way is to use the package repositories if your operating system is supported. The `macports` repositories support OS X (Section 2.1), while the `rpm/yum` repositories support Fedora 33-35, Scientific Linux 7, and CentOS 7 (Section 2.2). For any other operating system it is recommended to use the `install_esoreflex` script (Section 2.3).

In addition, the workflow requires a Python package (Scikit-learn) that is not distributed via the installation channels mentioned above. The installation of Scikit-learn is described in Section 2.4.

The full list of Python requirements for running the muse ZAP workflow is:

- matplotlib
- wxPython
- Python 3
- Numpy (1.6.0 or later)
- Astropy (1.0 or later)
- SciPy (0.13.3 or later)
- Scikit-learn

Note: Python 3 is recommended, but the workflow is backward compatible with Python 2.7.

2.1 Installing Reflex workflows via `macports`

This method is supported for OS X operating system. It is assumed that `macports` (www.macports.org) and `java` are installed. If you have any problem with this installation method, please read the full documentation at 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 workflows 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 Reflex, the MUSE pipeline, demo data, and the workflows:

```
# sudo port install esope-muse-all
```




Other useful installation options are:

- To show the list of available top level packages for different instruments is given by:

```
port list esopipe-*
```

- To install only the ZAP workflow, type:

```
sudo port install esopipe-muse-contrib-wkf
```

This procedure installs the workflow, the recipes, and all the Python dependencies, except for the Scikit-learn Python package: see Section 2.4 for instructions on how to install it.

Note: Issues with Mac OS and python-astropy version 2.0.10 have been reported. These issues can be solved installing a newer version of astropy (verified that works with 2.0.12).

2.2 Installing Reflex workflows via rpm/yum

This method is supported for Fedora 33-35, CentOS 7 and Scientific Linux 7 (SL 7 operating systems. If you have any problem with this installation method, please read the full documentation at 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 workflows support and Reflex itself:

1. Configure the ESO repository

- If you are running Fedora 33-35, run the following commands:

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

- If you are running CentOS 7, run the following commands:

```
sudo yum install yum-utils ca-certificates yum-conf-repos
sudo yum install epel-release
sudo yum-config-manager --add-repo=ftp://ftp.eso.org/pub/dfs/
pipelines/repositories/stable/centos/esorepo.repo
```

- If you are running SL 7, run the following commands:

```
sudo yum install yum-utils ca-certificates yum-conf-repos
sudo yum install yum-conf-epel
sudo yum-config-manager --add-repo=ftp://ftp.eso.org/pub/dfs/
pipelines/repositories/stable/sl/esorepo.repo
```

2. To install the muse_zap workflow only, use:



```
sudo dnf install esopipe-muse-contrib-wkf # (Fedora 26 or newer)
sudo yum install esopipe-muse-contrib-wkf # (CentOS 7, SL 7)
```

3. To list the pipelines to install, use:

```
sudo dnf list esopipe-*all # (Fedora 26 or newer)
sudo yum list esopipe-*all # (CentOS 7, SL 7)
```

This procedure installs the workflow, the recipes, and all the Python dependencies, except for the Scikit-learn Python package: see section 2.4 for instructions on how to install it.

2.3 Installation with the `install_esoreflex` script

Step 1:

First make sure that all the system requirements specified in

http://www.eso.org/sci/software/pipelines/reflex_workflows/

are satisfied. In particular, the installation of the libffi libraries are fundamental for the installation of zap-specific recipes.

Libffi libraries are supposed to be installed in

```
/usr/include/ffi.h
/usr/include/ffitarget.h
```

Depending on the operative system and the libffi version, the needed files of this library (ffi.h and ffitarget.h) might be installed on a different location. Therefore it is necessary to create links to the expected locations.

Type:

```
locate ffi.h
locate ffitarget.h
```

to find the installation path of these libffi files, and link them to the expected path.

For example:

In the case of Red Hat EL, CentOS, Scientific Linux 7.x official repositories, it might be necessary to do the following:

```
sudo ln /usr/lib64/libffi-3.0.5/include/ffi.h /usr/include/ffi.h
sudo ln /usr/lib64/libffi-3.0.5/include/ffitarget.h /usr/include/ffitarget.h
```

In the case of Ubuntu installation, it might be needed to do the following:

```
sudo ln /usr/include/x86_64-linux-gnu/ffi.h /usr/include/ffi.h
sudo ln /usr/include/x86_64-linux-gnu/ffitarget.h /usr/include/ffitarget.h
```



Step 2:

Download the `install_esoreflex` script

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

and execute it by typing:

```
chmod u+x install_esoreflex
./install_esoreflex
```

Select the MUSE pipeline.

Step 3:

On the top of the esoreflex requirements common for the other workflows that are specified at:

```
http://www.eso.org/sci/software/pipelines/reflex\_workflows/
```

the `muse_zap` workflow requires the following Python packages: `astropy` and `scipy`. If they are not present in your system, please install them by executing:

```
sudo yum install python-astropy scipy
```

for Fedora systems; or

```
sudo apt-get install python3-astropy python3-scipy
```

for Ubuntu systems.

For Mac OS X type use:

```
sudo port install astropy py-scipy
```

This procedure installs the workflow, the recipes, and all the Python dependencies, except for the Scikit-learn Python package: see Section 2.4 for instructions on how to install it.

2.4 Installation of Scikit-learn Python package

In addition to the Python packages installed via the procedures described above, The ZAP code version 2.1 requires an additional Python package: Scikit-learn.

If the user has sudo-rights, the most convenient way to install Scikit-learn is typing:



```
pip install scikit-learn
```

on the command line.

Alternatively, the user can install all the Python dependencies via miniconda. This will be described in [Section 2.5](#).

2.5 Installation of Python packages via miniconda

If the user does not have sudo privileges to install all the Python packages on the system Python, (s)he can always install all the needed Python modules via miniconda, by creating a dedicated Python environment.

An example of installation of Python3 packages needed to run the MUSE ZAP workflow via miniconda is given by (one single command line):

```
conda create -n my_environment matplotlib wxPython astropy  
numpy scipy scikit-learn
```

Note, specify matplotlib=3.0 if using Python 2.7.

Note that this procedure installs only the Python dependencies. The workflow and the recipes have to be installed following the procedures described in [Sections 2.1 – 2.3](#).

Once you have your Python environment that contains all the prerequisites including astropy, scipy, and Scikit-learn you have to instruct esoreflex to use it instead of the default Python. To do so, proceed as follows:

1. Creation of a esoreflex configuration file. From your terminal type:

```
esoreflex -create-config
```

This will create a configuration file in you home, named `.esoreflex/esoreflex.rc`. If you want to specify a different name, e.g. `muse.zap`, type type:

```
esoreflex -create-config muse.zap
```

and it will create a configuration file named `muse.zap` in the working directory.

2. Edit the newly created configuration file and change the Python command call. The line to modify is:

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

which is located around line 25. The value you have to enter is the full path of Python executable of the environment you'd like to use. For example, if you environment is saved in:

```
/home/user/miniconda/envs/my_environment/
```



the line to insert in the configuration file is:

```
esoreflex.python-command=/home/user/miniconda/  
envs/my_environment/bin/python
```

3. Instruct esoreflex to use the newly edited configuration file. Start the muse_zap esoreflex workflow with the command line:

```
esoreflex muse_zap
```

In this way, esoreflex will use the desired Python environment that is specified in the configuration file saved in your home directory. If you have specified a different name for the configuration file, type

```
esoreflex -config <full_path>/zap.rc muse_zap
```

Please, specify the full path to the zap.rc configuration file.

2.6 System requirements

Handling MUSE datacubes with nominal spatial sampling and wavelength coverage is highly demanding in terms of memory. We recommend to use at least 20 Gb RAM for single pointing datacubes. For computers with limited resources one way to process the data is to use a small number of principal components. See Sections 6 and 7.2 for further information.

The ZAP code is parallelized, therefore it takes advantage of all the available processors to speed up the entire process.

The demo dataset was created by setting the spatial sampling of the cube to 0.8 arcsec/pixel (full-size datacubes have 0.2 arcsec/pixel), and can be processed with a 8 Gb RAM computer using default parameters.

2.7 Demo dataset

The current distribution of the muse_zap workflow contains two demo datasets.

The first dataset contains a fully reduced and sky subtracted datacube (name: CUBE1.fits, category: DATACUBE_FINAL) and its corresponding white-light image (name: FOV1.fits, category: IMAGE_FOV).

The second dataset contains a fully reduced and sky subtracted datacube (name: CUBE2.fits, category: DATACUBE_FINAL), its corresponding white-light image (name: FOV2.fits, category: IMAGE_FOV), a cube of residual sky background from a dedicated sky exposure (name: DATACUBE_SkyRes.fits, category: DATACUBE_FINAL), and its corresponding white-light image (name: IMAGE_SkyRes.fits category: IMAGE_FOV).

Note: a cube of residual sky background from a dedicated sky exposures can be obtained from the main muse reflex workflow (see the main muse reflex tutorial, Section 8.3).



3 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 ZAP demo data set supplied with the `esoreflex 2.11.0` release. By following these steps, the user should have enough information to perform a reduction of his/her own data without any further reading:

1. First, type:

```
esoreflex -l
```

If the `esoreflex` executable is not in your path, then you have to provide the command with the executable full path `<install_dir>/bin/esoreflex -l`. For convenience, we will drop the reference to `<install_dir>`. A list with the available `esoreflex` workflows will appear, showing the workflow names and their full path.

2. Open the `MUSE_ZAP` by typing:


```
esoreflex muse_zap&
```

Alternatively, you can type only the command `esoreflex` the empty canvas will appear (Figure 3.1) and you can select the workflow to open by clicking on `File -> Open File`. Note that the loaded workflow will appear in a new window. The `MUSE_ZAP` workflow is shown in Figure 3.2.

3. 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 .
4. Change directories set-up. Under “Setup Directories” in the workflow canvas there are seven parameters that specify important directories (green dots).

By default, the `ROOT_DATA_DIR`, which specifies the working directory within which the other directories are organised. is set to your `$HOME/reflex_data` directory. All the temporary and final products of the reduction will be organized under sub-directories of `ROOT_DATA_DIR`, therefore make sure this parameter points to a location where there is enough disk space. To change `ROOT_DATA_DIR`, double click on it and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the button to select the directory from a file browser. When you have finished, click to save your changes.

Changing the value of `RAW_DATA_DIR` is the only necessary modification if you want to process data other than the demo data

5. Click the  button to start the workflow
6. 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 to different places this creates a problem as `esoreflex` cannot decide which one to use.

7. The `Data Set Chooser` actor will be highlighted next and will display a “Select Datasets” window (see Figure 3.3) that lists the datasets along with the values of a selection of useful header keywords¹. 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. A full description of the options offered by the `Data Set Chooser` will be presented in Section 6.3.2.
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.
9. Once the reduction of all datasets has finished, a pop-up window called *Product Explorer* will appear, showing the datasets which have been 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 3.5 shows the *Product Explorer* window. A full description of the *Product Explorer* will be presented in Section 6.4.1.
10. After the workflow has finished, all the products from all the datasets can be found in a directory under `END_PRODUCTS_DIR` named after the workflow start timestamp. Further subdirectories will be found with the name of each 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 ZAP workflow that merit a look at the rest of this tutorial.

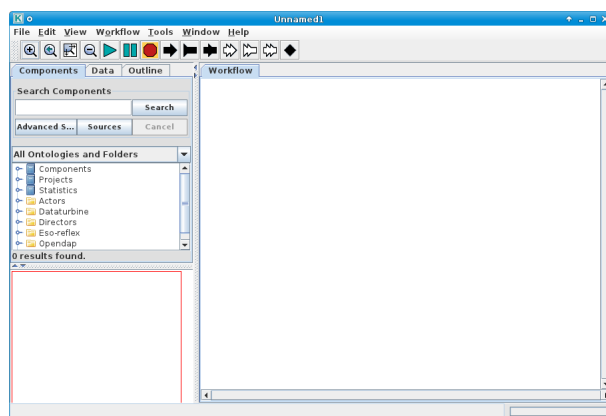


Figure 3.1: *The empty Reflex canvas.*

¹ The keywords listed can be changed by double clicking on the `DataOrganiser` Actor and editing the list of keywords in the second line of the pop-up window. Alternatively, instead of double-clicking, you can press the right mouse button on the `DataOrganiser` Actor and select `Configure Actor` to visualize the pop-up window.

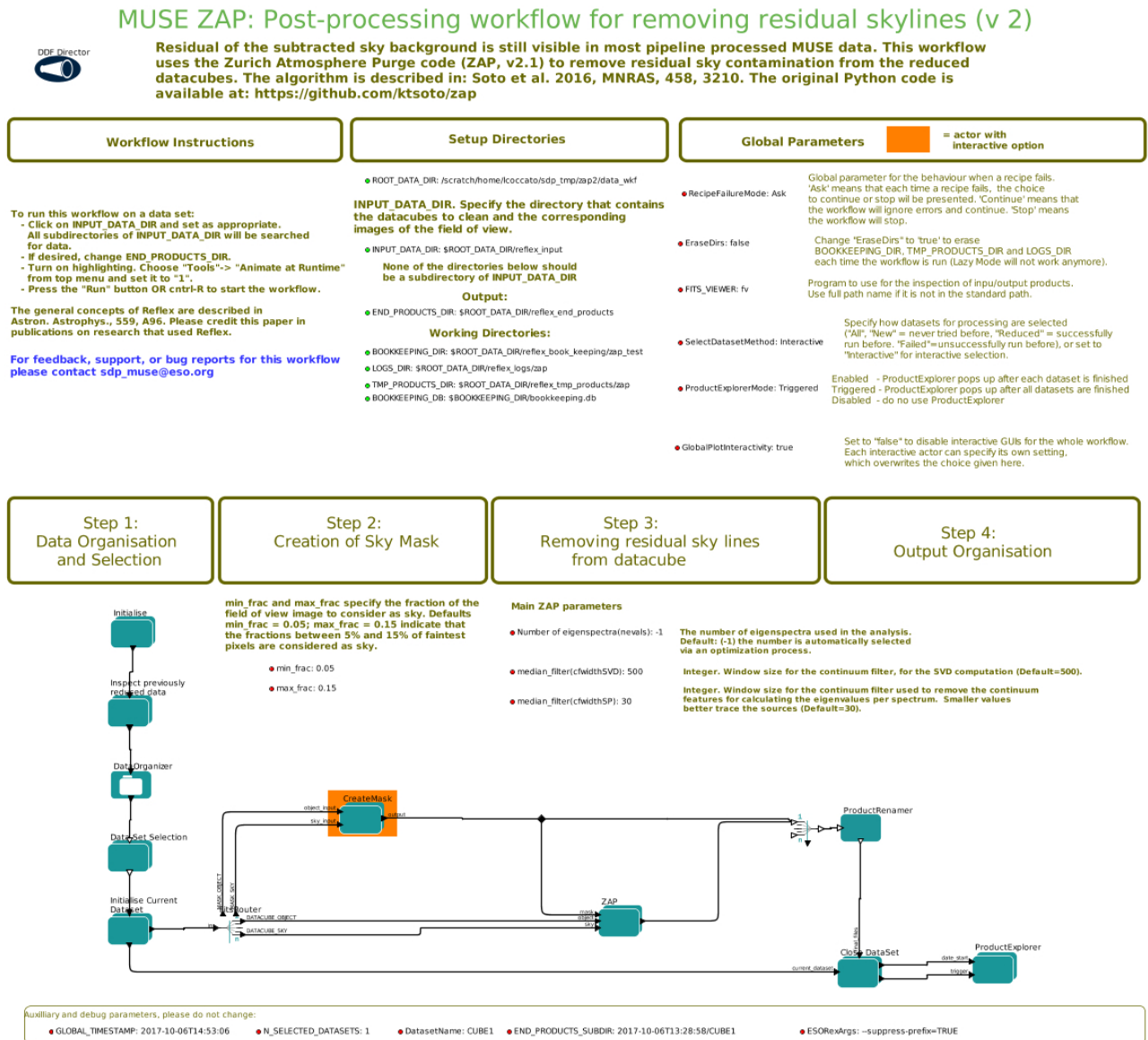


Figure 3.2: The muse_zap.xml Reflex workflow; it executes the Zurich Atmosphere Purge code (Soto et al. 2016) to remove residual sky contamination from the datacubes.

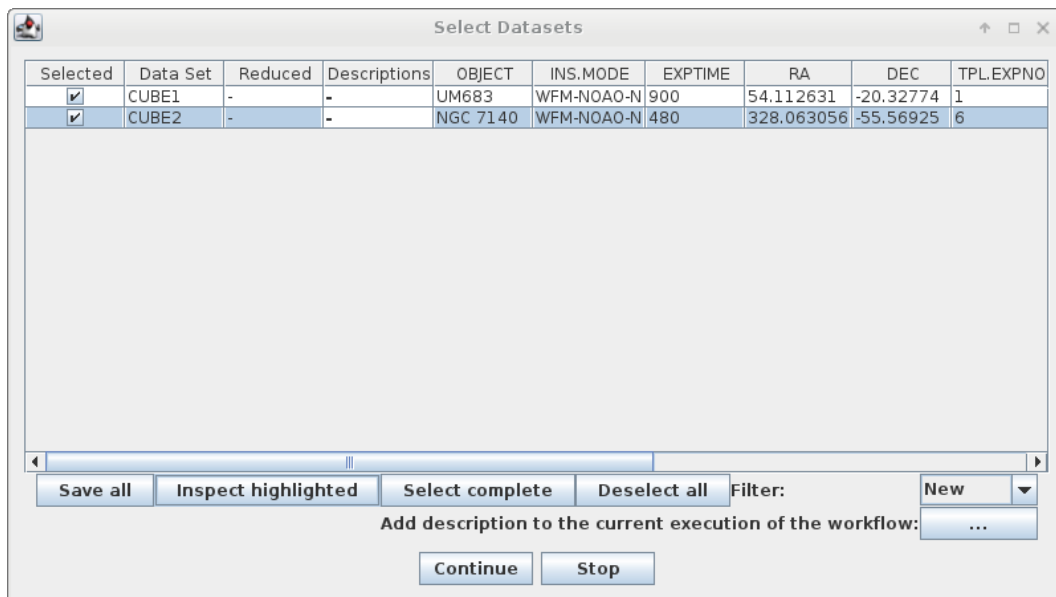


Figure 3.3: The Select Datasets window.

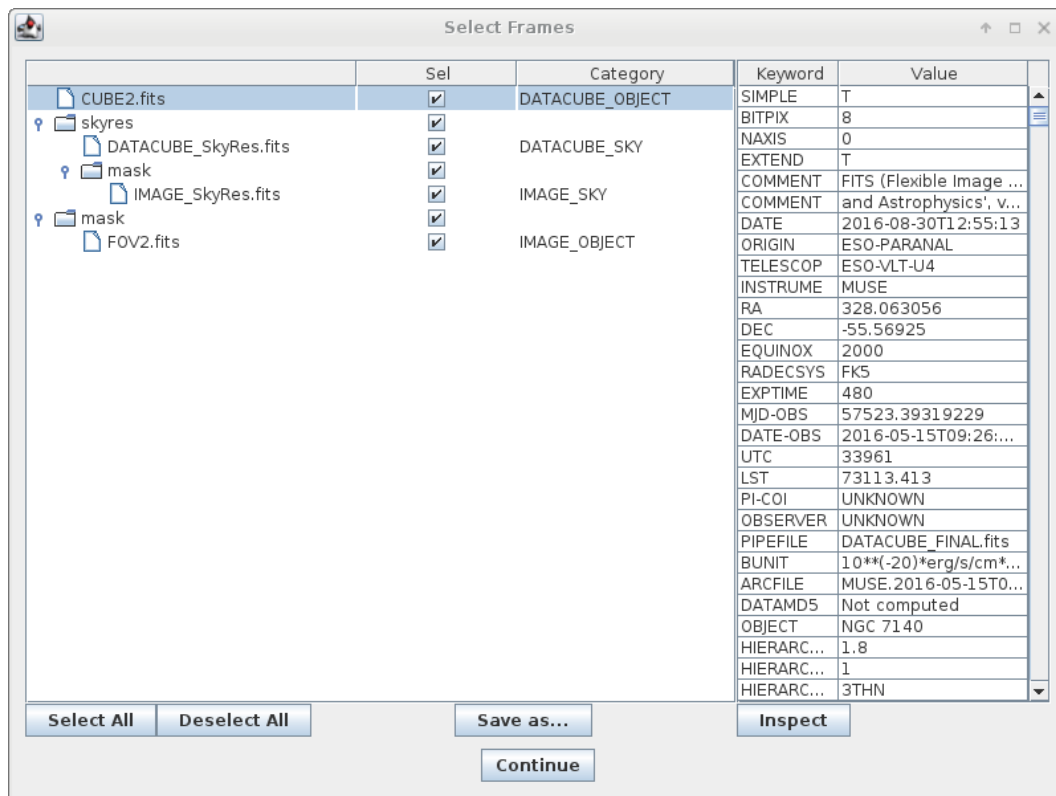


Figure 3.4: The Select Frames window (bottom).

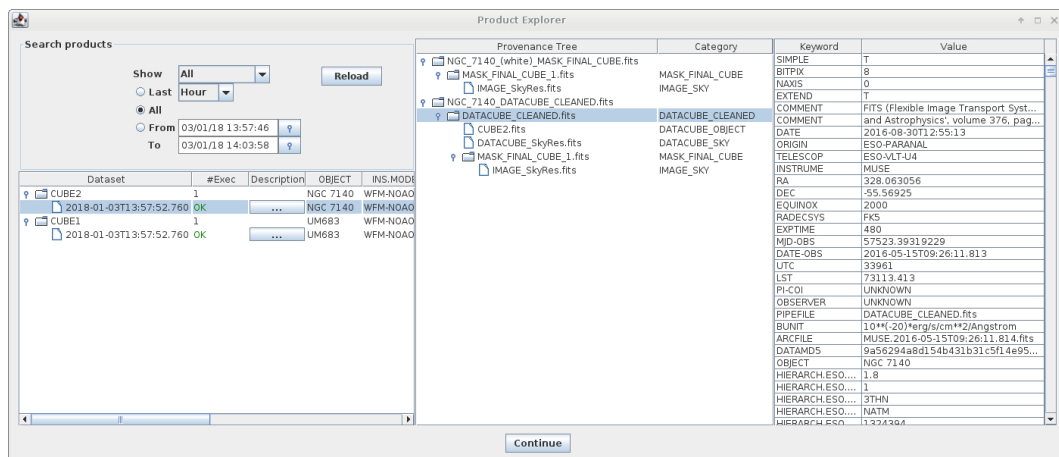


Figure 3.5: The Product Explorer window.










4 About the main `esoreflex` canvas

4.1 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 `esoreflex` 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 with another computer.








4.2 Buttons

At the top of the `esoreflex` canvas are a set of buttons which have the following 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.


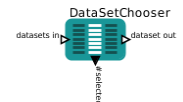
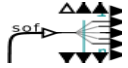


4.3 Workflow States

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.



4.3.1 Simple Actors

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

-  - The `DataOrganiser` actor.
-  - The `DataSetChooser` actor (inside a composite actor).
-  - The `FitsRouter` actor Redirects files according to their categories.
-  - The `ProductRenamer` actor.
-  - The `ProductExplorer` actor (inside a composite 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).

4.3.2 Lazy Mode

By default, all `RecipeExecutor` 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 re-reduction where it is not necessary.

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

The re-reduction of data at each execution may sometimes be desirable. To force a re-reduction of data for any single `RecipeExecutor` actor in the workflow, right-click the actor, select `Configure Actor`, and uncheck the Lazy mode parameter tick-box in the “Edit parameters” window that is displayed. For many workflows the `RecipeExecutor` actors are actually found inside the composite actors in the top level workflow. To access such embedded `RecipeExecutor` actors you will first need to open the sub-workflow by right-clicking on the composite actor and then selecting `Open Actor`.



To force the re-reduction of all data in a workflow (i.e. to disable Lazy mode for the whole workflow), you must uncheck the Lazy mode for every single `RecipeExecutor` actor in the entire workflow. It is also possible to change the name of the bookkeeping directory, instead of modifying any of the Lazy mode parameters. This will also force a re-reduction of the given dataset(s). A new reduction will start (with the lazy mode still enabled), but the results of previous reduction will not be reused. Alternatively, if there is no need to keep any of the previously reduced data, one can simply set the `EraseDirs` parameter under the “Global Parameters” area of the workflow canvas to `true`. This will then remove all previous results that are stored in the bookkeeping, temporary, and log directories before processing the input data, in effect, starting a new clean data reduction and re-processing every input dataset. *Note: The option `EraseDirs = true` does not work in esoreflex version 2.9.x and makes the workflow to crash.*



5 The MUSE ZAP Workflow

The MUSE ZAP 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.

5.1 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 `RAW_DATA_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 `RAW_DATA_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]).

There is a mode of the `Data Organiser` that skips the built-in data organisation and uses instead the data organisation provided by the `CalSelector` tool. To use this mode, click on `Use CalSelector associations` in the `Data Organiser` properties and make sure that the input data directory contains the XML file downloaded with the `CalSelector` archive request (note that this does not work for all instrument workflows).

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. Note that it is recommended to specify the full path to the visualization application (an alias will not work).

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 4.3.2), reusing the previous pipeline recipe outputs if 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 re-reduce 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 failures 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. Alternatively, the `Stop` mode will stop the workflow execution immediately.

The parameter `ProductExplorerMode` controls whether the `ProductExplorer` actor will show its window or not. The possible values are `Enabled`, `Triggered`, and `Disabled`. `Enabled` opens the Produc-



tExplorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

5.2 MUSE ZAP specific workflow parameters

The main workflow canvas contain some parameters that can regulate the data process with ZAP. They can be specified by the user from the main reflex canvas, or inside the ZAP composite actor before starting the data reduction. They are:

- **number of eigenspectra(nevals)**. It specifies the number of eigenspectra to use. The default value (-1) triggers an automatic and optimized calculation of the number of eigenspectra to use.
- **median_filter(cfwidthSVD)**. Window size for the continuum filter, for the SVD computation. See Section 2 of Soto et al. (2016). Default: 500.
- **median_filter(cfwidthSP)**. Window size for the continuum filter used to remove the continuum features for calculating the eigenvalues per spectrum. Smaller values better trace the sources. An optimal range of is typically 20 – 50 pixels. Default: 30.

For a proper description of the ZAP code and all its parameters, we refer the interested reader to the its documentation (<http://muse-vlt.eu/science/tools/>) and publication (Soto et al. MNRAS, 458, 3210).



6 Reducing your own data

In this section we describe how to reduce your own data set.

First, we suggest the reader to familiarize with the workflow by reducing the demo dataset first (Section 3), but it is not a requirement.

6.1 The `esoreflex` command

We list here some options associated to the `esoreflex` command. We recommend to try them to familiarize with the system. In the following, we assume the `esoreflex` executable is in your path; if not you have to provide the full path `<install_dir>/bin/esoreflex`

To see the available options of the `esoreflex` command type:

```
esoreflex -h
```

The output is the following.

```
-h | -help          print this help message and exit.
-v | -version       show installed Reflex version and pipelines and exit.
-l | -list-workflows list available installed workflows and from
                    ~/KeplerData/workflows.
-n | -non-interactive enable non-interactive features.
-e | -explore        run only the Product Explorer in this workflow
-p <workflow> | -list-parameters <workflow>
                    lists the available parameters for the given
                    workflow.
-config <file>       allows to specify a custom esoreflex.rc configuration
                    file.
-create-config <file> if <file> is TRUE then a new configuration file is
                    created in ~/.esoreflex/esoreflex.rc. Alternatively
                    a configuration file name can be given to write to.
                    Any existing file is backed up to a file with a '.bak'
                    extension, or '.bakN' where N is an integer.
-debug              prints the environment and actual Reflex launch
                    command used.
```

6.2 Launching the workflow

We list here the recommended way to reduce your own datasets. Steps 1 and 2 are optional and one can start from step 3.

1. Type: `esoreflex -n <parameters> muse_zap` to launch the workflow non interactively and reduce all the datasets with default parameters.



<parameters> allows you to specify the workflow parameters, such as the location of your raw data and the final destination of the products.

For example, type (in a single command line):

```
esoreflex -n
  -INPUT_DATA_DIR /home/user/my_input_data
  -ROOT_DATA_DIR /home/user/my_reduction
  -END_PRODUCTS_DIR $ROOT_DATA_DIR/reflex_end_products
muse_zap
```

to reduce the complete datasets that are present in the directory `/home/user/my_input_data` and that were not reduced before. Final products will be saved in `/home/user/my_reduction/reflex_end_products`, while book keeping, temporary products, and logs will be saved in sub-directories of `/home/user/my_reduction/`. If the reduction of a dataset fails, the reduction continues to the next dataset. It can take some time, depending on the number of datasets present in the input directory. For a full list of workflow parameters type `esoreflex -pmuse_zap`. Note that this command lists only the parameters, but does not launch the workflow.

Once the reduction is completed, one can proceed with optimizing the results with the next steps.

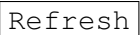
2. Type:

```
esoreflex -e muse_zap
```

to launch the Product Explorer. The Product Explorer allows you to inspect the data products already reduced by the `MUSE_ZAP esoreflex` workflow. Only products associated with the workflow default bookkeeping database are shown. To visualize products associated to given bookkeeping database, pass the full path via the `BOOKKEEPING_DB` parameter:

```
esoreflex -e BOOKKEEPING_DB <database_path> muse_zap
```


to point the product explorer to a given `<database_path>`, e.g., `/home/username/reflex/reflex_bookkeeping/test.db`

The Product Explorer allows you to inspect the products while the reduction is running. Press the button  to update the content of the Product Explorer. This step can be launched in parallel to step 1.

A full description of the Product Explorer will be given in Section [6.4.1](#)

3. Type:


```
esoreflex muse_zap &
```

to launch the `MUSE_ZAP esoreflex` workflow. The `MUSE_ZAP` workflow window will appear (Fig. [3.2](#)). Please configure the set-up directories `ROOT_DATA_DIR`, `INPUT_DATA_DIR`, and other workflow parameters as needed. Just double-click on them, edit the content, and press . Remember to specify the same `<database_path>` as for the Product Explorer, if it has been opened at step #2, to synchronize the two processes.

4. (Recommended, but not mandatory) On the main `esoreflex` menu set `Tools -> Animate at Runtime` to 1 in order to highlight in red active actors during execution.



5. Set the workflow parameter to specify the reduction strategy. See Section 5.2 for details.

Press the button  to start the workflow. First, the workflow will highlight and execute the `Initialise` actor, which among other things will clear any previous reductions if required by the user (see Section 5.1). Secondly, if set, the workflow will open the Product Explorer, allowing the user to inspect previously reduced datasets (see Section 6.4.1 for how to configure this option).

6.3 Workflow Steps

The first thing to know for understanding how the `muze_zap` workflow works and the adopted strategy, is understand how the datasets are generated.

The creation of datasets is done by the actor `DatasetOrganizer`, which follows a series of rules, as in all other workflows. The workflow handles these kind of files, which are identified by the category written in the `PRO REC1 RAW1 CATG` header keyword:

- **Datacubes.** Any category that contains the string `DATA_CUBE` is considered a datacube. These files could be either the datacubes of the objects we have to clean (classified as `DATA_CUBE_OBJECT` by the workflow), or the datacube of a residual sky background (classified as `DATA_CUBE_SKY` by the workflow). This distinction is done on the basis of the `PRO REC1 RAW1 CATG` header keyword, which can be either `OBJECT` or `SKY`. A `DATA_CUBE_SKY` is associated to a `DATA_CUBE_OBJECT` if they have the same value of the `OBJECT` header keyword.
- **Images.** Any category that contains the string `IMAGE_FOV` is considered an image, which will be used to create a sky mask. The images can be associated to the object (in this case they will be classified as `IMAGE_OBJECT` by the workflow) or to a sky residual cube (in this case they will be classified as `IMAGE_SKY` by the workflow). As for the datacubes, this distinction is done on the basis of the `PRO REC1 RAW1 CATG` header keyword, which can be either `OBJECT` or `SKY`. An image is associated to a datacube if they have the same value of the `MJD-OBS` header keyword.
- **Sky masks.** Sky masks have the category `MASK_FINAL_CUBE`. They can be classified as `MASK_OBJECT` or `MASK_SKY` depending on the value of the `PRO REC1 RAW1 CATG` header keyword, as for the previous cases. A sky mask is associated to a cube if they have the same `MJD-OBS` header keyword.

Each dataset contains a `DATA_CUBE_OBJECT` and an `IMAGE_OBJECT`. Eventually, they can have a `MASK_OBJECT` assigned to it. The user can decide whether to use the provided mask or create a new one from the image. These datasets reflect the first analysis strategy, where the principal components are evaluated on the object datacube and then fitted.

A `DATA_CUBE_SKY` with the corresponding `IMAGE_SKY` (and eventually a `MASK_SKY`) can be associated to the dataset, if they target the same object. These datasets reflect the second analysis strategy, where the principal components are evaluated on the sky cube and then fitted to the object datacube.



6.3.1 Data Organisation And Selection

The `DataOrganiser` (DO) is the first crucial component of a Reflex workflow. The DO takes as input `RAW_DATA_DIR` and `CALIB_DATA_DIR` and it detects, classifies, and organises the files in these directories and any subdirectories. The output of the DO is a list of “DataSets”. A DataSet is a special Set of Files (SoF). A DataSet contains one or several science (or calibration) files that should be processed together, and all files needed to process these data. This includes any calibration files, and in turn files that are needed to process these calibrations. Note that different DataSets might overlap, i.e. some files might be included in more than one DataSet (e.g., common calibration files).

A DataSet lists three different pieces of information for each of its files, namely 1) the file name (including the path), 2) the file category, and 3) a string that is called the “purpose” of the file. The DO uses the OCA² rules to find the files to include in a DataSet, as well as their categories and purposes. The file category identifies different types of files, and it is derived by information in the header of the file itself. A category could for example be `RAW_CALIBRATION_1`, `RAW_CALIBRATION_2` or `RAW_SCIENCE`, depending on the instrument. The purpose of a file identifies the reason why a file is included in a DataSet. The syntax is `action_1/action_2/action_3/ ... /action_n`, where each `action_i` describes an intended processing step for this file (for example, creation of a `MASTER_CALIBRATION_1` or a `MASTER_CALIBRATION_2`). The actions are defined in the OCA rules and contain the recipe together with all file categories required to execute it (and predicted products in case of calibration data). For example, a workflow might include two actions `action_1` and `action_2`. The former creates `MASTER_CALIBRATION_1` from `RAW_CALIBRATION_1`, and the later creates a `MASTER_CALIBRATION_2` from `RAW_CALIBRATION_2`. The `action_2` action needs `RAW_CALIBRATION_2` frames and the `MASTER_CALIBRATION_1` as input. In this case, these `RAW_CALIBRATION_1` files will have the purpose `action_1/action_2`. The same DataSet might also include `RAW_CALIBRATION_1` with a different purpose; irrespective of their purpose the file category for all these biases will be `RAW_CALIBRATION_1`.

The Datasets created via the `DataOrganiser` will be displayed in the `DataSet Chooser`. Here the users have the possibility to inspect the various datasets and decide which one to reduce. By default, DataSets that have not been reduced before are highlighted for reduction. Click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow. A full description of the `DataSet Chooser` is presented in Section 6.3.2.

Once the `Continue` is pressed, the workflow starts to reduce the first selected DataSet. Files are broadcasted according to their purpose to the relevant actors for processing.

The categories and purposes of raw files are set by the DO, whereas the categories and purpose of products generated by recipes are set by the `RecipeExecutor`. The file categories are used by the `FitsRouter` to send files to particular processing steps or branches of the workflow (see below). The purpose is used by the `SofSplitter` and `SofAccumulator` to generate input SoFs for the `RecipeExecutor`. The `SofSplitter` and `SofAccumulator` accept several SoFs as simultaneous input. The `SofAccumulator`

²OCA stands for OrganisationClassificationAssociation and refers to rules, which allow to classify the raw data according to the contents of the header keywords, organise 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/<pipeline-version>/reflex/`, carrying the extension `.oca`. The variable `<install_dir>` depends on the operative system and installation procedure. For installation through rpm: `<install_dir>=/usr`; for installation through macport `<install_dir>=/opt/local`; for installation through the installation script `install_esoreflex` it depends on the path specified during installation, e.g. `<install_dir>=<specified_path>/install`



creates a single output SoF from the inputs, whereas the `SofSplitter` creates a separate output SoF for each purpose.

6.3.2 DataSetChooser

The `DataSetChooser` displays the DataSets available in the “Select Data Sets” window, activating vertical and horizontal scroll bars if necessary (Fig. 3.3).

Some properties of the DataSets are displayed: the name, the number of files, a flag indicating if it has been successfully reduced (a green OK), if the reduction attempts have failed or were aborted (a red FAILED), or if it is a new dataset (a black "-"). The column "Descriptions" lists user-provided descriptions (see below), other columns indicate the instrument set-up and a link to the night log.

Sometimes you will want to reduce a subset of these DataSets rather than all DataSets, and for this you may individually select (or de-select) DataSets for processing using the tick boxes in the first column, and the buttons `Deselect All` and `Select Complete` at the bottom, or configure the “Filter” field at the bottom left. Available filter options are: "New" (datasets not previously reduced will be selected), "Reduced" (datasets previously reduced will be selected), "All" (all datasets will be selected), and "Failed" (dataset with a failed or aborted reduction will be selected).

You may also highlight a single DataSet in blue by clicking on the relevant line. If you subsequently click on `Inspect Highlighted`, then a “Select Frames” window will appear that lists the set of files that make up the highlighted DataSet including the full filename³, the file category (derived from the FITS header), and a selection tick box in the right column. The tick boxes allow you to edit the set of files in the DataSet which is useful if it is known that a certain calibration frame is of poor quality (e.g: a poor raw flat-field frame). The list of files in the DataSet may also be saved to disk as an ASCII file by clicking on `Save As` and using the file browser that appears.

By clicking on the line corresponding to a particular file in the “Select Frames” window, the file will be highlighted in blue, and the file FITS header will be displayed in the text box on the right, allowing a quick inspection of useful header keywords. If you then click on `Inspect`, the workflow will open the file in the selected FITS viewer application defined by the workflow parameter `FITS_VIEWER`.

To exit from the “Select Frames” window, click `Continue`.

To add a description of the reduction, press the button `...` associated with the field "Add description to the current execution of the workflow" at the bottom right of the Select Dataset Window; a pop up window will appear. Enter the desired description (e.g. "My first reduction attempt") and then press `OK`. In this way, all the datasets reduced in this execution, will be flagged with the input description. Description flags can be visualized in the `SelectFrames` window and in the `ProductExplorer`, and they can be used to identify different reduction strategies.



To exit from the “Select DataSets” window, click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow.

³keep the mouse pointer on the file name to visualize the full path name.



6.4 The processing cascade

The next steps are executed by the two main main composite actors in the `muse_zap` workflow:

-  **CreateMask**. This actor uses the `IMAGE_OBJECT` (or `IMAGE_SKY`, if they are present in the dataset) to identify the regions in the field of view that are dominated by the sky. Those regions will be used to evaluate the components that describe the sky residuals. It also allows to edit a user-provided mask if present in the dataset. It is an interactive actor, and it will be described in Section 7.1
-  **ZAP**. Removal of Sky residual from object cube (ZAP). It runs the ZAP code to remove residual sky contamination from the object cube. The principal components will be created on the sky-residuals cube (if present in the dataset) or directly on the object cube, using an appropriate mask defined in `CreateMask`.


6.4.1 The ProductExplorer

The ProductExplorer is an interactive component in the `esoreflex` workflow whose main purpose is to list the final products with the associated reduction tree for each dataset and for each reduction attempt (see Fig. 3.5).

Configuring the ProductExplorer

You can configure the ProductExplorer GUI to appear after or before the data reduction. In the latter case you can inspect products as reduction goes on.

1. To display the ProductExplorer GUI at the end of the data reduction:


- Click on the global parameter "ProductExplorerMode" before starting the data reduction. A configuration window will appear allowing you to set the execution mode of the Product Explorer. Valid options are:
 - "Triggered" (default). This option opens the ProductExplorer GUI when all the selected datasets have been reduced.
 - "Enabled". This option opens the ProductExplorer GUI at the end of the reduction of each individual dataset.
 - "Disable". This option does not display the ProductExplorer GUI.
- Press the  button to start the workflow.

2. To display the ProductExplorer GUI "before" starting the data reduction:

- double click on the composite Actor "Inspect previously reduced data". A configuration window will appear. Set to "Yes" the field "Inspect previously reduced data (Yes/No)". Modify the field "Continue reduction



after having inspected the previously reduced data? (Continue/Stop/Ask)". "Continue" will continue the workflow and trigger the DataOrganizer. "Stop" will stop the workflow; "Ask" will prompt another window deferring the decision whether continuing or not the reduction after having closed the Product Explorer.

- Press the  button to start the workflow. Now the ProductExplorer GUI will appear before starting the data organization and reduction.

Exploring the data reduction products

The left window of the ProductExplorer GUI shows the executions for all the datasets (see Fig. 3.5). Once you click on a dataset, you get the list of reduction attempts. Green and red flags identify successful or unsuccessful reductions. Each reduction is linked to the "Description" tag assigned in the "Select Dataset" window.

1. To identify the desired reduction run via the "Description" tag, proceed as follows:

- Click on the symbol at the left of the dataset name. The full list of reduction attempts for that dataset will be listed. The column Exec indicates if the reduction was successful (green flag: "OK") or not (red flag: "Failed").
- Click on the entries in the field "Description" to visualize the description you have entered associated to that dataset on the Select Dataset window when reducing the data.
- Identify the desired reduction run. All the products are listed in the central window, and they are organized following the data reduction cascade.

You can narrow down the range of datasets to search by configuring the field "Show" at the top-left side of the ProductExplorer (options are: "All", "Successful", "Unsuccessful"), and specifying the time range (Last, all, From-to).

2. To inspect the desired file, proceed as follows:

- Navigate through the data reduction cascade in the ProductExplorer by clicking on the files.
- Select the file to be inspected and click with the mouse right-hand button. The available options are:
 - Options available always:
 - * Copy full path. It copies the full name of the file onto the clipboard. Shift+Ctrl+v to paste it into a terminal.
 - * Inspect Generic. It opens the file with the fits viewer selected in the main workflow canvas.
 - * Inspect with. It opens the file with an executable that can be specified (you have to provide the full path to the executable).
 - Options available for files in the `TMP_PRODUCTS_DIR` directory only:
 - * command line. Copy of the environment configuration and recipe call used to generate that file.
 - * Xterm. It opens an Xterm at the directory containing the file.



- Options available for products associated to interactive windows only:
 - * Display pipeline results. It opens the interactive windows associated to the recipe call that generated the file. Note that this is for visualization purposes only; the recipe parameters cannot be changed and the recipe cannot be re-run from this window.



7 Removing the residuals of the sky background

In this Section we describe the three main steps performed by the `muse_zap` workflow aimed at removing the residual contamination of the sky background from a datacube.

7.1 Creation of Sky Mask

The `CreateMask` actor creates a sky mask, which is a 2D image file that covers the same field of view as the datacube and with the same spatial resolution. Its values are 0 (for sky pixel, on which the residuals need to be analyzed) and 1 (for object pixel, not to be used for the sky residual analysis). The category of the mask fits file is `MASK_FINAL_CUBE`, and it is stored in the header. The mask has the same `MJD-OBS` as the associated datacube.

In the case you are evaluating the principal components directly on the object cube, it is recommended to define the sky regions on areas surrounding the object you are interested in, making sure not to include bright sources.

7.1.1 Description of the interactive window

An interactive window will be displayed (Figure 7.1), allowing the user to inspect the current sky mask (either created from the `IMAGE_FOV` or provided in the `INPUT_DATA_DIR`) and, eventually, to change parameters to change the mask definition.

The `CreateMask` interactive window is divided into 2 parts. On the left side, there is the plotting area with two panels; on the right side there is the list of recipe parameters.

Both panels in the plotting area display the image of the field of view (`IMAGE_FOV`) and the current sky mask (`MASK_FINAL_CUBE`) with a reversed color scheme.

In the top panel (named “Sky”) the region dominated by the sky are in gray-scale, whereas the regions dominated by objects are in red. With this window, the user has the possibility to check whether there are bright objects inside the empty sky regions. If so, the sky regions need to be redefined not to include bright objects.

In the bottom panel (named “Object”) the region dominated by the sky are in red, whereas the regions dominated by objects are in gray-scale. With this window, the user has the chance to inspect the coverage of the sky region, and decide whether there are other areas around the object, free of bright sources, that can be included in the mask.

The red-colored pixels in the “Object” panel corresponds to the pixels that have value 0 in the created mask and identify empty sky regions.

If present, the user supplied mask is used by default. Otherwise, the one created with default parameters is used.



7.1.2 Description of the parameters

The mask can be create either from the image of the field of view (`IMAGE_OBJECT` or `IMAGE_SKY`) or from an user-provided mask (`MASK_OBJECT` or `MASK_SKY`, optional) present in the dataset.

The creation of a sky mask is regulated by a series of parameters that can be modified by the user through the interactive window. When a parameter has been modified, press the button “Re-run Recipe” to calculate and display the new mask. If you are happy with the results, press the button “Continue Wkf”.

If a mask is provided with the dataset, it will be displayed in the interactive window for inspection. The user has the possibility to modify it, use it as it is, or discard it and create a new one from the `IMAGE_OBJECT/SKY`.⁴

These 3 choices are determined by the value of the parameter **edit input mask?** in the interactive window (Figure 7.1). The editing of the user-provided mask is regulated by the parameters **n_grow** and **skybox**.

If a mask is not provided with the dataset, the actor generates a new one from the `IMAGE_OBJECT/SKY` (using default parameters) and displays it. The user has the possibility to modify it by changing the parameters **max_frac**, **min_frac**, **n_grow**, and **skybox**.

The parameters that regulate the creation of the sky mask are the following:

- Tab **Mask**. Note: this Tab and parameters are available only if a user-provided mask is present in the dataset.
 - **edit input mask?**. It determines whether a user-provided mask has to be edited or not. The allowed values are the following:
 - * *use*. The provided mask is the one displayed in the left panels. The recipes creates a copy of it and uses it.
 - * *add*. The provided mask will be edited using **n_grow** and **skybox** to add regions to it.
 - * *discard*. The input mask will not be used and a new one will be created using the `IMAGE_FOV` and all the recipe parameters.

Values need to be entered without quotation marks. Default = *use*.

- Tab **Flux Ranges**
 - **min_frac** and **max_frac**. The mask is created by rejecting the fraction **min_frac** of faintest spaxels in the image, and retain the fraction **max_frac** of faintest spaxels for sky evaluation. Defaults are **min_frac**=0.05 and **max_frac** = 0.15. This means that only the 5%-15% faintest pixels in `IMAGE_OBJECT/SKY` are used to define the sky regions in the sky mask. These parameters have an effect only if the mask is created from the `IMAGE_OBJECT/SKY`. The default values can also be changed by double clicking on the corresponding parameter in the workflow main canvas before starting the workflow. *Warning:* These parameters must follow the rule $0 \leq \text{min_frac} \leq \text{max_frac}$.
 - **n_grow**. It enlarges the existing map spaxels containing sky by the specified number of pixels. It does not modify regions defined with the **skybox** parameter. It **n_grow** is useful to enlarge isolated

⁴In most of the cases, the `IMAGE_OBJECT` or `IMAGE_SKY` associated with the cube to clean are so-called `WHITE LAMP IMAGES`, btained by averaging the datacube along wavelenghts. For targets where a large and diffuse emission is expected, it might be more convenient to create an image, collapsing the cube only on the wavelenghts of interest.

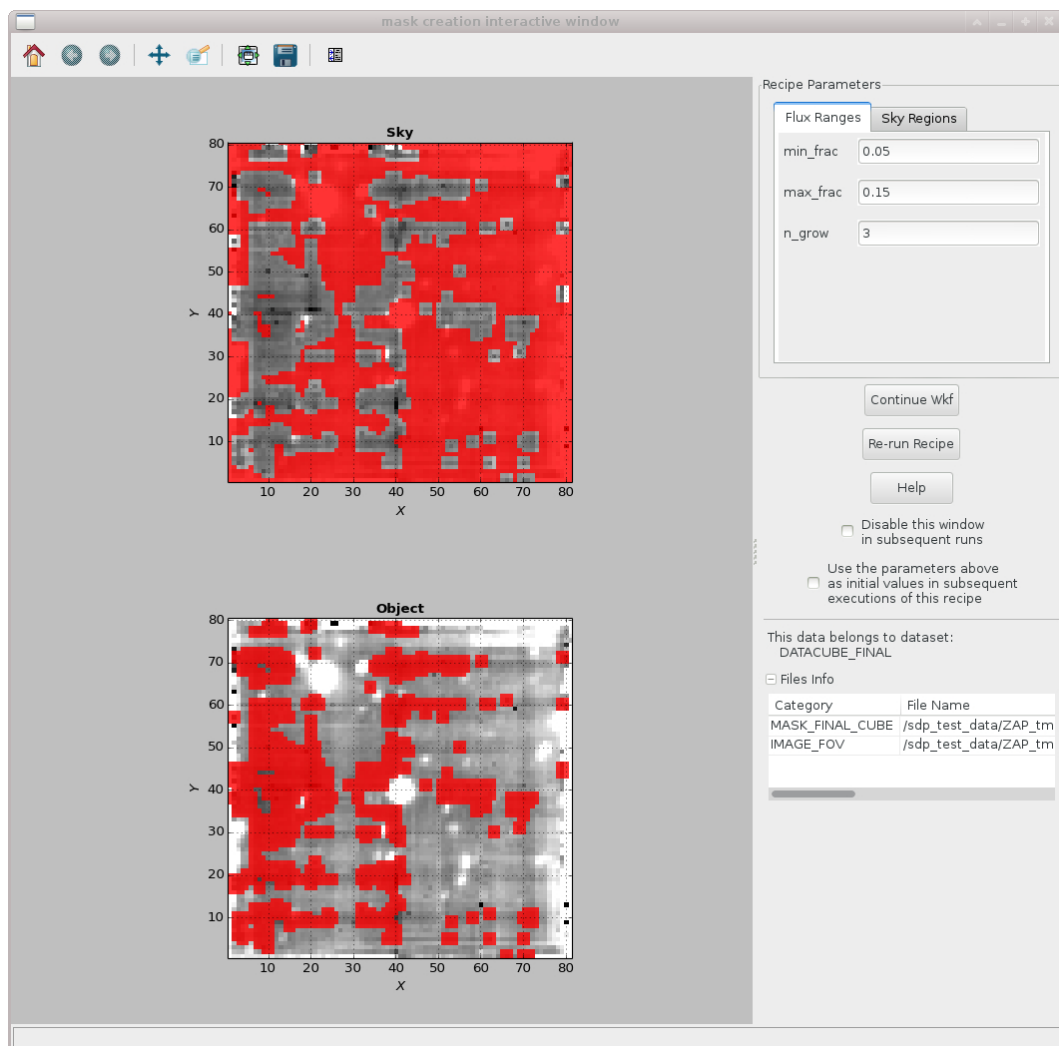


Figure 7.1: The interactive window associated to the CreateMask actor.



sky regions. Large values might increase the risk to include bright objects among the sky regions. Default= 3.

- Tab **Sky Regions**

- **skybox** N Up to $N = 10$ sky regions that can be added to the existing mask to include areas with only sky. Each region will be specified by 4 numbers indicating the xmin, xmax, ymin, and ymax pixel position in the field of view. An entry of -1 -1 -1 -1 will not add a sky region to the existing mask. Values needs to be entered without quotation marks and need be space separated. *Warning*. **n_grow** has no impact on the size of skyboxes. Default: -1 -1 -1 -1.

Tip: The suggested way to generate a custom sky mask is first to use the workflow to create one from the `IMAGE_OBJECT/SKY`. The mask will be saved into the reflex end products directory with the string `MASK_FINAL_CUBE` in its name. The user can either edit it (e.g., with the `imedit` task in `IRAF`) or copy it directly into the `INPUT_DATA_DIR` and then re-run the workflow to have the possibility to further modify it. It is important to remember that, when editing a mask via external tools, to maintain the same file structure, header content, spatial coverage and sampling as the original mask.

Tip: If you want to design your mask only using skyboxes, without any flux-based spaxel selection, you have to specify **min_frac=max_frac=0** and at least one **skybox**. It is not allowed to set **min_frac=max_frac=0** without specifying any skybox.

7.2 Removing sky residuals via ZAP

The `ZAP` actor executes the Zurich Atmosphere Purge (Soto et al. 2016) version 2.1 on the selected datacube to remove the residual sky contamination. The process is divided into two parts, that are both executed within the `ZAP` actor.

- **Calculation of the sky principal components (SVD).**

The basic idea is to describe the sky residuals (onto areas that contain only sky) in terms of eigenspectra via Single Value Decomposition. The components specified by the SVD are then fitted on the spectra containing the object in order to isolate the contribution of the sky residuals and subtract it.

The computation of the principal components (or, eigenspectra) is performed either on the object datacube or on the sky datacube if it is present in the dataset .

In the process, only the portion of the sky selected by the input mask (created in the actor `CreateMask`) will be used.

- **Fit the sky principal components to the object.**

In this step, the principal components computed at the previous step are fitted to each spaxel in datacube to evaluate and remove the residual sky contamination. As explained in the previous section, the principal components can be computed either on the object datacube or on the sky datacube if present in the dataset.

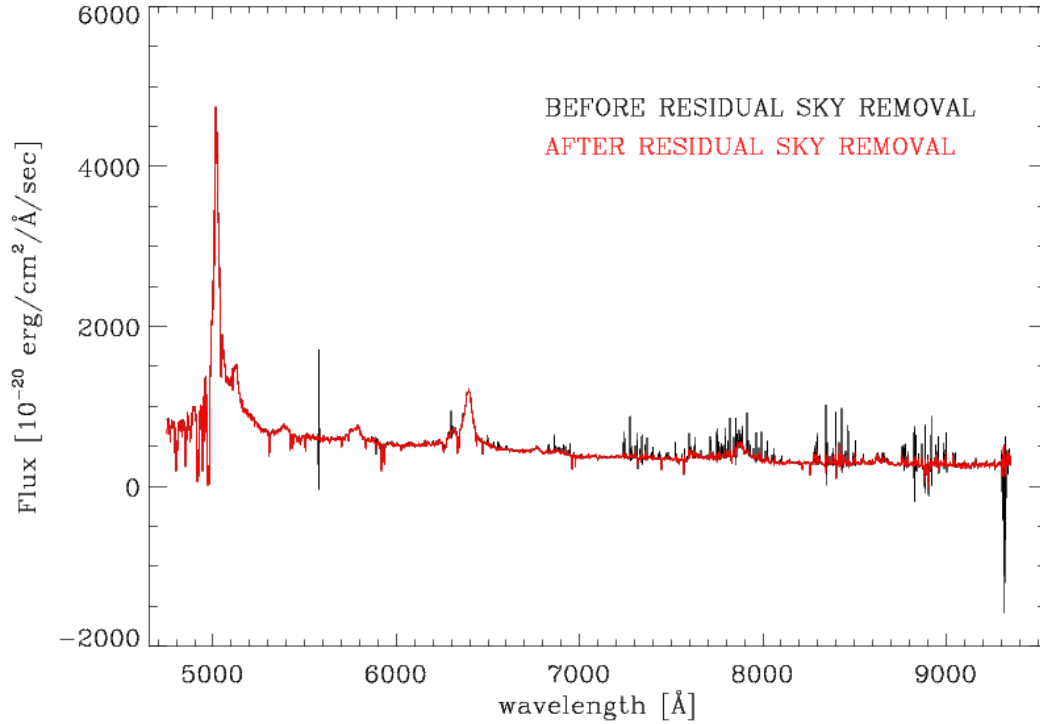


Figure 7.2: spectrum the central regions of the demo datacube (radius 5 pixels) before (black) and after (red) the post processing with ZAP. The used mask was obtained with **n_grow=1 min_max=0.05, max_frac=0.15, skybox1=10 80 10 20**, and **skybox2=60 70 10 80**, respectively.

Tip: in some cases, the blue part of the spectrum (below 6500 angstrom), which is not affected by many residual sky emission lines, is deteriorated by the zap cleaning. One can check this by measuring the signal-to-noise ratio before and after the ZAP cleaning on empty regions of the sky, or where the continuum of the source is low. If indeed, the S/N gets deteriorated, one can construct the final cube by taking the part below 6500 Å from the uncleaned cube, and the part above 6500 Å from the cleaned cube. The workflow does not support this procedure, that has to be conducted by a self-made script.

The user can use any datacube viewer (e.g., QFitsView; www.mpe.mpg.de/~ott/QFitsView/) to inspect the results.

Figure 7.2 shows the spectrum the central regions of the first demo datacube (radius 5 pixels) before and after the post processing with ZAP.

Warning: The user will be notified by a pop-up window in the case the code crashes for memory issues. See also Section 6.



8 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 could not yet be fixed.

- **I tried to `Open` (or `Configure`) an `Actor` while the workflow is running and now it does not react any more. What should I do?**

This is a limitation of the underlying Kepler engine. The only way out is to kill the workflow externally. If you want to change anything while a workflow is running you first need to pause it.

- **After a successful reduction of a data set, I changed this data set in some way (e.g. modified or removed some files, or changed the rules of the Data Organizer). When I restart Reflex, the Data Set Chooser correctly displays my new data set, but marks it as “reduced ok”, even though it was never reduced before. What does this mean?**

The labels in the column “Reduced” of the Data Set Chooser mark each dataset with “OK”, “Failed” or “-”. These labels indicate whether a data set has previously successfully been reduced at least once, all previous reductions failed, or a reduction has never been tried respectively. Data sets are identified by their name, which is derived from the first science file within the data set. As long as the data set name is preserved (i.e. the first science file in a data set has not changed), the Data Organizer will consider it to be the same data set. The Data Organizer recognizes any previous reductions of data sets it considers to be the same as the current one, and labels the current data set with “OK” if any of them was successful, even if the previously reduced data set differs from the current one.

Note that the Product Explorer will list all the previous reductions of a particular data set only at the end of the reduction. This list might include successful and/or unsuccessful reduction runs with different parameters, or in your case with different input files. The important fact is that these are all reductions of data sets with the same first raw science file. By browsing through all reductions of a particular raw science file, the users can choose the one they want to use.

- **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 for Kepler and the workflows to avoid opening any GUI elements (including pipeline interactive windows).



It is possible to specify workflow variables (those that appear in the workflow canvas) in the command line. For instance, the raw data directory can be set 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 executor. This actor must be manually instantiated using `Tools -> Instantiate Component`. Fill in the “Class name” field with `org.eso.RecipeExecutor` 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 manually run 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 a 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 on the command line a recipe that used a specific 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 above, 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 does not 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 were made to the pipeline. However there are situations in which a previously created bookkeeping directory will cause problems due to pipeline versions incompatibility. This is especially true if the parameters of the pipeline recipes have changed. In that case, please remove the bookkeeping directory completely.

- **How to insert negative values into a textbox?**

Due to a bug in wxPython, the GUI might appear to freeze when attempting to enter a negative number in a parameter's value textbox. This can be worked around by navigating away to a different control in the GUI with a mouse click, and then navigating back to the original textbox. Once focus is back on the original textbox the contents should be selected and it should be possible to replace it with a valid value, by typing it in and pressing the enter key.

- **I've updated my Reflex installation and when I run esoreflex the process aborts. How can I fix this problem?**

As indicated in Section 2, in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the `$HOME/KeplerData`, `$HOME/.kepler` directories if present, to prevent possible aborts (i.e. a hard crash) of the esoreflex process.

- **How can include my analysis scripts and algorithms into the workflow?**

EsoReflex is capable of executing any user-provided script, if properly interfaced. The most convenient way to do it is through the Python actor. Please consult the tutorial on how to insert Python scripts into a workflow available here: www.eso.org/sci/data-processing/Python_and_esoreflex.pdf



EsoReflex ZAP tutorial

Doc. Number: ESO-287180
Doc. Version: 6.0
Released on: 2023-05-01
Page: 40 of 40

- [1] Forchì V. *Reflex User's Manual*. ESO/SDD/DFS, <http://www.eso.org/gasgano/>, 0.7 edition, 2012. VLT-MAN-ESO-19000-5037. 22, 37, 38