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-19500-5296

Issue 1.3

Date 2013-04-01

Prepared:	`Lodovico Coccato`	`2013-04-01`	
	Name	Date	Signature
Approved:			
	Name	Date	Signature
Released:			
	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

Contents

1 Introduction And Scope

`Reflex` is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine (Kepler[1]) to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers.

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

This document is a tutorial designed to enable the user to employ the VIMOS workflow to reduce his/her data in a user-friendly way, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

A workflow accepts science and calibration data, as delivered to PIs in the form of PI-Packs (until October 2011) or downloaded from the archive using the CalSelector tool[2] and organises them into DataSets, 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 through 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 and 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] *http://kepler-project.org*
[2] *http://www.eso.org/sci/archive/calselectorInfo.html*

2 Software Installation

The software pre-requisites for `Reflex 2.4` may be found at:
http://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.4` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

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

2. Make the installation script executable:

   ```
   chmod u+x install_reflex
   ```

3. Execute the installation script:

   ```
   ./install_reflex
   ```

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

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

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

   ```
   <install_dir>/bin/reflex
   ```

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

3 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[3].

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

4 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 `Reflex 2.4` release. By following these steps, the user should have enough information to attempt a reduction of his/her own data without any further reading:

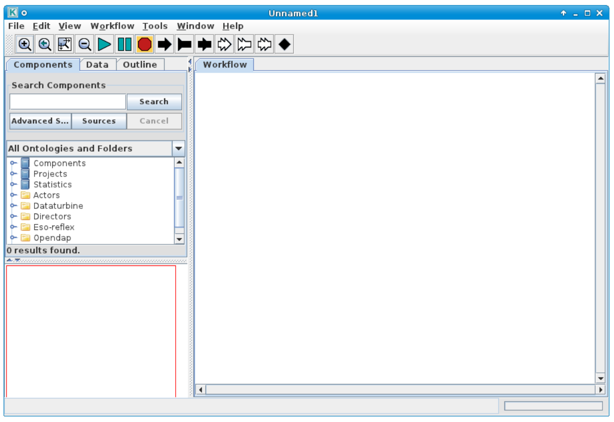


Figure 4.1: *The empty* `Reflex` *canvas.*

1. Start the `Reflex` application:

   ```
   reflex &
   ```

   The empty `Reflex` canvas as shown in Figure 4.1 will appear.

2. Now open the VIMOS workflow by clicking on `File -> Open File`, selecting first `vimos-2.9.7` and then the file `VimosIfu.xml` in the file browser. You will be presented with the workflow canvas shown in Figure 4.2. Note that the workflow will appear as a canvas in a new window.

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

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

5. Click the ▶ button to start the workflow

6. The workflow will highlight the `Data Organiser` actor which has recursively scanned the raw data directory (specified by the parameter `RAWDATA_DIR` under "Setup Directories" in the workflow canvas) and constructs the DataSets. Note that the calibration and reference data must be present either in `RAWDATA_DIR` or in `CALIB_DATA_DIR`, otherwise DataSets may be incomplete and cannot be processed. However, if the same reference file was downloaded twice in different places this creates a problem as `Reflex` cannot decide which one to use.

7. The `Data Set Chooser` actor will be highlighted next and will display a "Select Datasets" window (see Figure 4.3) that lists the DataSets along with the values of a selection of useful header keywords[5]. The first column consists of a set of tick boxes which allow the user to select the DataSets to be processed, and by default all complete DataSets are selected.

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.

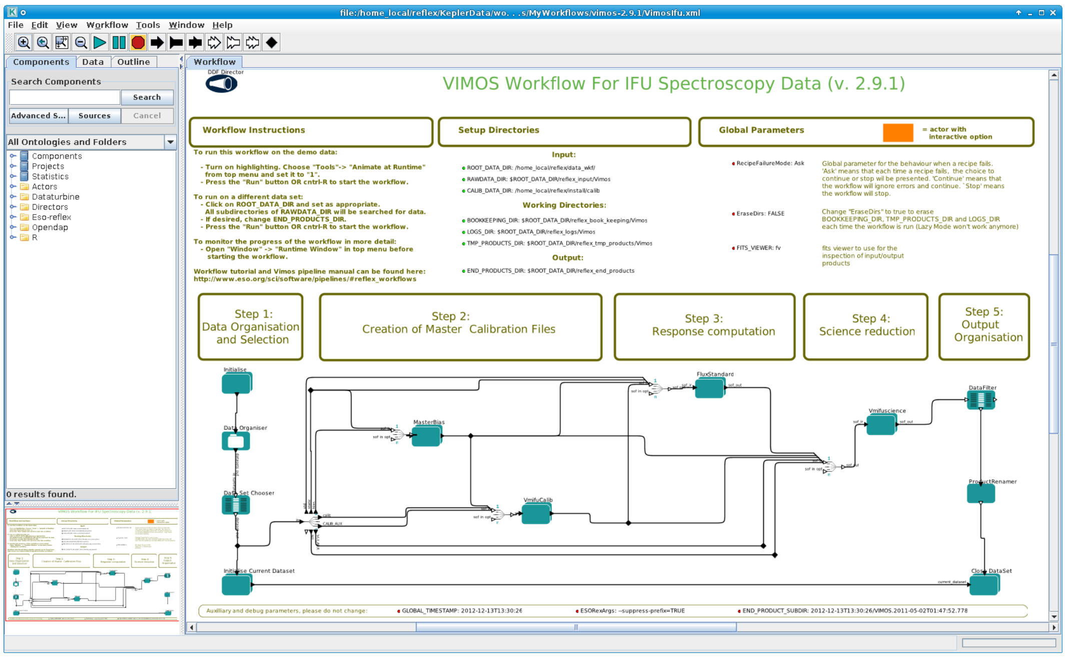
9. When the reduction of the current DataSet finishes, a pop-up window will appear showing the directory were the final products have been saved.

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

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

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.

[5]The keywords listed can be changed by right-clicking on the `DataOrganiser` Actor, selecting `Configure Actor`, and then changing the list of keywords in the second line of the pop-up window. Make sure that the `Lazy Mode` is not active and then click on `Commit` to save the change.



VIMOS Workflow For IFU Spectroscopy Data (v. 2.9.1)

Workflow Instructions

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 ROOT_DATA_DIR and set as appropriate. All subdirectories of RAWDATA_DIR will be searched for data.
- If desired, change END_PRODUCTS_DIR.
- Press the "Run" button OR cntrl-R to start the workflow.

To monitor the progress of the workflow in more detail:

- Open "Window" -> "Runtime Window" in top menu before starting the workflow.

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

Setup Directories

Input:

- ROOT_DATA_DIR: /home_local/reflex/data_wkf/
- RAWDATA_DIR: $ROOT_DATA_DIR/reflex_input/Vimos
- CALIB_DATA_DIR: /home_local/reflex/install/calib

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

Output:

- END_PRODUCTS_DIR: $ROOT_DATA_DIR/reflex_end_products

Global Parameters 🟧 = actor with interactive option

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	fits viewer to use for the inspection of input/output products

Step 1: Data Organisation and Selection

Step 2: Creation of Master Calibration Files

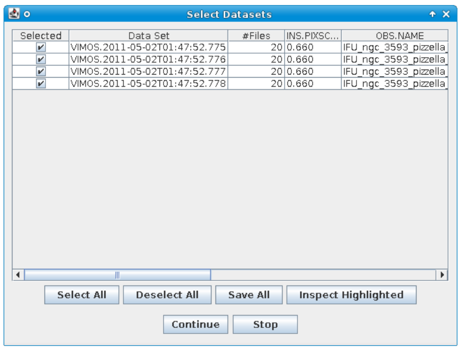
Step 3: Response computation

Step 4: Science reduction

Step 5: Output Organisation

Auxilliary and debug parameters, please do not change:
- GLOBAL_TIMESTAMP: 2012-12-13T13:30:26
- ESORexArgs: --suppress-prefix=TRUE
- END_PRODUCT_SUBDIR: 2012-12-13T13:30:26/VIMOS.2011-05-02T01:47:52.778

Figure 4.2: *VIMOS/IFU workflow general layout.*



Select Datasets

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

Figure 4.3: *The “Select Datasets” pop-up window.*

5 About The Reflex Canvas

5.1 Saving And Loading Workflows

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

5.2 Buttons

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

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

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

5.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the ▶, ⏸, and ⏹ buttons, respectively. A workflow is executed by clicking the ▶ button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the ⏹ button, or the workflow may be paused by clicking the ⏸ button which will allow the current actor/recipe to finish execution before the workflow is actually paused. Note that after clicking the ⏸ button, it is possible that more than one actor is executed, since this behaviour depends on the workflow scheduling. For instance, if there are two actors in parallel, and you pause the workflow while one is being executed, then both of them will be executed before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the ▶ button again.

6 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.

6.1 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 `ROOT_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. Raw data should be stored in a subdirectory of `ROOT_DATA_DIR`, defined by the parameter `RAWDATA_DIR`, which is recursively scanned by the `Data Organiser` actor for input raw data. The directory `CALIB_DATA_DIR`, which is 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)).

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

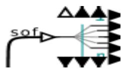
By default the `EraseDirs` parameter is set to `false`, which means that no directories are cleaned before executing the workflow, and the recipe actors will work in Lazy mode (see Section 6.2.4), reusing the previous pipeline recipe outputs where input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the `EraseDirs` parameter to `true`, which forces the workflow to recursively delete the contents of the directories specified by `BOOKKEEPING_DIR`, `LOGS_DIR`, and `TMP_PRODUCTS_DIR`. This is useful for keeping disk space usage to a minimum and will force the workflow to fully rereduce the data each time the workflow is run.

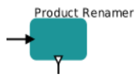
6.2 Workflow Actors

6.2.1 Simple Actors

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

-
DataOrganizer
input data
datasets out
 - The `Data Organiser` actor.

-
DataSetChooser
datasets in
dataset out
selected
 - The `Data Set Chooser` actor.
- 

 - The `Fits Router` actor
- 

 - The `Product Renamer` actor.
-
DataFilter
sof in
sof out
sof rejected
 - The `Data Filter` 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).

6.2.2 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`.

6.2.3 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`[6], 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`).

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 6.1, 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`

[6]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`).

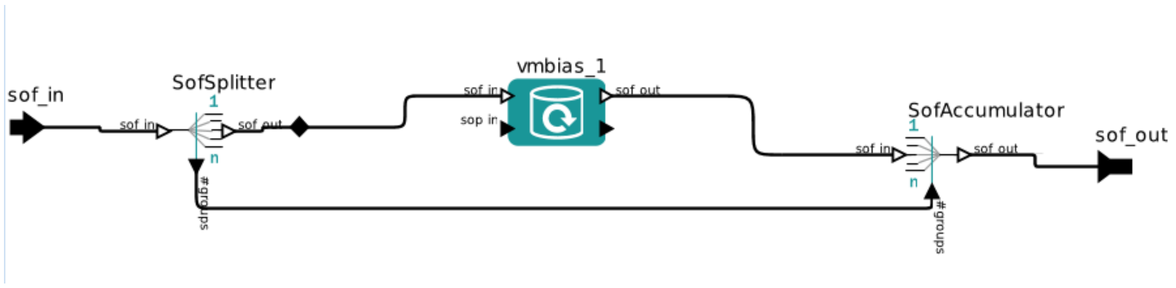


Figure 6.1: *This is the window you get when you choose* `Open Actor` *for the Composite Actor* `MasterBias`. *This is the most simple case for a Composite Actor. Using* `Configure Actor` *on* `vimos_bias_1` *gives you Fig. 6.2.*

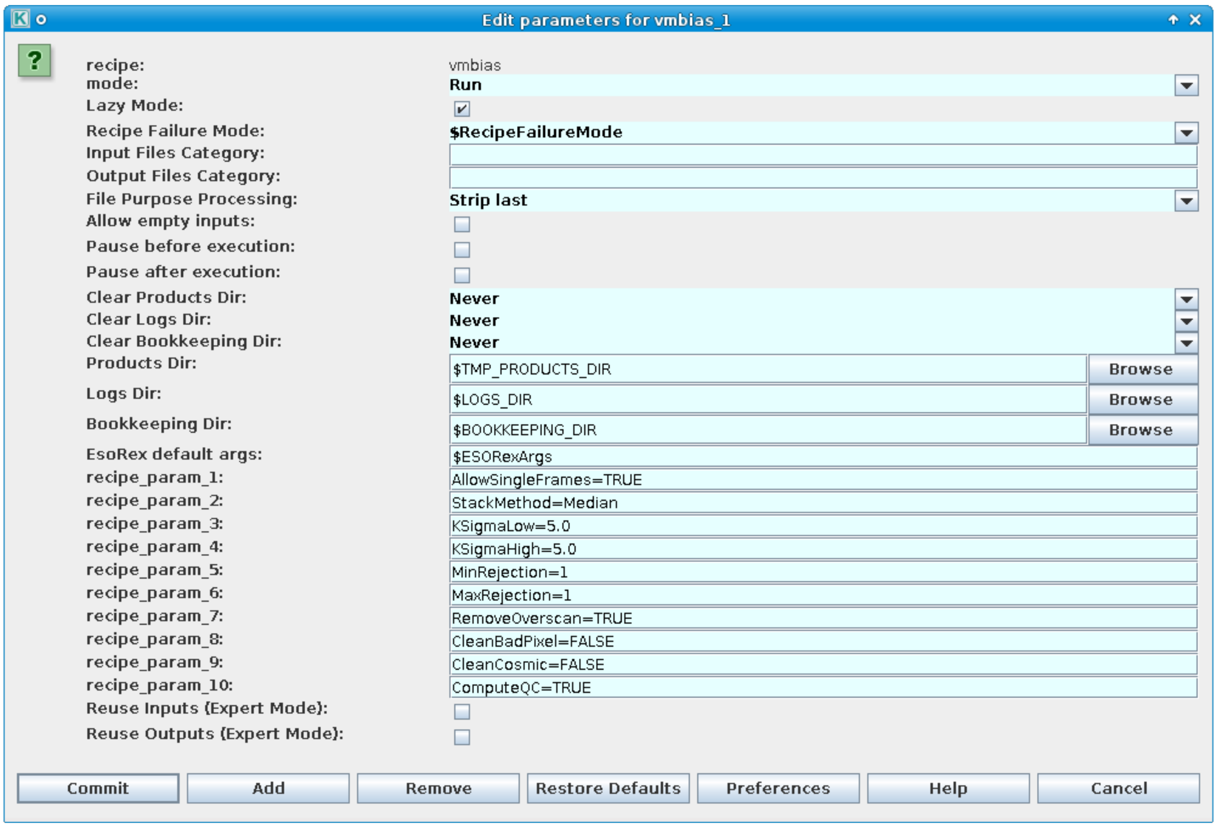
is used to communicate the number of different SoFs created by the `SofSplitter`. A workflow will only work as intended if the purpose of all the files a recipe needs as input is identical. The only exception to this rule is that a purpose can also be "default". In this case, the file is included in 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. 6.1).

In Figure 6.2 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 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 6.2.4). 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.



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

- 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. 6.2.4.
- 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.
- 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[7]).

[7] Available at *ftp://ftp.eso.org/pub/dfs/pipelines/vimos/vimos-pipeline-manual-6.8.pdf*

Table 6.1: 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

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.

6.2.4 Lazy Mode

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

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

The forced rereduction of data at each execution may of course be desirable. To force a rereduction of all data for all `RecipeExecuter` actors in the workflow (i.e. to disable Lazy mode for the whole workflow), set the `EraseDirs` parameter under the "Global Parameters" area of the workflow canvas to `true`. This will then remove all previous results as well. To force a rereduction of data for any single `RecipeExecuter` actor in the workflow (which will be inside the relevant composite actor), right-click the `RecipeExecuter` actor, select `Configure Actor`, and uncheck the Lazy mode parameter tick-box in the "Edit parameters" window that is displayed.

6.3 Workflow Steps

6.3.1 Step 1: Data Organisation And Selection

On clicking the ▶ button on the `Reflex` canvas, 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 6.1).

1. The `DataOrganiser` (DO) is the first crucial component of a Reflex workflow. The DO takes as input `RAWDATA_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.

   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 OCA[8] 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. A category could for example be `IFU_SCREEN_FLAT`, `IFU_ARC_SPECTRUM` or `IFU_SCIENCE`. 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. 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 `BIAS` and `IFU_CAL`. The former creates a master bias from raw biases, and the later creates (among other products) a master flat from raw flats. The `IFU_CAL` action needs raw lamp frames (arc and flat field) and the master bias (or a set of raw biases) as input. In this case, these biases will have the purpose `BIAS.IFU_CAL`. The same DataSet might also include biases with a different purpose, e.g. `BIAS.IFU_SCIENCE`. Irrespective of their purpose the file category for all these biases will be `BIAS`.

2. Next the `DataSet Chooser` displays the DataSets available in the "Select Data Sets" window[9], activating a vertical scroll bar on the right if necessary (see Figure 4.3). 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 `Select All` and `Deselect All` at the bottom left.

   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 and path for each file, the file category (from the FITS header), and a selection tick box in the right column (see Figure 6.3). 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 (see Figure 6.3), 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`, and to exit from the "Select DataSets"

[8]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`

[9]If you run the `Data Organiser` in Lazy Mode, changes in the `Keywords to be displayed` list will have no effect on the output shown in the `DataSet Chooser`.

window, click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow.

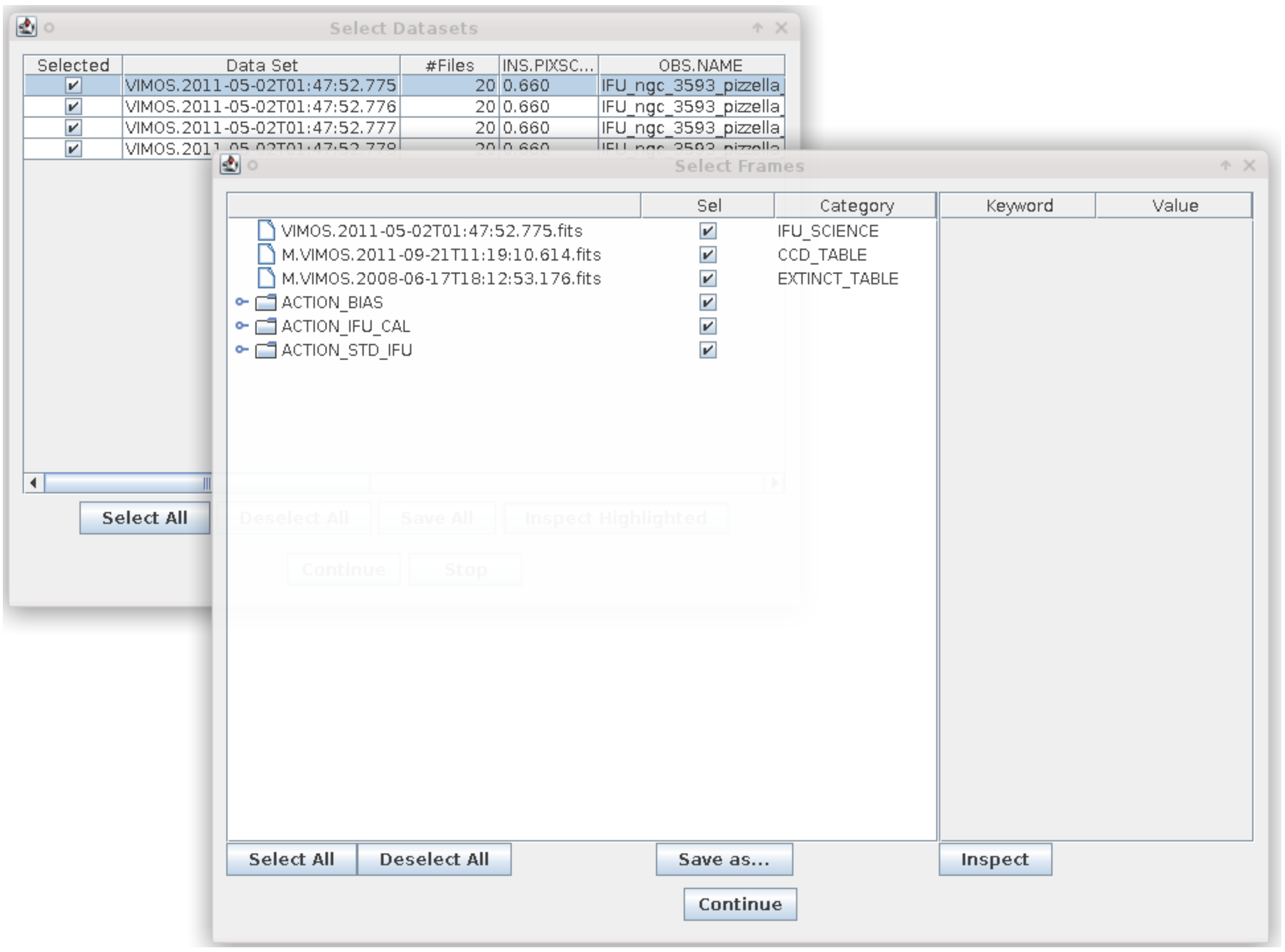
The categories and purposes of raw files are set by the DO, whereas the categories and purpose of products generated by recipes are set by the `RecipeExecuter` (see Sect. 6.2.3). The file categories are used by the `FitsRouter` to send files to particular processing steps or branches of the workflow (see below). The purpose is used by the `SofSplitter` to generate input SoFs for the `RecipeExecuter` and the results are collected by the `SofAccumulator`. Note that while the DO includes files into a DataSet for a reason, and records this reason as the "purpose" of the file, the workflow itself can use these files in a different manner. 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 Step 2: 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 6.2.

6.3.3 Step 3: 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.



Select Datasets

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

Select Frames

	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	✓	

Figure 6.3: *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.*

7 Frequently Asked Questions

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas) and organised further in directories by pipeline recipe.
- **Can I use different sets of bias frames to calibrate my flat frames and science data?** Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it (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 launch `Reflex` from the command line?** Yes, use the command:

  ```
  reflex -runwf -nocache -nogui <workflow_path>/<workflow>.xml
  ```

  Note that this mode is not fully supported, and the user should be aware of two points. Firstly, the execution prompt is not returned after the workflow finishes, and therefore `Reflex` must be manually killed. Secondly, all the interactive windows will still appear (if activated in the workflow), so it is not suitable for batch processing.
- **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 run manually the recipes executed by Reflex?** If a user wants to re-run a recipe on the command line he/she has to go to the appropriate reflex_book_keeping directory, which is generally reflex_book_keeping/VIMOS/<recipe_name>_<number> (for instance reflex_book_keeping/VIMOS/bias_1/). There, subdirectories exist with the time stamp of the recipe execution (e.g. 2013-01-25T12:33:53.926/). If the user wants to re-execute the most recent processing he/she should go to the `latest` directory and then execute the script `cmdline.txt`. Alternatively, to ensure that the path to `esorex` is the correct one, the user can execute `ESOREX_CONFIG="REFLEX_INST/etc/esorex.rc REFLEX_INST/bin/eso` `-recipe-config=<recipe>.rc <recipe> data.sof`, where REFLEX_INST is the directory where Reflex and the pipelines were installed. If the user knows the name of the input raw files

for the recipe, the correct directory among the many time stamps can be found via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as before. The products will appear in the directory from which the recipe is called, and not in the reflex_tmp_products or reflex_end_products directory, and they will not be renamed.

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 [10] 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).

[10] `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).

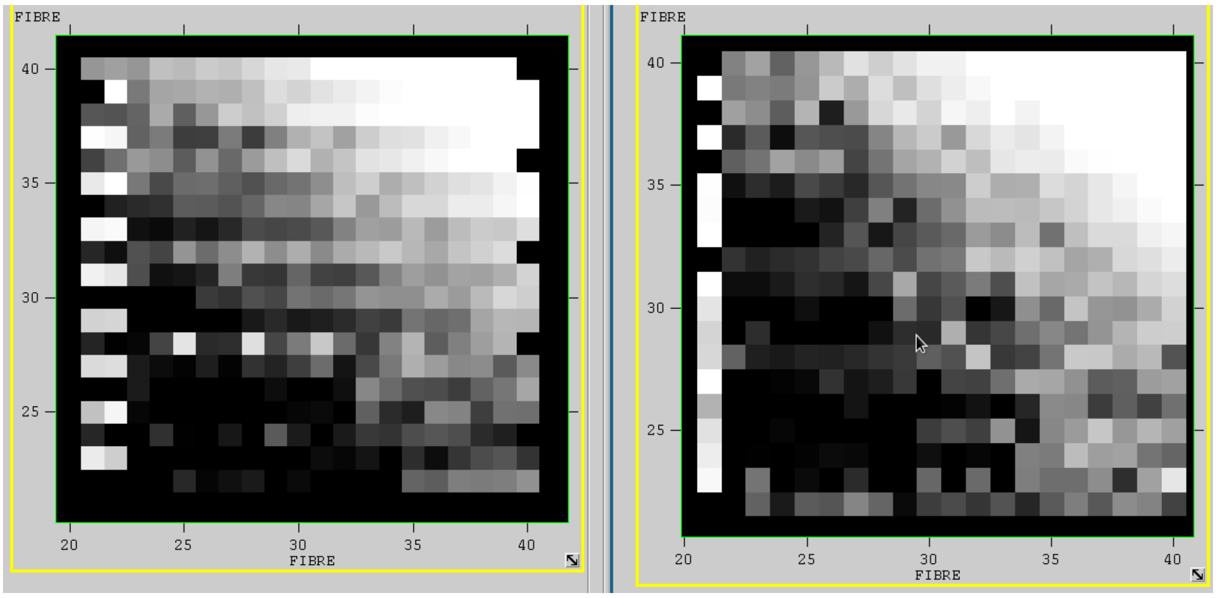


Figure 8.1: Example of field of view where fibre misidentification shuffled horizontally the fibre position (left panel), compared to one where fibre identification worked properly (right panel). The field of view refers only to quadrant 3.

- 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` [11]. 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

[11] 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.

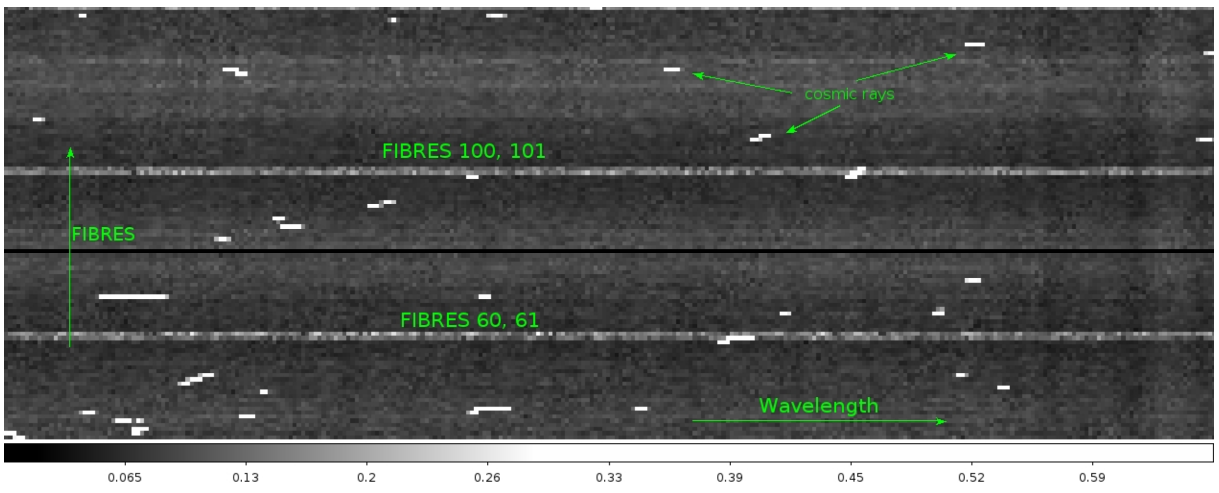


Figure 8.2: Example of 2D spectra where the marked fibres (60-61, and 100-101) are properly identified.

general trend, or because the signal level is too low. When the percentage of rejected positions is more 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 6.2.

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[12] 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.

[12] `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).

Forchì V., 2012, Reflex User Manual, VLT-MAN-ESO-19000-5037, Issue 0.7, *ftp://ftp.eso.org/pub/dfs/reflex/ReflexUserManual-3.1.pdf* 13, 17, 21

ESO VIMOS Pipeline Team, VIMOS Pipeline User Manual, VLT-MAN-ESO-19500-3355, Issue 6.6 16