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

FORS2 Imaging Cookbook and Reflex Tutorial

VLT-MAN-ESO-19500-....

Issue 2.0

Date 12th March 2025

Prepared:	H.M.J. Boffin, S. Moehler, C.E. García Dabo	12th March 2025	
	Name	Date	Signature
Approved:	W. Freudling		
	Name	Date	Signature
Released:	M. Sterzik		
	Name	Date	Signature

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	June 2017	All	First public release
1.2	October 2017	All	Changes for release 5.3.32
1.3	April 2019	All	Changes for release 5.4.3
1.5	June 2020	All	Changes for release 5.5.4
1.6	May 2022	All	Changes for release 5.6.2
2.0	12/03/2025	All	Update for release 5.8.1

This page was intentionally left blank

Contents

1 Imaging with the FORS2 Instrument: an introduction

FORS is the visual and near-UV FOcal Reducer and low dispersion Spectrograph for the Very Large Telescope (VLT) of the European Southern Observatory (ESO). Two versions of FORS have been built, upgraded and moved to the Cassegrain foci of different telescopes in the past years. In April 2009, FORS1 was retired, leaving only FORS2 in operation. FORS2 covers the wavelength range from 330 nm to 1100 nm and provides by default an image scale of $0''.25$/pixel and a field-of-view of $6.8' \times 6.8'$. It is by default equipped with a red-optimized detector system with a very low level of fringes (mosaic of two 2k×4k MIT CCDs). A blue-optimised detector system (mosaic of 2 2k×4k EEV CCDs) can be used for Visitor Mode observation. The geometries of both detector systems are similar, with the optical axis falling $\sim 30''$ above the gap and offsets of a few arc-seconds between the two chips. In the past monolithic detectors were used (FORS1 until March 2007, FORS2 until March 2002), that provided a default pixel scale of $0''.2$/pixel.

FORS2 offers imaging with a wide range of broad- and narrow-band filters, and can be used with the Standard (SR) or High Resolution (HR) collimator. The HR collimator will provide a field of view of $4' \times 4'$ with a pixel scale of $0''.125$/pixel with the default a 2×2 binning.

This document explains how to reduce data taken in the imaging mode of `FORS1` and `FORS2` and is based in part on the FORS data reduction cookbook (O'Brien et al., 2007, VLT-MAN-ESO-13100-4030). Other modes are described in different cookbook and tutorial documents.

A presentation of FORS2 is given in *https://eso.org/sci/facilities/paranal/instruments/fors.html*, which also provides access to several manuals. We recommend to read the imaging-related parts of the section on "Calibrating and Reducing FORS2 Data" in the FORS User Manual as well as the imaging-related parts of the sections 8 ("Data Reduction"), 9 ("Pipeline Recipe Interfaces") and 10 ("Algorithms") of FORS Pipeline User Manual

Much work has been done to investigate the limiting absolute photometric accuracy of FORS observations (see the technical report VLT-TRE-ESO-13112-5429; FORS Zero Point Monitoring and Absolute Photometry[1]), with the conclusion that the current calibration plan should make it possible to reach absolute photometric accuracies of 1.4–1.8%.

---

[1]*https://eso.org/sci/facilities/paranal/instruments/fors/doc/VLT-TRE-ESO-13112-5429_iss2.pdf*

2 Data reduction of FORS2 images

The reduction of FORS2 images consists in essentially 5 steps:

1. Create a master bias

2. Create a master flat field

3. Reduce the science images (bias and flat field correction)

4. Improve the World Coordinate System using Gaia data

5. Perform an in-situ photometric calibration using Gaia data or static zeropoints and extinction coefficients

Images can then also be co-added if wished, to obtain a better signal-to-noise ratio. And finally, several images obtained in various filters can be combined if useful. This last step, however, are not supported by the FORS pipeline.

Reducing FORS2 science images requires the following frames:

- Science frames obtained during the night

- Calibration frames

  - Bias frames
  - Twilight sky frames
  - Static calibrations providing static zeropoints and extinction coefficients

The calibration frames can be obtained from the ESO Science Archive, using `CalSelector`.

Two notes of caution:

1. The VLT images come with keywords matching the pixel numbers of the images and the RA-DEC coordinates, the "world coordinate system" (WCS). The WCS takes into account the scale and the rotation of the instrument, but no further distortion. The accuracy of these WCS is typically 1-3″, i.e. suitable for identifying the field and possibly an object, but they are *NOT* accurate enough for any decent astrometric work. To achieve the latter, it is recommended to get rid of the WCS keywords, and to work with the pixel numbers until the final steps, when a real astrometric calibration can be performed.

2. If one has many frames, the average can be replaced by the median, with the advantage that the cosmic rays and the moving objects (asteroids and such) will be removed from the combined frame. Note, however, that the median is *not* a linear process, and that this can affect the photometry of the combined frame. This should be done only if the number of frames is large, and the median is then becoming equal to the mean of the valid values (ie. not those affected by a cosmic), leading to a better result than the average.

2.1 Bias frames

Bias frames are taken with an exposure time of 0 seconds and a closed shutter. They thus record only the signal that is added during the read-out of the CCD to avoid negative numbers. A sequence of 5 or 20 bias frames[2] are taken the day following the observations as part of the FORS calibration plan.

2.2 Flat field

The purpose of the flat-field is to remove the sensitivity variations across the detector. As these variations act as a noise source, the precision of the flatfield correction will have direct consequences on the photometric precision that can be achieved, and on the limiting magnitude of the observations. The variations are caused by (i) intrinsic sensitivity variations of the pixels (either caused by the substrate, or the coating), (ii) extrinsic variations, e.g. a dust grain or hair sitting on the chip, (iii) optical design, and (iv) dust or external bodies on the optics. The first 2 causes are extremely stable over time, with a time scale of years.The optical effect can vary with a time-scale of hours, e.g. because of flexures; fortunately, in FORS, these effects are both extremely small and stable. As the FORSes are fairly well sealed and not often opened, the dust on the optics is reasonably stable, with a timescale of weeks.

It was shown that FORS2 twilight flats generally do not vary over at least 3 weeks, and the FORS2 calibration plan therefore ensures that twilight flats should be taken with the same settings (readout mode, binning, filter) as the science within 14 days.

Various types of frames are used to generate a master flat field, depending on the needs:

- Twilight flat-fields: pointing the telescope at a field devoid of bright stars during twilight. Pros: the pupil illumination matches pretty well that of real observations. Cons: the brightness level changes rapidly (this is taken into account by the FORS twilight flatfield templates); their number is rather limited, so the S/N achievable is also limited. Also, stars can become visible toward the dark end of the twilight, and have to be dealt with. These are the best "simple" flatfields, which explains their popularity. This is what is used for FORS imaging, as part of the standard calibration plan.

- Night sky flat fields: long night exposures can have a sky background of a few thousands counts. Pros: perfect match of the science frames. Cons: low exposure level; for instruments affected by sky concentration, the night sky flats are very difficult to use (this is **not** the case of the FORSes). Works only on fairly empty fields (hint: load the frame setting the cuts at mean-3sigma, mean+10sigma; if more than 25% of the pixels appear white, either because of the number of objects, or because some very bright objects are in the field, then forget about using this frame as a flatfield). Using these flatfields is sometimes called "super-flatfielding". This can be done by users for their specific needs.

  For Night Sky flat fields to work, it is absolutely critical that the images are taken with offets applied to the telescope between each exposure. The size of the offset must be sufficiently large so that the halo of an object does not overlap from frame to frame. So, a safe minimum distance between exposures is 3-4× seeing. Also, it is better that the offsets be computed using a random number generator, in order to avoid that the various exposures of a series be aligned along the lines and/or the columns, as this would potentially create some systematic effects. Some observers prefer *not* to offset the telescope between

---

[2]The calibration plan was changed in Nov. 2014, and 20 bias frames are now taken per setting instead of 5.

exposures, leaving the object always on the same pixel. While there might be some very specific cases in which the benefits of that method are significant, doing so will absolutely prevent a proper flatfielding of the data using night sky flats. Moreover, because of the instrumental flexure and because of the seeing variations, the object of interest will *not* always illuminate the same set of pixels. As a good night sky flat field will not be possible, that method will lead to some uncontrolled systematics that will render any photometric variation suspicious as it will be difficult to discriminate it from a systematic detector effect. In summary, offset.

- Dome flat-fields: pointing the telescope at a fairly uniformly illuminated screen. Their pro: unlimited in numbers, therefore in S/N. Con: the illumination is usually an halogen lamp, whose color is very different from that of the sky (introducing some weird color effects in broad band filters), and the pupil illumination is only very approximately matching that of real observations, introducing some slopes, gradients and low spatial frequencies variations in the flatfield. There are no dome flat-fields for FORS2.

- Internal flat-fields: using an internal pupil screen. These are usually not appropriate for imaging flat-fields. In the case of FORS2 (and also FORS1) they show bright spots caused by reflection at the LADC.

2.3 Static Calibration Data

In addition to calibration frames taken regularly, the FORS pipeline also uses static calibration tables related to direct imaging. These are

1. Photometric table (`PHOT_TABLE`). This table lists parameters related to each standard filter in use in the FORS1 and FORS2 instruments. There is one such table for each CCD in the mosaic (blue E2V and red MIT CCDs). Currently available are:

    - `fors1_Marlene_phot.fits`
    - `fors1_Norma_phot.fits`
    - `fors1_TK_phot.fits`
    - `fors2_1453_phot.fits`
    - `fors2_1456_phot.fits`
    - `fors2_Marlene_phot.fits`
    - `fors2_Norma_phot.fits`

2. Photometric standard stars catalogue (`FLX_STD_IMG`). This table is a list of photometric standard stars parameters. Currently two catalogues are included in the static calibrations supplied with the pipeline: selected UBVRI photometric standard stars from Landolt, and all entries from the Stetson's photometric standard stars catalogue, respectively in the files `landolt_std_UBVRI.tfits` and `stetson_std_BVRI.fits`.

3. A table defining the illuminated part of the detector (`DETECTOR_ILLUMINATED_REGION`)

4. A table with zeropoints, extinction coefficients, and central wavelength for all filters (`STATIC_PHOT_-COEFF_TABLE`)

A description of these tables is given in the FORS Pipeline User Manual. We refer to this manual for detailed information on the recipes and on what parameters can be configured. These tables are provided together with the FORS pipeline and located in the directory <INSTALL-DIR>/share/esopipes/datastatic/fors-<version> where <INSTALL-DIR> is: a) the installation directory set up during the installation if done with the installation script (e.g. /home/user/reflex/install/), or b) /usr in the case of rpm installation, or c) /opt/local in the case of macports installation.

3   Running the FORS2 pipeline

There are three ways to run the ESO pipelines, in all cases the executed recipes are, however, the same. The differences are in the user interfaces.

1. Reflex is the **recommended** environment to reduce ESO data. It automatically organizes input files according to their category and runs the entire reduction chain at the push of a button. It supports break points in the reduction sequence in order to inspect and interact with intermediate and final products and rerun the corresponding step if necessary. A more detailed description on how to use Reflex to reduce FORS2 data is provided in sections 6 to 9.3.

2. Gasgano is a Java-based data file organizer developed and maintained by ESO. It can be used to manage and organize in a systematic way the astronomical data observed and produced by all VLT compliant telescopes. Gasgano offers functionalities for data viewing, grouping, sorting, classification, searching, and filtering of data. And, of course, Gasgano will run any requested CPL recipe on the selected data. Gasgano is automatically installed when installing the stand-alone FORS pipeline kit available from `http://www.eso.org/sci/software/pipelines/`, but not as part of the Reflex installation.

3. Esorex, a command-line utility for running pipeline recipes is also available (and is used also by Reflex to run the pipeline recipes). Esorex may be embedded by users into data reduction scripts for the automation of processing tasks. See `http://www.eso.org/sci/software/cpl/esorex.html` for more information.

The underlying algorithms and recipes are the same for a given instrument pipeline, irrespective if Reflex, Gasgano or Esorex is used.

The pipeline itself can be accessed from the web at `http://www.eso.org/sci/software/pipelines/`.

To reduce FORS imaging data, one uses typically 3 recipes, in succession:

1. `fors_bias`: used to create a bias master calibration frame from a set of raw bias frames. If the over/prescan regions are present, the overscan correction is applied and the regions trimmed from the result.

2. `fors_img_sky_flat`: used to create a master twilight calibration frame from a set of raw twilight sky exposures. In order to eliminate the contributions of field stars on the jittered sequence of flat fields, the frame combination method must be based on a rejection algorithm (rather than on a simple average). The overscan regions, if present, are trimmed from the result. Note that images are normalised to the same illumination level before applying the rejection algorithm.

3. `fors_img_basic_science`: used to reduce a direct imaging scientific exposure. The master bias calibration is subtracted. The debiased signal is then divided by the normalised sky flat field, and the overscan regions, if present, are trimmed from the result.

4. `fors_img_wcs`: used to detect and extract sources on the reduced image, which are then matched against a catalog (by default the Gaia astrometric catalog) to improve the World Coordinate System (WCS) solution. Also the background of the reduced image is fitted and subtracted. Finally (if the

DETECTOR_ILLUMINATED_REGION is provided), the list of sources is filtered to remove any sources found in non-illuminated regions of the detector.

5. fors_img_science_photom: used to perform a photometric calibration on the reduced image with the refined WCS. By defaults the sources found before are matched against the synphot Gaia catalog, which contains photometric values derived from Gaia spectra for the standard FORS broad-band filters. If no suitable Gaia stars are found or non-standard filters are used the pipeline will instead determine the photometric calibration from static zeropoints and extinction coefficients. Finally (if the DETECTOR_ILLUMINATED_REGION is provided), the image will be trimmed to the actual illuminated region.

The FORS pipeline also includes the following additional recipes:

- `fors_zeropoint`: used to estimate the atmospheric extinction from one imaging exposure on a photometric standard stars field[3]. The bias master calibration is subtracted from the raw exposure, and the median residual in the overscan regions is subtracted too. The unbiased signal is then divided by the normalised sky flat field produced by the recipe `fors_img_sky_flat`. The calibrated image is sent to a source detection and extraction application (SExtractor). The detected sources are compared to a catalogue of standard stars for identification. The recipe `fors_zeropoint` uses whatever catalogue is specified in input, either Landolt's or Stetson's, or even both[4].

- `fors_photometry`: used for the processing of several `ALIGNED_PHOT` tables produced by the recipe `fors_zeropoint`, enabling the estimation of quantities such as the best atmospheric extinction and linear color correction terms related to a filter. In particular it enables accurate photometry by determining, from `ALIGNED_PHOT` tables derived from a set of dithered (rotated, translated) exposures of a photometric standard star field, a correction map relative to a given flat field frame (the same used in the `fors_zeropoint` processing).

- `fors_dark`: computes a master dark frame from a set of raw dark exposures. Dark levels are so low that the master dark calibration is not applied to the data. This recipe is just used for instrument quality control.

More information of the FORS pipeline is available in the associated FORS Pipeline User Manual.

The following sections describe the data reduction using Reflex.

---

[3]This requires that the zeropoints in the `PHOT_TABLE` are up-to-date, which is not always the case. The user may want to check the instrumental zeropoints at the QC1 database

[4]It should be noted that using simultaneously two different catalogues carries a significant risk to produce inconsistent results, as the magnitudes are not derived with the same method in both catalogues, and as a consequence common stars can differ by up to 0.2 magnitudes. The use of two catalogues is therefore strongly discouraged.

4 Introduction to `EsoReflex`

This document is a tutorial designed to enable the user to to reduce his/her data with the ESO pipeline run under an user-friendly environmet, called `EsoReflex`, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

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

Reflex and the data reduction workflows have been developed by ESO and instrument consortia and they are fully supported. If you have any issue, please have a look to https://support.eso.org to see if this has been reported before or open a ticket for further support.

A workflow accepts science and calibration data, as downloaded from the archive using the CalSelector tool[5] (with associated raw calibrations) and organises them into DataSets, where 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 FORS `Reflex` workflow described in the next sections supports the reduction of FORS1 and FORS2 imaging observations. The user is referred to the FORS User Manual[6] for more information on the instrument itself, and the FORS Pipeline User Manual[7] for the details of the FORS2 imaging pipeline recipes. There are also other tutorials that guide you through the MOS/MXU/LSS workflow and the PMOS workflow, respectively. Check the ESO pipeline main webpage.

The quick start section (see Section 6) describes the minimum effort to get started, and it makes up only two pages of text in this document.

---

[5] *https://www.eso.org/sci/archive/calselectorInfo.html*

[6] available at *http://www.eso.org/sci/facilities/paranal/instruments/fors/doc*

[7] available at *https://www.eso.org/sci/software/pipelines/index.html* (Documentation)

5 Software Installation

`Esoreflex` and the 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 pipelines and Reflex can be installed from the ESO `macports` repositories that support macOS platforms, the and the `rpm/yum` repositories that support Fedora and CentOS platforms. For any other operating system it is recommended to use the `install_esoreflex` script.

The installation from package repository requires administrative privileges (typically granted via sudo), as it installs files in system-wide directories under the control of the package manager. If you want a local installation, or you do not have sudo privileges, or if you want to manage different installations on different directories, then use the `install_esoreflex` script. Note that the script installation requires that your system fulfill several software prerequisites, which might also need sudo privileges.

Reflex 2.11.x needs java JDK 11 to be installed.

Please note that 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.

5.1 Installing `Esoreflex` **workflows via** `macports`

This method is supported for the macOS operating system. It is assumed that macports (*https://www.macports.org*) is installed. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/macports.html*, which also describes the versions of macOS that are currently supported.

5.2 Installing `Esoreflex` **workflows via** `rpm/yum/dnf`

This method is supported for Fedora and CentOS platforms and requires sudo rights. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/rpm.html*, which also describes the versions of Fedora and CentOS that are currently supported.

5.3 Installing `Esoreflex` **workflows via** `install_esoreflex`

This method is recommended for operating systems other than what indicated above, or if the user has no sudo rights. Software dependencies are not fulfilled by the installation script, therefore the user has to install all the prerequisites before running the installation script.

The software pre-requisites for `Reflex 2.11` may be found at:
https://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.11` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

    ```
    wget https://eso.org/sci/software/pipelines/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. Follow all the script instructions; you will be asked whether to use your Internet connection (recommended: yes), the pipelines and demo-datasets to install (note that the installation will remove all previously installed pipelines that are found in the same installation directory).

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

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

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

5.4 Demo Data

Together with the pipeline you will also receive a demo data set, that allows you to run the `Reflex FORS` workflow without any changes in parameters. This way you have a data set to experiment with before you start to work on your own data.

Note that you will need a minimum of ∼0.5 GB, ∼0.6 GB and ∼1 GB of free disk space for the directories `<download_dir>`, `<install_dir>` and `<data_dir>`, respectively. The FORS2 demo data have been retrieved with the CalSelector tool[8].

---

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

6   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 FORS2 demo data set supplied with the `esoreflex 2.11` 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 FORS2 Imaging by typing:

   ```
   esoreflex fors_imaging&
   ```

   Alternatively, you can type only the command `esoreflex` the empty canvas will appear (Figure 6.1) 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 FORS2 Imaging workflow is shown in Figure 6.2.

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 OK .

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 Browse button to select the directory from a file browser. When you have finished, click OK 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 6.3) that lists the datasets along with the values of a selection of useful header keywords[9]. 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 9.3.2.

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 6.4 shows the Product Explorer window. A full description of the *Product Explorer* will be presented in Section 9.3.3.

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 FORS2 workflow that merit a look at the rest of this tutorial.
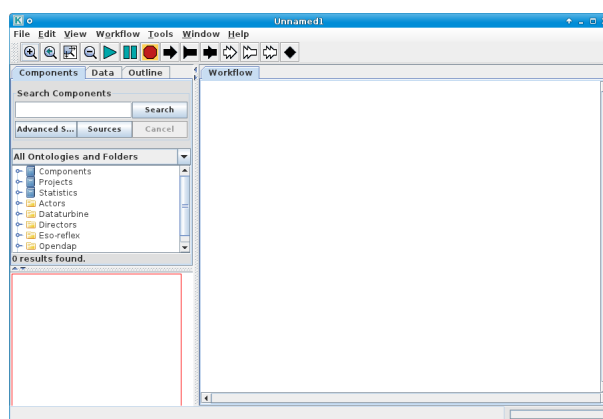

Figure 6.1: *The empty* `Reflex` *canvas.*

---

[9]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.
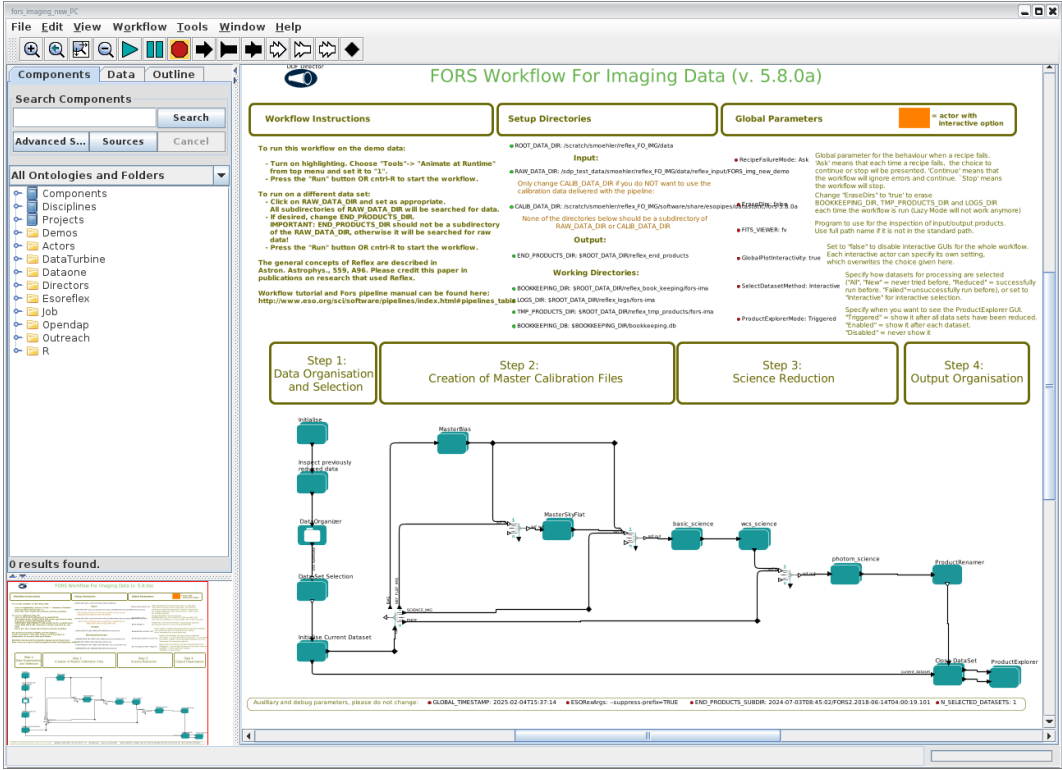
Figure 6.2: *FORS2 imaging workflow general layout.*

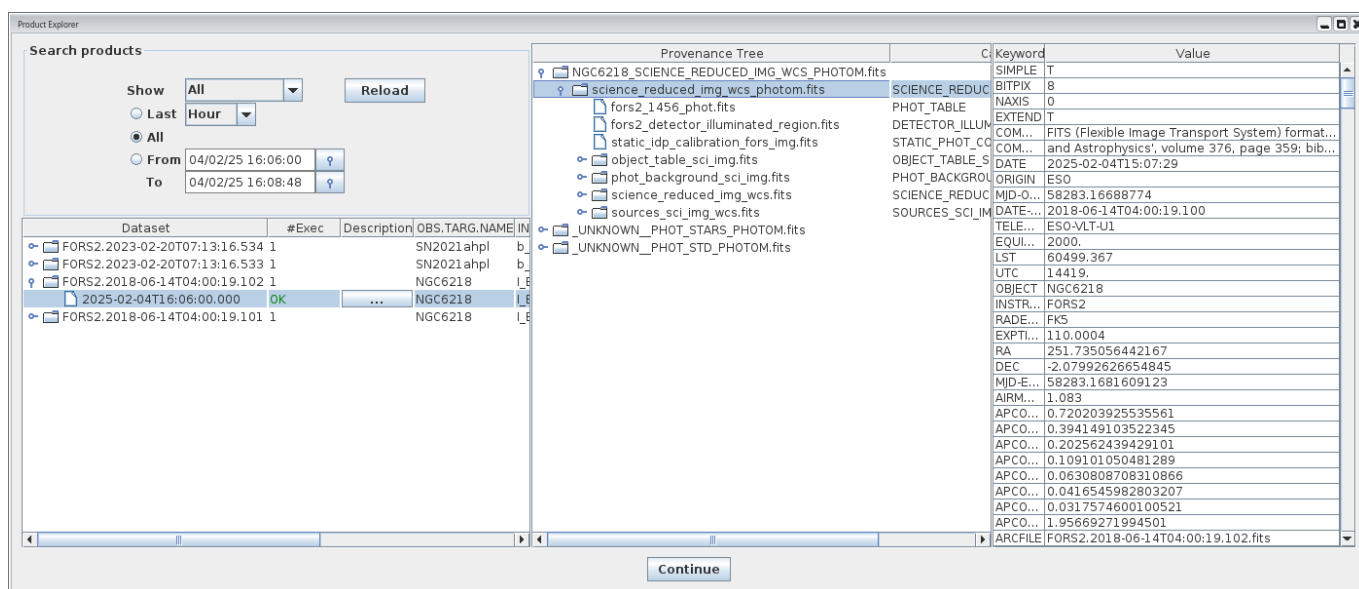Figure 6.3: *The "Select Datasets" pop-up window.*

Figure 6.4: *The Product Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products.*

7 About the main `esoreflex` canvas

7.1 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.

7.2 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.

7.3 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.

8 The FORS2 Workflow

The FORS2 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.

8.1 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; Forchì (2012)).

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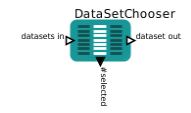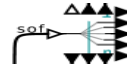
Explorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

8.2 Workflow Actors

8.2.1 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).
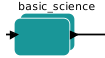
Access to the parameters for a simple actor is achieved by right-clicking on the actor and selecting `Configure Actor`. This will open an "Edit parameters" window. Note that the `Product Renamer` actor is a jython script (Java implementation of the Python interpreter) meant to be customised by the user (by double-clicking on it).

8.2.2 Composite Actors

Composite Actors have workflow symbols that consist of multiple-layered green-blue rectangles. They generally do not have a logo within the rectangle. A Composite Actor represents a combination of one or more Simple or Composite Actors which hides over-complexity from the user in the top-level workflow. In the FORS workflow, the following actors are composite actors:

- - The `Initialise` actor.

- - The `Initialise Current DataSet` actor.

- - The `Product Explorer` actor (contains the simple actor).

- - The `Initialise Current DataSet` actor.

- - The `MasterBias` actor.

- - The `MasterSkyFlat` actor.

- - The `basic_science` actor.

- - The `wcs_science` actor.

- - The `photom_science` actor.

- - The `Close DataSet` actor.

Composite Actors may also be expanded for inspection. To do this, right-click on the actor and select `Open Actor`, which will expand the Composite Actor components in a new `Reflex` canvas window. If the Composite Actor corresponds to a pipeline recipe, then the corresponding `RecipeExecuter` actor will be present as a Simple Actor, and its parameters are accessible as for any other Simple Actor. Alternatively you may still find Composite Actors, on which you need to repeat the first step to access the `RecipeExecuter`.

8.2.3   Recipe Execution within Composite Actors

The FORS workflow contains Composite Actors to run pipeline recipes. This is in the most simple case due to the `SoF Splitter`/`SoF Accumulator`[10], which allow to process calibration data from different settings within one given dataset (e.g. lamp frames taken with different slits/masks). More complex Composite Actors contain several actors (e.g. `Recipe Executer`).

The central elements of any Reflex workflow are the `RecipeExecuter` actors that actually run the recipes. One basic way to embed a `RecipeExecuter` in a workflow is shown in Fig 8.1, which is the most simple version of a Composite Actor. The `RecipeExecuter` is preceded by an `SofSplitter`, and followed by an `SofAccumulator`. The function of the `SofSplitter` is to investigate the incoming SoFs, sort them by "purpose", and create separate SoFs for each purpose. The `RecipeExecuter` then processes each of the SoFs independently (unless they are actually the same files). Finally, the `SofAccumulator` packs all the results into a single output SoF. The direct relation between the `SofSplitter` and `SofAccumulator`

---

[10]SoF stands for Set of Files, which is an ASCII file containing the name (and path) of each input file and its category (e.g. `BIAS`).
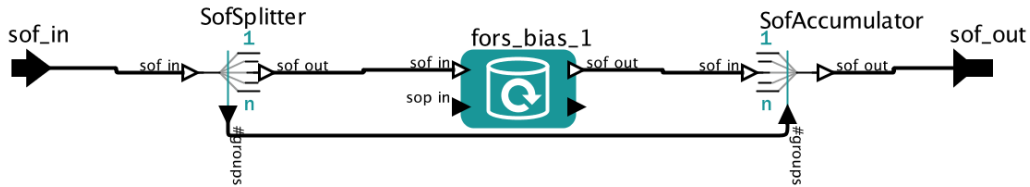
Figure 8.1: *This is the window you get when you choose* Open Actor *for the Composite Actor* MasterBias. *This is the most simple case for a Composite Actor. Using* Configure Actor *on* fors_bias_1 *gives you Fig.* 8.2*.*

Table 8.1: The FORS imaging pipeline actors and their contents

actor	recipes	description
MasterBias	fors_bias	create master bias
MasterSkyFlat	fors_img_sky_flat	create master twilight flat
Science	fors_img_basic_science	correct science data for bias and flat field
Science	fors_img_wcs	determine world coordinate systems for science data
Science	fors_img_science_photom	perform photometric calibration of science data

is used to communicate the number of different SoFs created by the SofSplitter. A workflow will only work as intended if the purpose of all the files a recipe needs as input is identical. The only exception to this rule is that a purpose can also be "default". In this case, the file is included in any output SoF created by the SoFsplitter and SofAccumulator.

The reason for this scheme is best explained by an example. For a complex dataset, the Data Organiser might have selected a large number of individual twilight flat fields. The different twilight flat fields are to be used to calibrate different science frames. The Data Organiser determines and records this "purpose" of each twilight flat field, and this information is included in the dataset and each SoF created from this dataset. The FitsRouter directs all the raw twilight flat field frames of the current dataset to the MasterSkyFlat Composite Actor. The SofSplitter then creates SoFs, one per science frame, for the twilight flat fields to be used for the science frames. The fors_img_sky_flat recipe creates one master flat field (and other products) for each SoF, and the SofAccumulator then creates a SoF that contains all the products.

A RecipeExecuter actor is used in the workflow to run a single FORS pipeline recipe (e.g: in the MasterBias actor the recipe fors_bias is executed). In order to configure the RecipeExecuters, one has to first use Open Actor to get to the level of the recipe executors (see Fig. 8.1).

In Figure 8.2 we show the "Edit parameters" window for a typical RecipeExecuter actor, which can be displayed by right-clicking on the actor and selecting Configure Actor. In the following we describe in more detail the function of some of the parameters for a RecipeExecuter actor:

- The "recipe" parameter states the FORS imaging pipeline recipe which will be executed.

- The "mode" parameter has a pull-down menu allowing the user to specify the execution mode of the actor.

Figure 8.2: *The "Edit parameters" window for a typical* `RecipeExecuter` *actor, the* `fors_bias_1` *actor which runs the* `fors_bias` *pipeline recipe.*

The available options are:

- `Run`: The pipeline recipe will be executed, possibly in Lazy mode (see Section 8.2.4). This option is the default option.

- `Skip`: The pipeline recipe is not executed, and the actor inputs are passed to the actor outputs.

- `Disabled`: The pipeline recipe is not executed, and the actor inputs are not passed to the actor outputs.

• The "Lazy Mode" parameter has a tick-box (selected by default) which indicates whether the `RecipeExecuter` actor will run in Lazy mode or not. A full description of Lazy mode is provided in Sect. 8.2.4.

• The "Recipe Failure Mode" parameter has a pull-down menu allowing the user to specify the behaviour of the actor if the pipeline recipe fails. The available options are:

- `Stop`: The actor issues an error message and the workflow stops.

- `Continue`: The actor creates an empty output and the workflow continues.

- `Ask`: The actor displays a pop-up window and asks the user whether he/she wants to continue or stop the workflow. This option is the default option.

Generally this parameter is set to the global variable $RecipeFailureMode to ensure a common behaviour for a given workflow.

- The set of parameters which start with "recipe param" and end with a number or a string correspond to the parameters of the relevant FORS pipeline recipe. By default in the RecipeExecuter actor, the pipeline recipe parameters are set to their pipeline default values. If you need to change the parameter value for any pipeline recipe, then this is where you should edit the value.

The description of the remainder of the RecipeExecuter actor parameters are outside the scope of this tutorial, and the interested user is referred to the Reflex User Manual for further details. Any changes that you make in the "Edit parameters" window may be saved in the workflow by clicking the ⌴Commit button when you have finished.

8.2.4 Lazy Mode

By default, all RecipeExecuter 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 RecipeExecuter 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 RecipeExecuter actors are actually found inside the composite actors in the top level workflow. To access such embedded RecipeExecuter 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 RecipeExecuter 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.*

9 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 6), but it is not a requirement.

9.1 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.
```

9.2 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>` FORS2 Imaging 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
  -RAW_DATA_DIR /home/user/my_raw_data
  -ROOT_DATA_DIR /home/user/my_reduction
  -END_PRODUCTS_DIR $ROOT_DATA_DIR/reflex_end_products
  fors_imaging
```

to reduce the complete datasets that are present in the directory `/home/user/my_raw_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 -p` FORS2 Imaging. 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 fors_imaging
```

to launch the Product Explorer. The Product Explorer allows you to inspect the data products already reduced by the FORS2 Imaging `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> fors_imaging
```

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 `Refresh` 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 9.3.3

3. Type:

```
esoreflex fors_imaging &
```

to launch the FORS2 Imaging `esoreflex` workflow. The FORS2 Imaging workflow window will appear (Fig. 6.2). Please configure the set-up directories `ROOT_DATA_DIR`, `RAW_DATA_DIR`, and other workflow parameters as needed. Just double-click on them, edit the content, and press `OK`. 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. 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 8.1).

Secondly, if set, the workflow will open the Product Explorer, allowing the user to inspect previously reduced datasets (see Section 9.3.3 for how to configure this option).
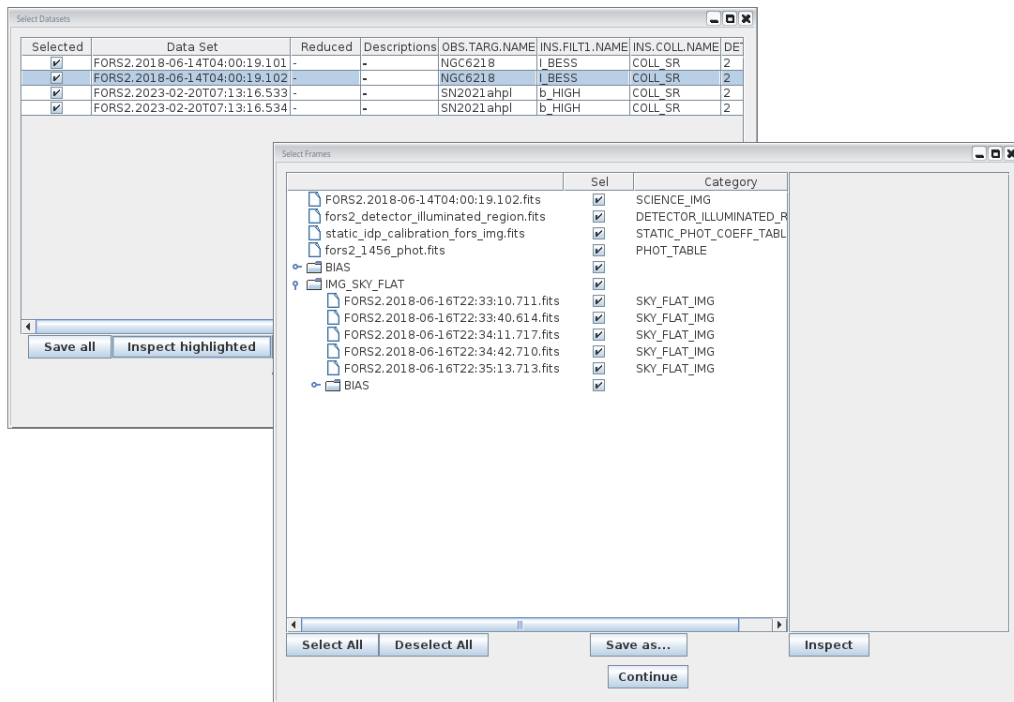
Figure 9.1: *The "Select Frames" window with a single file from the current Data Set highlighted in blue, and the corresponding FITS header displayed in the text box on the right. Hidden partially behind the "Select Frames" window is the "Select DataSets" window with the currently selected DataSet highlighted in blue.*

9.3 Workflow Steps

9.3.1 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

OCA[11] 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 9.3.2.

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 RecipeExecuter. 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 RecipeExecuter. The SofSplitter and SofAccumulator accept several SoFs as simultaneous input. The SofAccumulator creates a single output SoF from the inputs, whereas the SofSplitter creates a separate output SoF for each purpose.

9.3.2 DataSetChooser

The DataSetChooser displays the DataSets available in the "Select Data Sets" window, activating vertical and horizontal scroll bars if necessary (Fig. 6.3).

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

---

[11]OCA 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

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 $\boxed{\texttt{Deselect All}}$ and $\boxed{\texttt{Select Complete}}$ 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 $\boxed{\texttt{Inspect Highlighted}}$, then a "Select Frames" window will appear that lists the set of files that make up the highlighted DataSet including the full filename[12], 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 $\boxed{\texttt{Save As}}$ 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 $\boxed{\texttt{Inspect}}$, 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 $\boxed{\texttt{Continue}}$.

To add a description of the reduction, press the button $\boxed{\texttt{...}}$ 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 $\boxed{\texttt{OK}}$. 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 $\boxed{\texttt{Continue}}$ in order to continue with the workflow reduction, or $\boxed{\texttt{Stop}}$ in order to stop the workflow.

9.3.3 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. 6.4).

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 datareduction:

- 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:

---

[12]keep the mouse pointer on the file name to visualize the full path name.

- "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. 6.4). Once you click on a dataset, you get the list of reduction attemps. Green and red flags identify successfull or unsucessfull 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 succesful (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+Ctr+v to past 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.

9.3.4 Creation Of Master Calibration Files

In this step of the workflow, the following FORS2 recipes are executed in the order listed below. Please refer to the FORS Pipeline User Manual (in particular, its Sections 9 and 10) for the details of each recipe and the algorithms employed:

1. The `MasterBias` actor will execute the FORS2 pipeline recipe `fors_bias` in order to create a combined master bias frame from the set of raw bias frames

2. The `MasterSkyFlat` actor will execute the FORS2 pipeline recipe `fors_img_sky_flat` in order to create from the set of raw twilight flat frames a combined master twilght flat.

9.3.5 Science Reduction

Also here, the following FORS2 recipes are executed in the order listed below. Please refer to the FORS Pipeline User Manual (in particular Sections 9 and 10) for the details of each recipe and the algorithms employed.

1. The `basic_science` actor will execute the FORS2 pipeline recipe `fors_img_basic_science`, to remove the bias and divide by the master flat field.

2. The `wcs_science` actor will execute the FORS2 pipeline recipe `fors_img_wcs` to detect and extract sources, fit and subtract the background, and refine the world coordinate system.

3. The `photom_science` actor will execute the FORS2 pipeline recipe `fors_img_science_photom` to photometrically calibrate the reduce science frame.

The recipe `fors_img_stack`, which stacks, combines and calibrates reduced science frames, is currently not part of the workflow.

9.3.6 Output Organisation

After having processed the input data for a DataSet, the workflow highlights and executes the `Product Renamer` actor, which, by default, will copy the defined final products of the `ForsScience` actor to the directory specified by `END_PRODUCTS_DIR` and rename them with names derived from the values of certain FITS header keywords. Specifically, final products are renamed by default with names of the form `<HIERARCH.ESO.OBS.NAME>_<HIERARCH.ESO.PRO.CATG>.fits` with `<HIERARCH.ESO.OBS.NAME>` and `<HIERARCH.ESO.PRO.CATG>` representing the values of the corresponding FITS header keywords (`<HIERARCH.ESO.OBS.NAME>` is the name of the OB and `<HIERARCH.ESO.PRO.CATG>` is the category of the product file). These names are fully configurable by right-clicking on the `Product Renamer` actor, selecting `Configure Actor`, and then editing the string as appropriate. In some cases the keyword `<HIERARCH.ESO.OBS.TARG.NAME>` (target name) may be more useful than `<HIERARCH.ESO.OBS.NAME>`.

For imaging data the final products that are copied and renamed are (for better readability we replace `<HIERARCH.ESO.OBS.NAME>` by `<OB_NAME>`):

- `<OB_NAME>_SCIENCE_REDUCED_IMG_WCS_PHOTOM.fits` Reduced science image with astrometric and photometric calibration.

- `<OB_NAME>_PHOT_STD_PHOTOM.fits` Table with Gaia stars in the field of view for the filter used (only for broad-band filters BVRI).

- `<OB_NAME>_PHOT_STARS_PHOTOM.fits` Table with observed sources that may be used to determine a zeropoint. Zeropoints using Gaia stars can be determined only for broad-band filters BVRI. Only sources with a value different from zero in the column `WEIGHT` are used for the zeropoint calculation.

The following actors in this step of the workflow are concerned with the termination of the data flow for the current DataSet and will highlight briefly as they are executed.

Finally, the `Product Explorer` window will appear as shown in Fig. 6.4 with a list of datasets on the left menu. By unfolding the menu under each dataset, all the renamed products appear, and if one is interested in the files, including all intermediate steps, that are used to produce that final product, just click on it and a dependency tree will show the whole reduction chain.

10   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 (Forchì (2012)). 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 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 (Forchì (2012)) 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 (Forchì (2012)) 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 5, 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`

10.1 FORS specific questions

- **I have standard star observations in my data pool, but they are not processed - why?**

  The pipeline recipe `fors_zeropoint` that is used to process such data assumes a fixed extinction coefficient, which may not be correct for a given night; it is therefore not included in the workflow. Instead you can find the instrumental zeropoints and nightly extinction values as well as information on the photometric stability of the night for the years 2012 to 2023 at http://archive.eso.org/qc1/qc1_cgi?action=qc1_browse_table&t... For more recent data an in-situ calibration using Gaia stars is implemented.

  See http://www.eso.org/observing/dfo/quality/FORS2/qc/zeropoints/zeropoints.html for more details.

- **Does the pipeline combine different detectors chips into a common product?**

  The recipe `fors_img_stack` can be used to combine processed science data from different detectors into one file.

- **Does the pipeline stack the images of a given detector?**

  The recipe `fors_img_stack` combines, stacks and calibrates reduced science data. The other science recipes work on single frames.
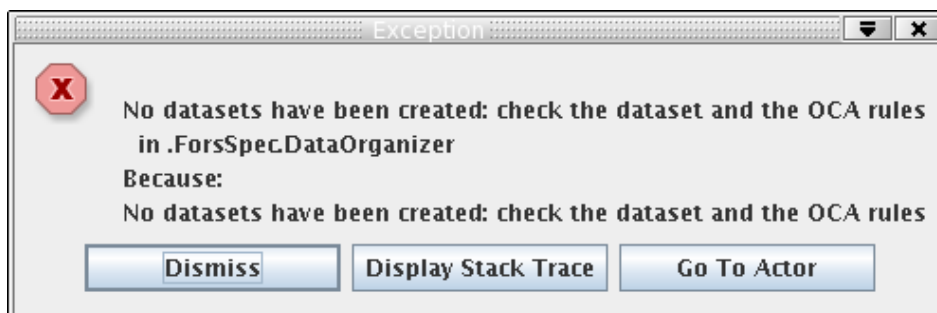
11 Troubleshooting

Figure 11.1: *TheDataOrganizer interactive window reports an error ":No DataSets have been created, check the data set and the OCA rules.".*

1. **I downloaded the data from the ESO archive, put them into a new directory, tried to run** `Reflex` **on them, but**

   (a) **it crashes**

   The current release of the FORS pipeline includes some additional data in the static calibration frames. The recipes would choke if this data is not present. However, the ESO archive with CalSelector may associate calibration data which is old and Reflex will pick the files either from the installed pipeline static data or from the CalSelector in a non-deterministic way. In order to solve the issue, remove the static calibration data downloaded from the archive (all the files starting with M.FORS2).

   This may happen if one of the files was downloaded only partially (check for a file with the extension `fits.Z.part`. You will have to download that file again in order to have an uncorrupted file (and remove the partial one).

   (b) **The DataOrganiser fails with the error message ":No DataSets have been created, check the data set and the OCA rules."(see Figure 11.1.)**

   This error may be due to the fact that the data provided by the ESO archive are compressed (`<filename>.fits.Z`). Please remember to uncompress the data before running the workflow in `Reflex`.

   Also, please remember that the FORS2 imaging workflow supports only imaging data (IMG). It is possible that your data consists entirely of LSS/MOS/MXU/IPOL/PMOS observations, in which case the `Data Organiser` actor will not construct any datasets, showing the mentioned error message.

2. **The "Select DataSets" window displays my datasets, but some/all of them are greyed out. What is going on?**

   If a dataset in the "Select DataSets" window is greyed out, then it means that the dataset which was constructed is missing some key calibration(s) (i.e. the dataset is incomplete). To find out what calibration(s)

are missing from a greyed out dataset, click on the dataset in question to highlight it in blue, and then click on the button $\boxed{\texttt{Inspect Highlighted}}$. The "Select Frames" window that appears will report the category of the calibration products that are missing (e.g. MASTER_BIAS). From this the user has then to determine the missing raw data (in this case bias frames). If static calibrations are missing the mechanism unfortunately does not work, but such data should be found by reflex in

`<install_directory>/calib/<pipeline_version>/cal`

Acknowledgements

Forchì V., 2012, Reflex User Manual, VLT-MAN-ESO-19000-5037, Issue 0.7, *https://ftp.eso.org/pub/dfs/reflex/ReflexUserManual-3.1.pdf* 23, 37, 38

FORS User Manual, VLT-MAN-ESO-13100-1543 *https://www.eso.org/sci/facilities/paranal/instruments/fors/doc.html* 7, 15

FORS Pipeline User Manual, VLT-MAN-ESO-19500-4106 *https://www.eso.org/sci/software/pipelines/index.html* 7, 11, 14, 15, 35