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

Reflex VIMOS/IFU Tutorial

VLT-MAN-ESO-19540-5898

Issue 4.1.11

Date Jan 15th 2025

Prepared:	ESO VIMOS Pipeline Team	Jan 15th 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	28/06/2012	All	Initial version
1.1	04/01/2013	All	Put into the standard tutorial format
2.0	04/01/2013	All	New version for archiving purposes
2.3	01/04/2016	All	Updates for version 3.1.3
2.6	01/05/2020	All	Updates for Reflex 2.10
2.7	20/05/2021	All	Updates for version 4.1.6
2.8	19/05/2022	All	Updates for version 4.1.7
4.1.8	19/05/2023	None	Updates for version 4.1.8 - Version number aligned to pipeline
4.1.10	15/05/2024	None	Updates for version 4.1.10 - Version number aligned to pipeline
4.1.11	Jan 2025	None	CDS URL updated for downloads

Contents

1 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[1] (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.

This tutorial deals with the reduction of VIMOS Integral Field Unit observations only via the VIMOS/IFU workflow. The user is referred to the VIMOS web page (http://www.eso.org/sci/facilities/paranal/instruments/vimos/) for more information on the instrument itself, and the VIMOS pipeline user manual for the details of the pipeline recipes (http://www.eso.org/sci/software/pipelines/).

The workflow uses association rules known to work with files downloaded from the ESO archive with the CalSelector tool (from year 2009 onwards). For older datasets where the data were directly delivered to the PI (e.g. in DVDs) see 8.

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

2 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.

2.1 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.

2.2 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.

2.3 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 2.11.5` may be found at:
https://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.11.5` 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.

2.4 Demo Data

Together with the pipeline you will also receive a demo data set, that allows you to run the `Reflex VIMOS` 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. The demo data for VIMOS includes both data for the IFU and MOS workflows, but a given workflow will only use the data for that mode.

Note that you will need a minimum of ∼1.3 GB, ∼0.6 GB and ∼7.0 GB of free disk space for the directories `<download_dir>`, `<install_dir>` and `<data_dir>`, respectively. The VIMOS demo data have been retrieved with the CalSelector tool[2].

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

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 VIMOS demo data set supplied with the `esoreflex 2.11.5` 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 vimos ifu by typing:

   ```
   esoreflex vimos_ifu&
   ```

   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 vimos ifu 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 `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 3.3) that lists the datasets along with the values of a selection of useful header keywords[3]. 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 3.4 shows the Product Explorer window. A full description of the *Product Explorer* will be presented in Section 6.3.5.

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

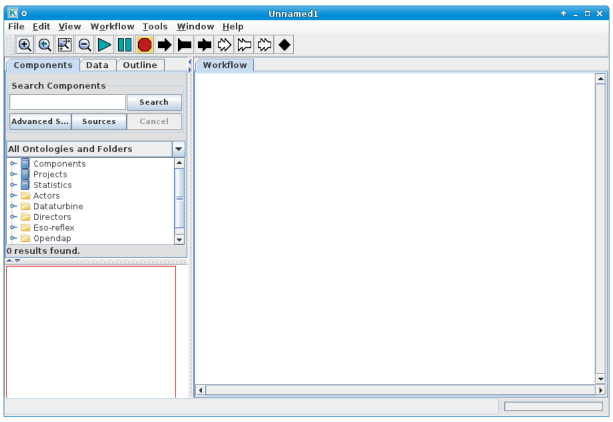
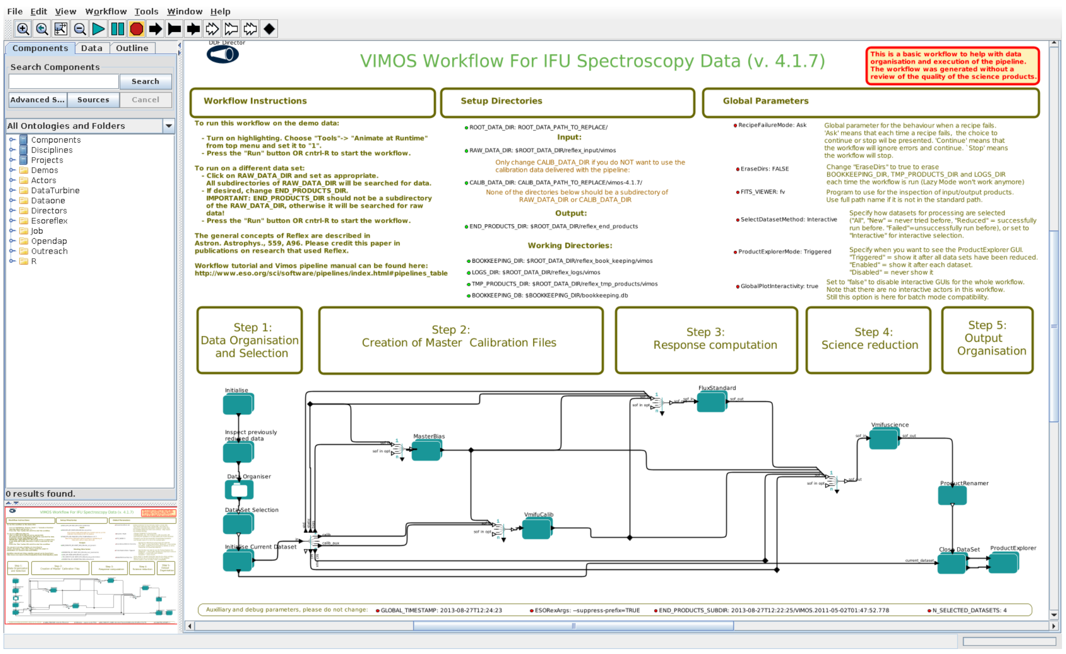


Figure 3.1: *The empty* Reflex *canvas.*

[3]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.


File Edit View Workflow Tools Window Help
Components Data Outline
Search Components
Search
Advanced S... Sources Cancel
All Ontologies and Folders
Components
Disciplines
Projects
Demos
Actors
DataTurbine
Dataone
Directors
Esoreflex
Job
Opendap
Outreach
R
0 results found.
DDF Director
VIMOS Workflow For IFU Spectroscopy Data (v. 4.1.7)
Workflow Instructions
Setup Directories
Global Parameters
Step 1: Data Organisation and Selection
Step 2: Creation of Master Calibration Files
Step 3: Response computation
Step 4: Science reduction
Step 5: Output Organisation
Initialise
Inspect previously reduced data
Data Organiser
DataSet Selection
Initialise Current Dataset
MasterBias
VmifuCalib
FluxStandard
Vmifuscience
ProductRenamer
Close DataSet
ProductExplorer
current_dataset

This is a basic workflow to help with data organisation and execution of the pipeline. The workflow was generated without a review of the quality of the science products.

To run this workflow on the demo data:

- Turn on highlighting. Choose "Tools"-> "Animate at Runtime" from top menu and set it to "1".
- Press the "Run" button OR cntrl-R to start the workflow.

To run on a different data set:
- Click on RAW_DATA_DIR and set as appropriate. All subdirectories of RAW_DATA_DIR will be searched for data.
- If desired, change END_PRODUCTS_DIR. IMPORTANT: END_PRODUCTS_DIR should not be a subdirectory of the RAW_DATA_DIR, otherwise it will be searched for raw data!
- Press the "Run" button OR cntrl-R to start the workflow.

The general concepts of Reflex are described in Astron. Astrophys., 559, A96. Please credit this paper in publications on research that used Reflex.

Workflow tutorial and Vimos pipeline manual can be found here: http://www.eso.org/sci/software/pipelines/index.html#pipelines_table

- ROOT_DATA_DIR: ROOT_DATA_PATH_TO_REPLACE/

Input:

- RAW_DATA_DIR: $ROOT_DATA_DIR/reflex_input/vimos

Only change CALIB_DATA_DIR if you do NOT want to use the calibration data delivered with the pipeline:

- CALIB_DATA_DIR: CALIB_DATA_PATH_TO_REPLACE/vimos-4.1.7/

None of the directories below should be a subdirectory of RAW_DATA_DIR or CALIB_DATA_DIR

Output:

- END_PRODUCTS_DIR: $ROOT_DATA_DIR/reflex_end_products

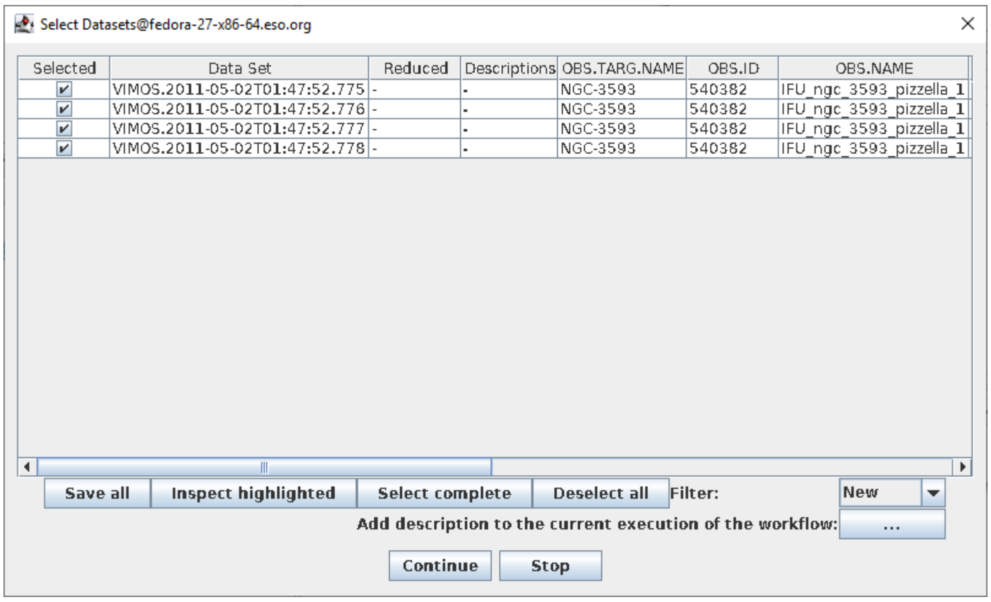
Working Directories:

- BOOKKEEPING_DIR: $ROOT_DATA_DIR/reflex_book_keeping/vimos
- LOGS_DIR: $ROOT_DATA_DIR/reflex_logs/vimos
- TMP_PRODUCTS_DIR: $ROOT_DATA_DIR/reflex_tmp_products/vimos
- BOOKKEEPING_DB: $BOOKKEEPING_DIR/bookkeeping.db

RecipeFailureMode: Ask	Global parameter for the behaviour when a recipe fails. 'Ask' means that each time a recipe fails, the choice to continue or stop wil be presented. 'Continue' means that the workflow will ignore errors and continue. `Stop' means the workflow will stop.
EraseDirs: FALSE	Change "EraseDirs" to true to erase BOOKKEEPING_DIR, TMP_PRODUCTS_DIR and LOGS_DIR each time the workflow is run (Lazy Mode won't work anymore)
FITS_VIEWER: fv	Program to use for the inspection of input/output products. Use full path name if it is not in the standard path.
SelectDatasetMethod: Interactive	Specify how datasets for processing are selected ("All", "New" = never tried before, "Reduced" = successfully run before. "Failed"=unsuccessfully run before), or set to "Interactive" for interactive selection.
ProductExplorerMode: Triggered	Specify when you want to see the ProductExplorer GUI. "Triggered" = show it after all data sets have been reduced. "Enabled" = show it after each dataset. "Disabled" = never show it
GlobalPlotInteractivity: true	Set to "false" to disable interactive GUIs for the whole workflow. Note that there are no interactive actors in this workflow. Still this option is here for batch mode compatibility.

Auxilliary and debug parameters, please do not change: GLOBAL_TIMESTAMP: 2013-08-27T12:24:23 ESORexArgs: --suppress-prefix=TRUE END_PRODUCTS_SUBDIR: 2013-08-27T12:22:25/VIMOS.2011-05-02T01:47:52.778 N_SELECTED_DATASETS: 4

Figure 3.2: *VIMOS/IFU workflow general layout.*

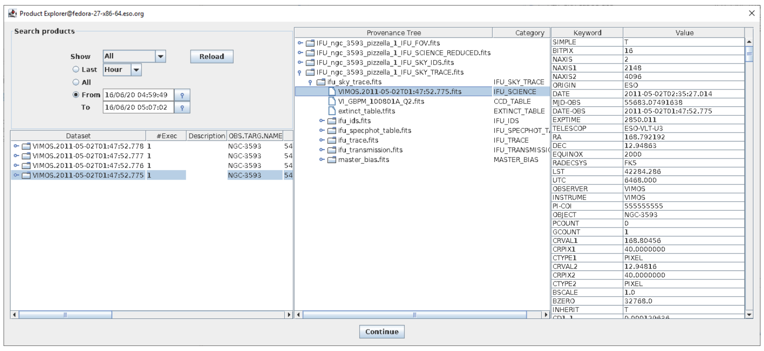


Selected	Data Set	Reduced	Descriptions	OBS.TARG.NAME	OBS.ID	OBS.NAME
✓	VIMOS.2011-05-02T01:47:52.775	-	-	NGC-3593	540382	IFU_ngc_3593_pizzella_1
✓	VIMOS.2011-05-02T01:47:52.776	-	-	NGC-3593	540382	IFU_ngc_3593_pizzella_1
✓	VIMOS.2011-05-02T01:47:52.777	-	-	NGC-3593	540382	IFU_ngc_3593_pizzella_1
✓	VIMOS.2011-05-02T01:47:52.778	-	-	NGC-3593	540382	IFU_ngc_3593_pizzella_1

Filter: New

Add description to the current execution of the workflow: ...

Figure 3.3: *The "Select Datasets" pop-up window.*



Show: All
Last: Hour
All
From: 16/06/20 04:59:49
To: 16/06/20 05:07:02

Dataset	#Exec	Description	OBS.TARG.NAME	
VIMOS.2011-05-02T01:47:52.778	1		NGC-3593	54
VIMOS.2011-05-02T01:47:52.777	1		NGC-3593	54
VIMOS.2011-05-02T01:47:52.776	1		NGC-3593	54
VIMOS.2011-05-02T01:47:52.775	1		NGC-3593	54

Provenance Tree	Category
IFU_ngc_3593_pizzella_1_IFU_FOV.fits	
IFU_ngc_3593_pizzella_1_IFU_SCIENCE_REDUCED.fits	
IFU_ngc_3593_pizzella_1_IFU_SKY_IDS.fits	
IFU_ngc_3593_pizzella_1_IFU_SKY_TRACE.fits	
ifu_sky_trace.fits	IFU_SKY_TRACE
VIMOS.2011-05-02T01:47:52.775.fits	IFU_SCIENCE
VI_GBPM_100801A_Q2.fits	CCD_TABLE
extinct_table.tfits	EXTINCT_TABLE
ifu_ids.fits	IFU_IDS
ifu_specphot_table.fits	IFU_SPECPHOT_T
ifu_trace.fits	IFU_TRACE
ifu_transmission.fits	IFU_TRANSMISSI
master_bias.fits	MASTER_BIAS

Keyword	Value
SIMPLE	T
BITPIX	16
NAXIS	2
NAXIS1	2148
NAXIS2	4096
ORIGIN	ESO
DATE	2011-05-02T02:35:27.014
MJD-OBS	55683.07491638
DATE-OBS	2011-05-02T01:47:52.775
EXPTIME	2850.011
TELESCOP	ESO-VLT-U3
RA	168.792192
DEC	12.94863
EQUINOX	2000
RADECSYS	FK5
LST	42284.286
UTC	6468.000
OBSERVER	VIMOS
INSTRUME	VIMOS
PI-COI	555555555
OBJECT	NGC-3593
PCOUNT	0
GCOUNT	1
CRVAL1	168.80456
CRPIX1	40.0000000
CTYPE1	PIXEL
CRVAL2	12.94816
CRPIX2	40.0000000
CTYPE2	PIXEL
BSCALE	1.0
BZERO	32768.0
INHERIT	T

Figure 3.4: *The Product Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products.*

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.

5 The VIMOS Workflow

The VIMOS 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; 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 5.2.4), 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.

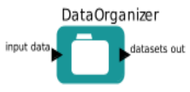
The parameter `ProductExplorerMode` controls whether the `ProductExplorer` actor will show its window or not. The possible values are `Enabled`, `Triggered`, and `Disabled`. `Enabled` opens the Product-

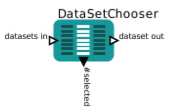
Explorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

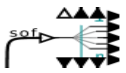
5.2 Workflow Actors

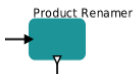
5.2.1 Simple Actors

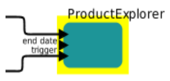
Simple actors have workflow symbols that consist of a single (rather than multiple) green-blue rectangle. They may also have an icon within the rectangle to aid in their identification. The following actors are simple actors:

- 

 - 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).

5.2.2 Composite Actors

Composite Actors have workflow symbols that consist of multiply-layered green-blue rectangles. They generally do not have a logo within the rectangle. A Composite Actor represents a combination of more Simple or Composite Actors which hides over-complexity from the user in the top-level workflow.

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 `Recipe Executer`.

5.2.3 Recipe Execution within Composite Actors

The VIMOS workflow contains Composite Actors to run pipeline recipes. This is in the most simple case due to the `SoF Splitter/SoF Accumulator`[4], which allow to process calibration data from different setting within one given DataSet (e.g. lamp frames taken with different slits/masks). More complex Composite Actors contain several actors (e.g. `Recipe Executer`).

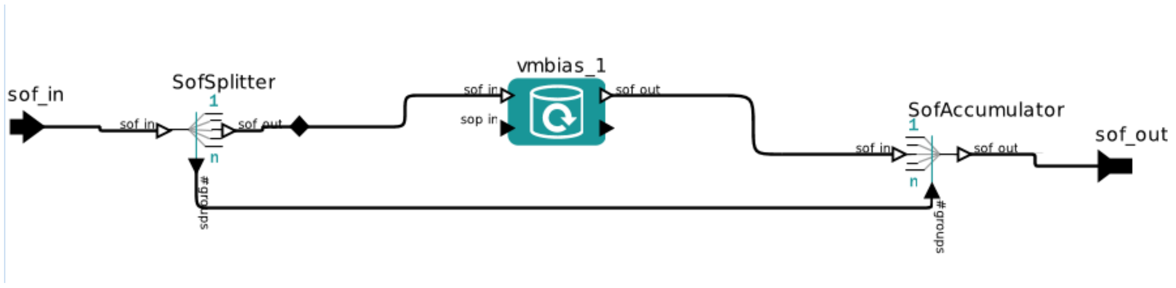


Figure 5.1: *This is the window you get when you choose* `Open Actor` *for the Composite Actor* `MasterBias`. *This is the most simple case for a Composite Actor. Using* `Configure Actor` *on* `vimos_bias_1` *gives you Fig. 5.2.*

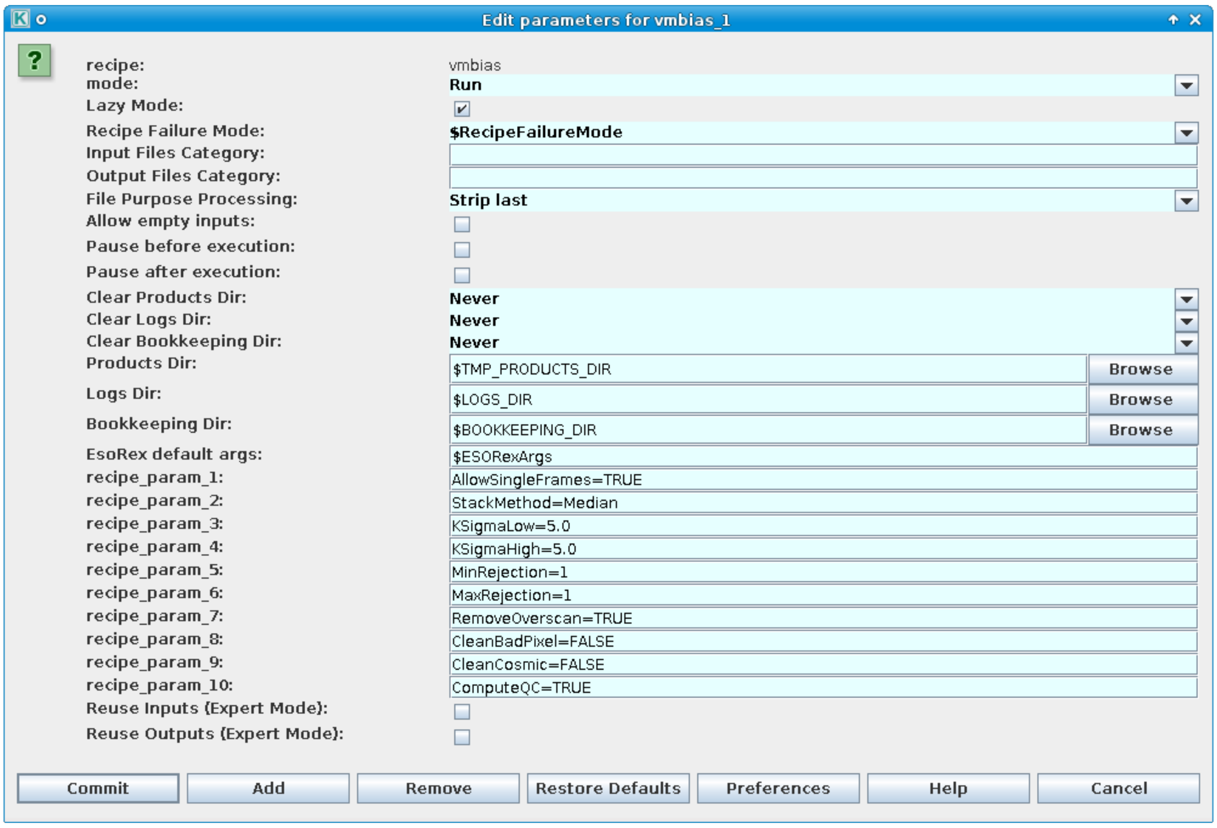
The central elements of any Reflex workflow are the `RecipeExecuter` actors that actually run the recipes. One basic way to embed a `RecipeExecuter` in a workflow is shown in Fig 5.1, which is the most simple version of a Composite Actor. The `RecipeExecuter` is preceded by an `SofSplitter`, and followed 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` 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 raw lamp frames (arc and flat field). The different lamp frames are to be used to calibrate different frames, e.g. the science frames and the standard star frames. The `Data Organiser` determines and records this “purpose” of each lamp frame, and this information is included in the DataSet and each SoF created from this DataSet. The `FitsRouter` directs all raw lamp frames to the calibration Composite Actor. The `SofSplitter` then creates SoFs, one for the lamp frames to be used for the science frames, and (probably) separate ones for the lamp frames to be used for the standard star observations. The calibration 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 VIMOS pipeline recipe (e.g: in the `MasterBias` actor the recipe `vmbias` 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. 5.1).

In Figure 5.2 we show the “Edit parameters” window for a typical `RecipeExecuter` actor, which can be

[4]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`).



recipe:	vmbias
mode:	Run
Lazy Mode:	✓
Recipe Failure Mode:	$RecipeFailureMode
Input Files Category:	
Output Files Category:	
File Purpose Processing:	Strip last
Allow empty inputs:	☐
Pause before execution:	☐
Pause after execution:	☐
Clear Products Dir:	Never
Clear Logs Dir:	Never
Clear Bookkeeping Dir:	Never
Products Dir:	$TMP_PRODUCTS_DIR
Logs Dir:	$LOGS_DIR
Bookkeeping Dir:	$BOOKKEEPING_DIR
EsoRex default args:	$ESORexArgs
recipe_param_1:	AllowSingleFrames=TRUE
recipe_param_2:	StackMethod=Median
recipe_param_3:	KSigmaLow=5.0
recipe_param_4:	KSigmaHigh=5.0
recipe_param_5:	MinRejection=1
recipe_param_6:	MaxRejection=1
recipe_param_7:	RemoveOverscan=TRUE
recipe_param_8:	CleanBadPixel=FALSE
recipe_param_9:	CleanCosmic=FALSE
recipe_param_10:	ComputeQC=TRUE
Reuse Inputs (Expert Mode):	☐
Reuse Outputs (Expert Mode):	☐

Figure 5.2: *The "Edit parameters" window for a typical* `RecipeExecuter` *actor, the* `vmbias_1` *actor which runs the* `vmbias` *pipeline recipe.*

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 VIMOS 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. The available options are:
  - `Run`: The pipeline recipe will be executed, possibly in Lazy mode (see Section 5.2.4). 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. 5.2.4.
- 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:

Table 5.1: The VIMOS/IFU pipeline actors and their contents

actor	recipes	description
MasterBias	`vmbias`	create master bias
VmIfuCalib	`vmifucalib`	create master flat, determine coefficients for wavelength calibration and correction of spatial distortion
FluxStandard	`vmifustandard`	determine response function
VmIfuscience	`vmifuscience`	reduce science data

  - `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.

- The set of parameters which start with "recipe param" and end with a number or a string correspond to the parameters of the relevant VIMOS 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 default parameter value for any pipeline recipe, then this is where you should edit the value. For more information on the VIMOS pipeline recipe parameters, the user should refer to the VIMOS pipeline user manual (Izzo et al. 2012[5]).

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 (Forchì 2012). Any changes that you make in the "Edit parameters" window must be saved in the workflow by clicking the `Commit` button when you have finished to take effect. If you want to reuse the parameters you have to save the workflow with the saved parameters.

5.2.4 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

[5] Available at *https://ftp.eso.org/pub/dfs/pipelines/vimos/vimos-pipeline-manual-4.1.11.pdf*

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.*

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>` vimos ifu 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
  vimos_ifu
```

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` vimos ifu. 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 vimos_ifu`

   to launch the Product Explorer. The Product Explorer allows you to inspect the data products already reduced by the vimos ifu `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> vimos_ifu`

   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 6.3.5

3. Type:

   `esoreflex vimos_ifu &`

   to launch the vimos ifu `esoreflex` workflow. The vimos ifu workflow window will appear (Fig. 3.2). 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 5.1).

Secondly, if set, the workflow will open the Product Explorer, allowing the user to inspect previously reduced datasets (see Section 6.3.5 for how to configure this option).

6.3 Workflow Steps

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 OCA[6] 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.

[6]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`

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.

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 filename[7], 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.

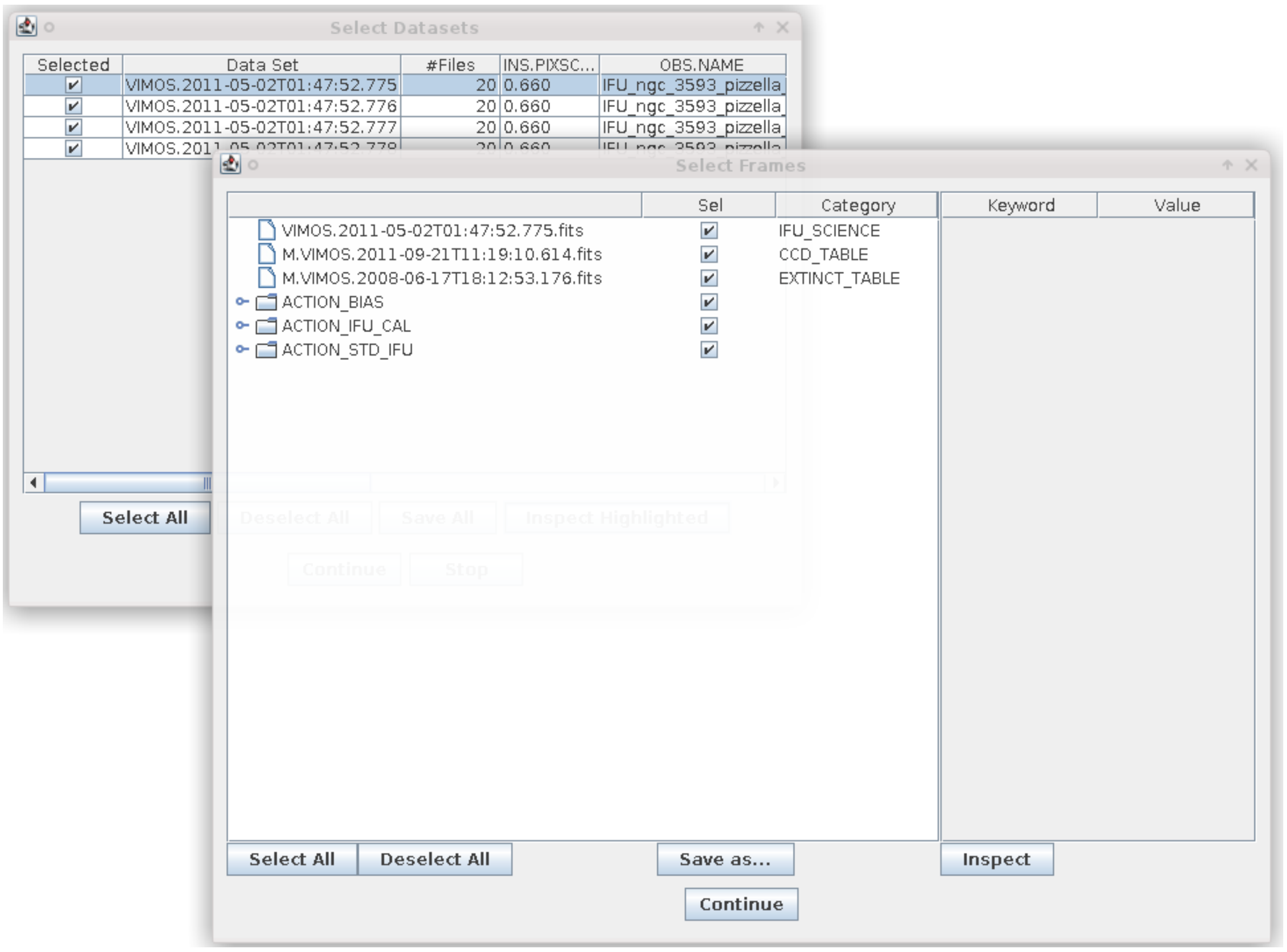
By clicking on the line corresponding to a particular file in the “Select Frames” window, the file will be highlighted in blue, and the file FITS header will be displayed in the text box on the right, allowing a quick inspection of useful header keywords. If you then click on `Inspect`, the workflow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

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

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.



Selected	Data Set	#Files	INS.PIXSC...	OBS.NAME
✓	VIMOS.2011-05-02T01:47:52.775	20	0.660	IFU_ngc_3593_pizzella
✓	VIMOS.2011-05-02T01:47:52.776	20	0.660	IFU_ngc_3593_pizzella
✓	VIMOS.2011-05-02T01:47:52.777	20	0.660	IFU_ngc_3593_pizzella
✓	VIMOS.2011-05-02T01:47:52.778	20	0.660	IFU_ngc_3593_pizzella

	Sel	Category
VIMOS.2011-05-02T01:47:52.775.fits	✓	IFU_SCIENCE
M.VIMOS.2011-09-21T11:19:10.614.fits	✓	CCD_TABLE
M.VIMOS.2008-06-17T18:12:53.176.fits	✓	EXTINCT_TABLE
ACTION_BIAS	✓	
ACTION_IFU_CAL	✓	
ACTION_STD_IFU	✓	

Keyword	Value

Figure 6.1: *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.*

6.3.3 Recipe execution

Once the datasets to be reduced are selected, press the `Continue` button on the dataset organised window to proceed with the data reduction. The workflow will automatically execute the pipeline recipes and construct the `.sof` files to feed the pipeline recipes with. Each sof file will be saved in the `BOOKKEPING_DIR` directory (and subdirectory within it), depending on the recipe it is associated to and the execution time. The pipeline parameters can be changed as shown in figure 5.2.

6.3.4 Final products

Once a dataset is reduced (i.e. when the `vmifuscience` recipe is terminated), a window containing the list of science product is popped out. Each file can be inspected with the selected fits viewer. Final science products will be stored in the `END_PRODUCTS_DIR`, and sorted by execution time and dataset identifier (i.e. the name of the science frame the dataset is for). Default names for the science products are: `IFU_<OB name>_IFU_FOV.fits`, `IFU_<OB name>_IFU_SCIENCE_FLUX_REDUCED.fits`, `IFU_<OB name> _IFU_SCIENCE_REDUCED.fits`, `IFU_<OB name>_IFU_SKY_IDS.fits`, and `IFU_<OB name>_IFU_ SKY_TRACE.fits`. We refer the user to the VIMOS pipeline manual for the description of these files.

6.3.5 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. 3.4).

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:
     – "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. 3.4). 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.

7 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 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`

8 Troubleshooting

In this section we describe some of the problems that may occur when reducing the VIMOS/IFU with the ESOrex pipeline. For a more comprehensive description we refer the user to the VIMOS user manual (`http://www.eso.org/sci/software/pipelines/`).

1. **I have data from the old PI packs**

   For older datasets where the data were directly delivered to the PI (e.g. in DVDs) different set of association rules must be used. These are no longer supported, but we provide a compatible set of rules. To use them, right click on the DataOrganiser actor and select Configure Actor. Then in the OCA file configuration, choose file vimos_ifu_wkf.dvd.oca, which is in the same directory as the default vimos_ifu_wkf.oca one. Moreover, the data from the DVD has to be cleaned up:

   - remove *all* the pipeline products i.e. master bias (whose filename contains the string `MBIA`), transmission response files (whose filename contains the string `PTNF`) and directories labelled as `proc` or `reduced`
   - remove duplicate files (i.e. files with the same name, but stored in different directories, such as arc lamps).

2. **Should I change the CALIB_DATA_DIR configuration?**

   This directory is setup automatically to point to the calibration database provided with the pipeline and in principle shouldn't be changed. However, if static calibration data are present in the RAWDATA_DIR (e.g. calibrations are downloaded from the archive, or copied from ESO-DVD distribution), then you might have to set this directory equal to RAWDATA_DIR (otherwise an obsolete static calibration file may be selected instead of the most appropriate one).

3. **Fibre misidentification**

   The recipe `vmifucalib` uses the `IFU_IDENT` calibration file [8] to identify the fibre in the flat field. If a fibre identification file is not specified, the fibre spectra identification is still attempted, but the result is not always correct.

   The IFU fibre identification performed by recipe `vmifucalib` appears to be negatively affected by changes in temperature. If in the recipe products more than about 50 fibres appear to be "lost" in one pseudo-slit, it may help to rerun the recipe using the `blind` fibre identification method: this method is always triggered if no fibre identification table is specified in the input set-of-frames.

   There are 2 main ways to recognise fibre misidentification problems.

   - A defined set of fibres has intentionally null transmission in each quadrant. This help the cross-correlation in the fibre identification process. These dead fibres are 20-21, 60-61, 100-101 and so on, i.e. two fibres each 40. By inspecting the 2D reduced spectrum `SCIENCE_FLUX_REDUCED.fits` it is possible to see whether the "dead fibres" are placed in the correct positions (see Figure 8.2).

[8] `IFU_IDENT` consists of intensity profiles (one for each IFU pseudo-slit) cut along the cross-dispersion direction of a reference flat field exposure where the fibre spectra have been safely identified. The fibres corresponding to the peak positions of each profile are listed in the `IFU_IDENT` file. Such safe identifications would then be transferred to the new input flat fields by cross-correlation. In the calibration directories there is ideally one `IFU_IDENT` file for each quadrant/grism combination, named `ifu_ident_grism_q.fits` (where `q` indicates the VIMOS quadrant number, and `grism` the grism name).

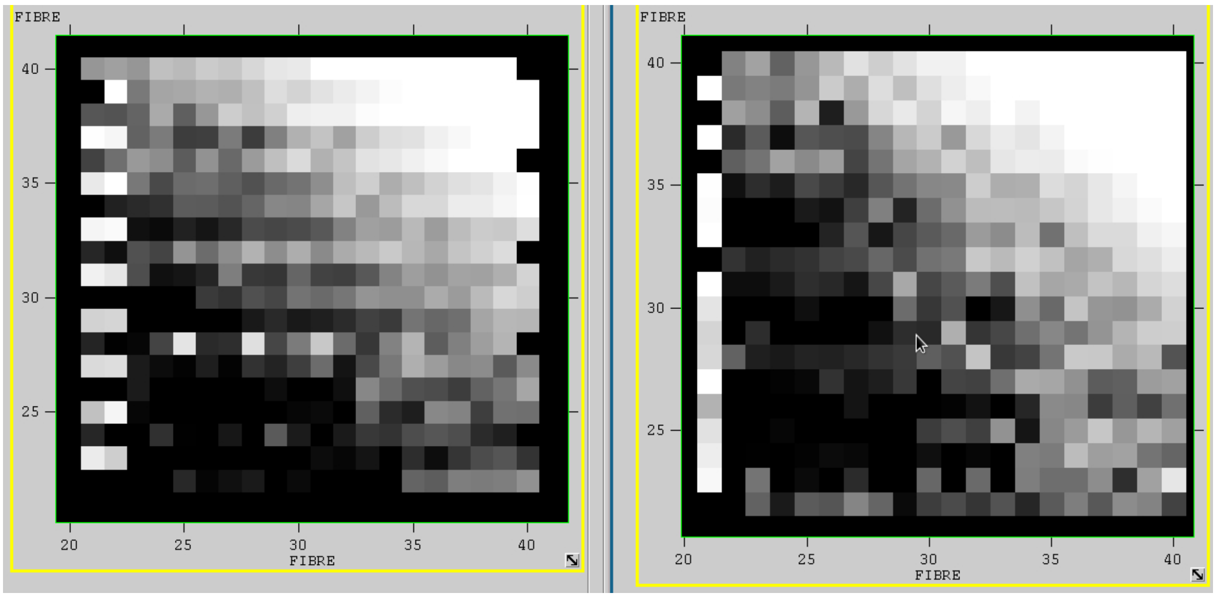


Figure 8.1: Example of field of view where fibre misidentification shuffled horizontally the fibre position (left panel), compared to one where fibre identification worked properly (right panel). The field of view refers only to quadrant 3.

- A fibre misidentification would appear later on the reconstructed image of the field-of view (generated by the `vmifuscience` recipe) as zig-zagging patterns breaking the generally smooth look of the intensity distribution. The field of view file `FOV` can be inspected each time the workflow has reduced a dataset setting `mode` to `INSPECT` to the `DataFilter2` actor (right-click with the mouse on the actor, and select `configure actor`).

  An example of the effect that fibre misidentification has on the field-of-view is shown in Figure 8.1.

If a fibre misidentification occurs, and the `blind` method in `vmifucal` does not work (see point n° 2 above), one solution could be to manually shuffle the fibre of 1 position, according to the fibre coordinates specified in the `IFU_TABLES` [9]. For example with reference to quadrant number 3, if the fibre n° 290 at pixel coordinates (30,35) on the field of view should be placed in at the pixel coordinates (31,35) instead, because of a fibre mismatch problem, it need to be re-identified as fibre number 291, and so on. The inspection of the re-shuffled FOV helps in understanding if the correction was done in a proper way.

Unfortunately, this shuffle correction is not supported in the VIMOS/IFU reflex workflow, and must be operated by the user on the reduced files on a case-by-case basis.

4. **Optimising the number of recovered fibres.**

The fibre identification process operated in `vmifucal` select useful fib-res on the basis of a trace rejection algorithm. The parameter that regulates this procedure is `MaxTraceRejection`. MaxTraceRejection sets the maximum percentage of rejected positions in fibre spectra tracing (default = 50). In the fibre tracing operation, a number of pixel positions may be rejected because the detected position outlays the general trend, or because the signal level is too low. When the percentage of rejected positions is more

[9]The `IFU_TABLES` are static calibration files that contain the correspondence between fibre number in the 2D spectrum and the pixel coordinate on the field of view. See Section 6.21 of the VIMOS manual (version 6.5) for further details.

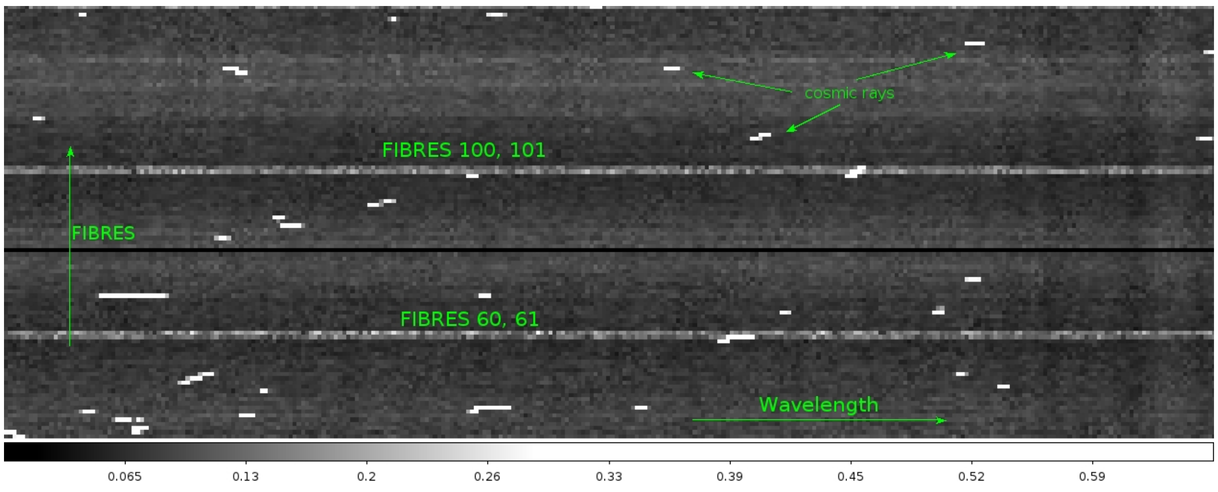


Figure 8.2: Example of 2D spectra where the marked fibres (60-61, and 100-101) are properly identified.

than what is specified here, then the corresponding fibre is flagged as *dead* and excluded from further processing. This parameter can be modified to recover the majority of useful fibres. For example, for the HR grism the range 10– 80 gives good results. The expected number of good fibres is about 380 (quadrants 1 and 3), 310 (quadrant 2) and 360 (quadrant 4).

The `MaxTraceRejection` can be optimised in the VIMOS IFU Reflex workflow by entering the VmIfuCalib subworkflow (right click and *Open Actor*) and double clicking on the recipe actor vmifucalib_1, similar to the figure 5.2.

5. **Bug: MODE mismatch between BIAS frames and pixel tables**

   It can happen that the BIAS frames associated to the reduction flow of one IFU observations was taken in the imaging mode. The bias can be used, as there is no difference between bias taken in imaging or IFU modes. Nevertheless, the `vmbias` recipe requires the bias frame and the bad pixel table[10] to have the same observing mode. This bug can be overcome by changing the `OBSMODE` in the bias frames from `IMG` to `IFU`. This bug is in the `vmbias` recipe, independent from the reflex workflow, and it may be solved in future VIMOS pipeline releases.

[10] `CCD_TABLEs` are static calibration files, one per quadrant, one set per observing mode, containing the list of bad pixels. See Section 6.2 in the VIMOS pipeline user manual (version 6.5).

References

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* 15, 19, 28, 29

ESO VIMOS Pipeline Team, VIMOS Pipeline User Manual, VLT-MAN-ESO-19500-3355, Issue 6.6 19