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 UVES Tutorial

VLT-MAN-ESO-19540-5296

Issue 6.6.1

Date 2026-05-05

	Name	Date	Signature
Prepared:	ESO UVES Pipeline Team	2026-05-05	
Approved:	W.Freudling		
Released:	M. Sterzik		

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.1	15/01/2011	All	Public release
2.0	26/04/2011	Updated details	Update
3.0	05/04/2012	All	Updated for Reflex 2.0
5.0	22/05/2012	All	Updated for Reflex 2.1
5.1	15/10/2012	All	Updated for Reflex 2.2
5.2	21/12/2012	All	Updated for Reflex 2.3
5.3	06/05/2013	All	Updated for Reflex 2.4
5.4	15/11/2013	All	Updated for Reflex 2.5
6.1.3	03/06/2020	All	Updated for Reflex 2.10
6.1.6	20/05/2021	All	Public release 6.1.6
6.1.8	19/05/2022	All	Public release 6.1.8
6.6.1	05/05/2026	All	Public release 6.6.1

This page was intentionally left blank

Contents

1 Introduction to `Esoreflex` **7**

2 Software Installation **8**

2.1 Installing `Esoreflex` workflows via `macports` . . . 8

2.2 Installing `Esoreflex` workflows via `rpm/yum/dnf` . . . 8

2.3 Installing `Esoreflex` workflows via `install_esoreflex` . . . 8

2.4 Demo Data . . . 9

3 Quick Start: Reducing The Demo Data **10**

4 About the main `esoreflex` canvas **14**

4.1 Saving And Loading Workflows . . . 14

4.2 Buttons . . . 14

4.3 Workflow States . . . 14

5 The UVES Workflow **15**

5.1 Workflow Canvas Parameters . . . 15

5.2 Workflow Actors . . . 16

5.2.1 Simple Actors . . . 16

5.2.2 Composite Actors . . . 16

5.2.3 Recipe Execution within Composite Actors . . . 18

5.2.4 Lazy Mode . . . 21

6 Reducing your own data **23**

6.1 The `esoreflex` command . . . 23

6.2 Launching the workflow . . . 23

6.3 Workflow Steps . . . 25

6.3.1 Data Organisation And Selection . . . 25

6.3.2 DataSetChooser . . . 26

6.3.3 The ProductExplorer . . . 27

6.3.4 Creation Of Master Calibration Files . . . 29

6.3.5 Wavelength And Response Calibration 30

6.3.6 Spectrum Extraction 30

6.3.7 Output Organisation 31

7 Optimising Your Results Through Workflow Interaction 35

8 Frequently Asked Questions 46

8.1 UVES Specific Questions 48

9 Troubleshooting 51

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 **blue and red arm UVES echelle data for point source observations only** via the UVES `Reflex` workflow. The user is referred[2] to the UVES user manual (Kaufer et al. 2010) for more information on the instrument itself, and the UVES pipeline user manual (Larsen et al. 2012) for the details of UVES pipeline recipes. The quick start section (see Section 3) describes the minimum effort to get started, and it makes up only two pages of this tutorial.

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

[2] These documents are available from *http://www.eso.org/sci/facilities/paranal/instruments/uves/doc/index.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` UVES workflow without any changes in parameters. This way you have a data set to experiment with before you start to work on your own data.

Note that you will need a minimum of ∼1.2 GB, ∼0.9 GB and ∼2.3 GB of free disk space for the directories `<download_dir>`, `<install_dir>` and `<data_dir>`, respectively, at least ∼2.0 GB of total memory (physical + swap), and a screen with a resolution better than 1024x800 pixels .

The UVES demo data include five blue arm science observations, and three red arm science observations, covering a range of signal-to-noise and object types. In addition we provide one incomplete `RED` data set taken with central wavelength 760 nm to illustrate how an incomplete DataSet looks in the "Select DataSets" window (see Fig. 3.3).

3 Quick Start: Reducing The Demo Data

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the UVES 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 UVES Echelle by typing:

   ```
   esoreflex uves&
   ```

   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 UVES Echelle 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 6.3.3.

10. After the workflow has finished, all the products from all the datasets can be found in a directory under END_PRODUCTS_DIR named after the workflow start timestamp. Further subdirectories will be found with the name of each dataset.

Well done! You have successfully completed the quick start section and you should be able to use this knowledge to reduce your own data. However, there are many interesting features of Reflex and the UVES workflow that merit a look at the rest of this tutorial.

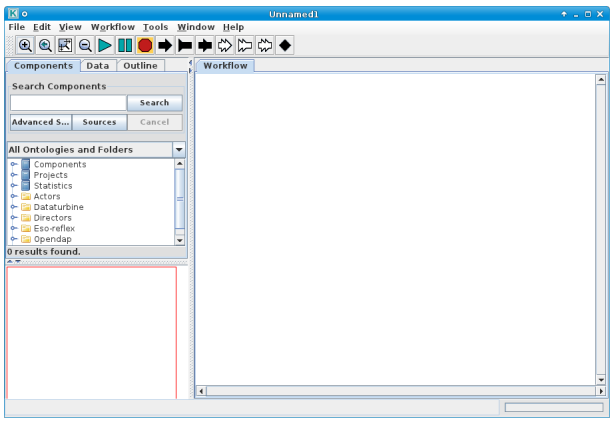


Figure 3.1: *The empty* Reflex *canvas.*

[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.

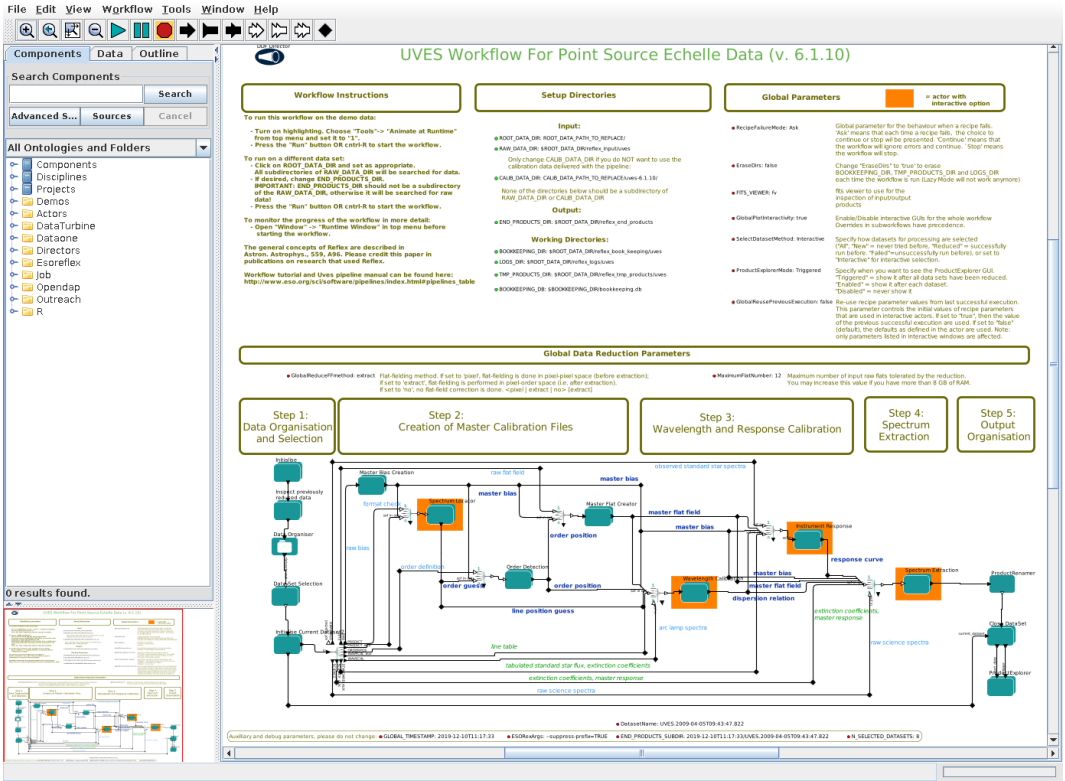


Figure 3.2: *UVES workflow general layout.*

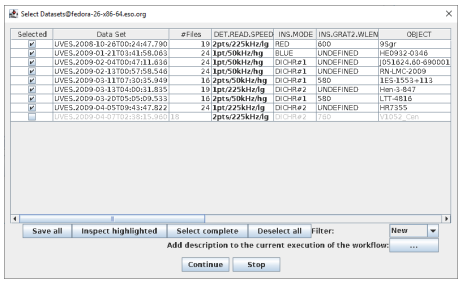


Figure 3.3: *The "Select Datasets" pop-up window. On purpose the demo includes an incomplete (grey) DataSet.*

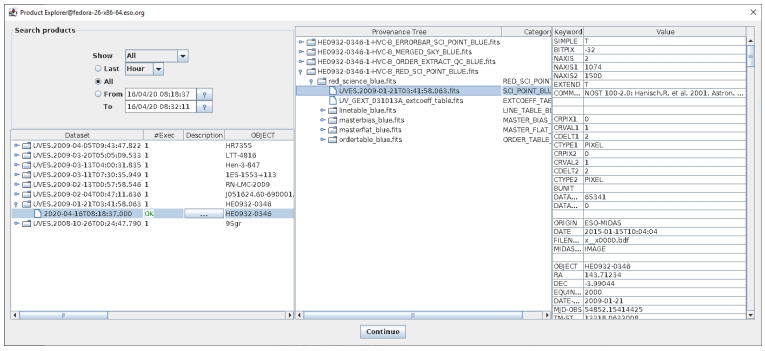


Figure 3.4: *The Product Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products.*

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 UVES Workflow

The UVES 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 `GlobalPlotInteractivity` controls whether the interactive windows will appear for those windows which are *enabled* by default. The possible values are `true, false`. Take into account that some

windows are disabled in the default configuration and therefore are not affected by this parameter.

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

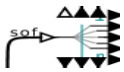
5.2 Workflow Actors

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.
-

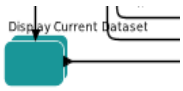
- The `ProductExplorer` actor (inside a composite actor).

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. An orange box marks an actor with an associated interactive window. In the UVES workflow, the following actors are composite actors:

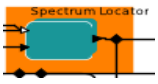
-

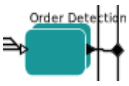
- The `Initialise` actor.
- 

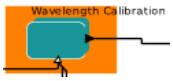
- The `Display Current DataSet` actor.
-

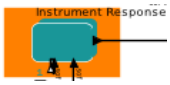
- The `Master Bias Creation` actor.
-

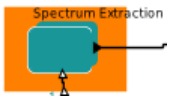
- The `Master Dark Creation` actor.
-

- The `Master Flat Creation` actor.
- 

- The `Spectrum Locator` actor.
- 

- The `Order Detection` actor.
- 

- The `Wavelength Calibration` actor.
- 

- The `Instrument Response` actor.
- 

- The `Spectrum Extraction` actor.
-

- The `Close DataSet` actor.

Access to the parameters for a composite actor is achieved by right-clicking on the actor and selecting `Configure Actor`. This will open an "Edit parameters" window (see Figure 5.1).

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

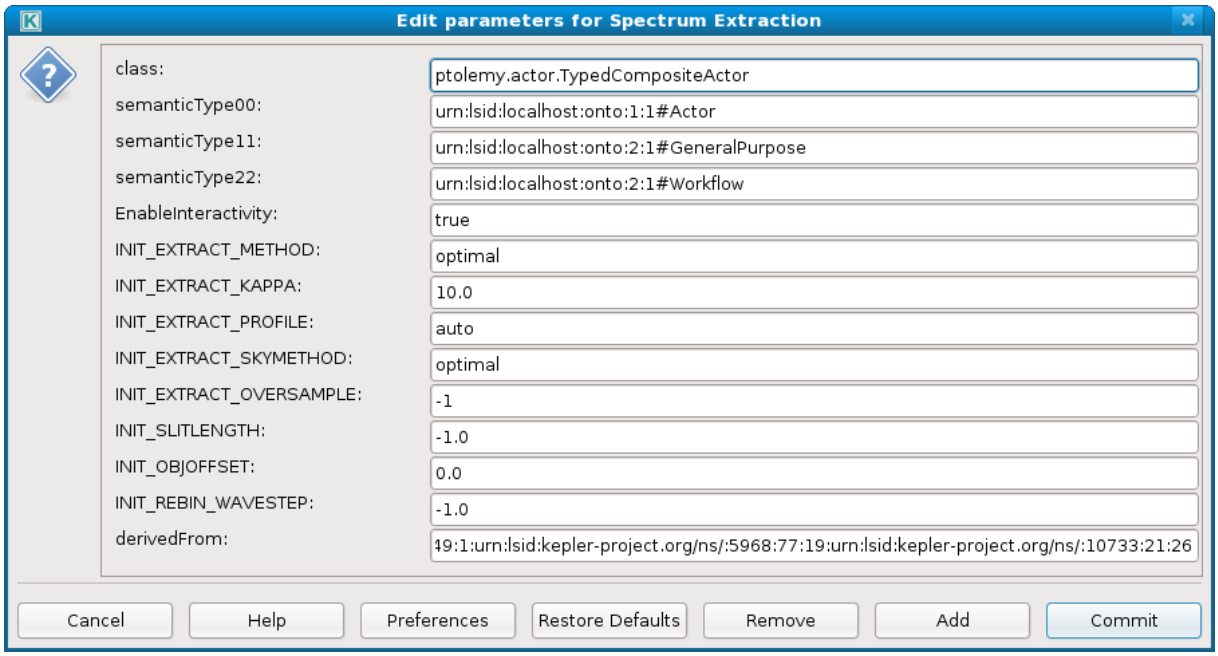


Figure 5.1: *The "Edit parameters" window for a typical composite actor..*

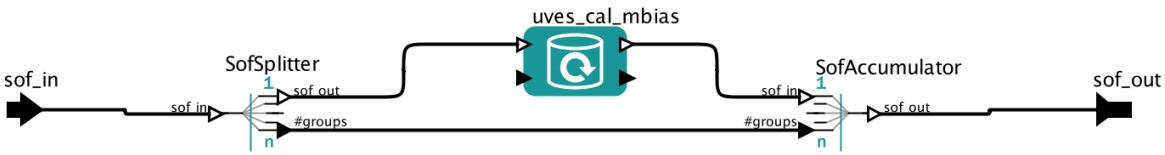


Figure 5.2: *This is the window you get when you choose* `Open Actor` *for the composite actor* `Master Bias Creation`*. This is the most simple case of a composite actor in the UVES workflow.*

5.2.3 Recipe Execution within Composite Actors

The UVES workflow contains composite actors to run the pipeline recipes. This is in the most simple case due to the `SoF Splitter/Accumulator`[4], which allows to process calibration data from different settings within one given DataSet (e.g. bias frames with different binnings, dark frames with different DIT, etc.). More complex composite actors contain several actors.

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 Figure 5.2, which is the most simple version of a composite actor in the UVES workflow. The `RecipeExecuter` is preceded by an `SofSplitter`, and followed by an `SofAccumulator`. The function of the `SofSplitter` is to investigate

[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_UVB`)

Table 5.1: The UVES pipeline actors and their contents

actor	recipes	description
Master Bias Creation	`uves_cal_mbias`	create master bias
Master Dark Creation	`uves_cal_mdark`	create master dark
Spectrum Locator	`uves_cal_predict`	create line and order guess tables
Order Detection	`uves_cal_orderpos`	determine centers of orders
Master Flat Creation	`uves_cal_mflat`	create master flat
Wavelength Calibration	`uves_cal_wavecal`	determine wavelength solution
Instrument Response	`uves_cal_response`	determine response function
Spectrum Extraction	`uves_obs_scired`	reduce science data

the incoming SoFs, sort them by "purpose", and create separate SoFs for each purpose. The `RecipeExecuter` then processes each of the SoFs independently. 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 bias frames. The different bias frames are to be used to de-bias different frames, e.g. the science frames, the various arc lamp and flat field frames and the standard star frames. The `Data Organiser` determines and records this "purpose" of each bias, and this information is included in the DataSet and each SoF created from this DataSet. The `FitsRouter` directs all raw biases to the `Master Bias Creation` composite actor. The `SofSplitter` then creates SoFs, one for the biases to be used for the science images, and (possibly) separate ones for the biases to be used for the calibration images and flat fields. The `uves_cal_mbias` recipe creates one master bias for each SoF, and the `SofAccumulator` then creates a SoF that contains all the master biases. Table 5.2.3 shows which actors contain which recipes and what they do.

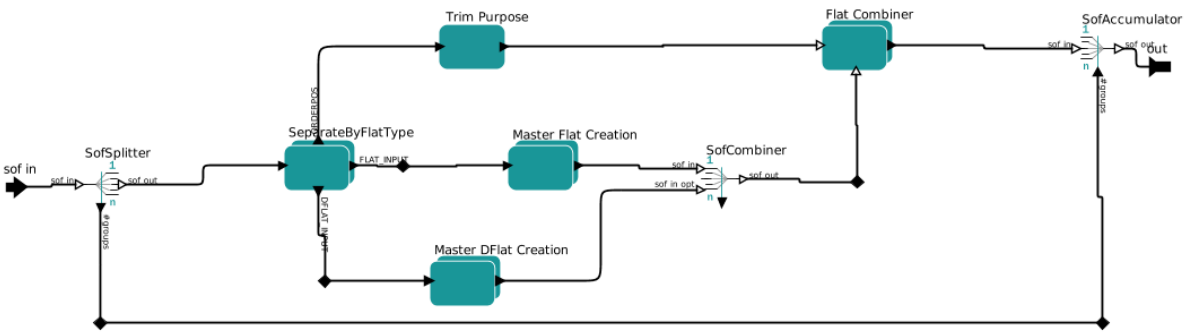


Figure 5.3: *The window you get when you choose* `Open Actor` *for the composite actor* `Master Flat Creation`. *Using* `Open Actor` *on* `Master Flat Creation` *gives you Figure 5.4.*

A `RecipeExecuter` actor is used in the workflow to run a single UVES pipeline recipe (e.g: the "Master Bias

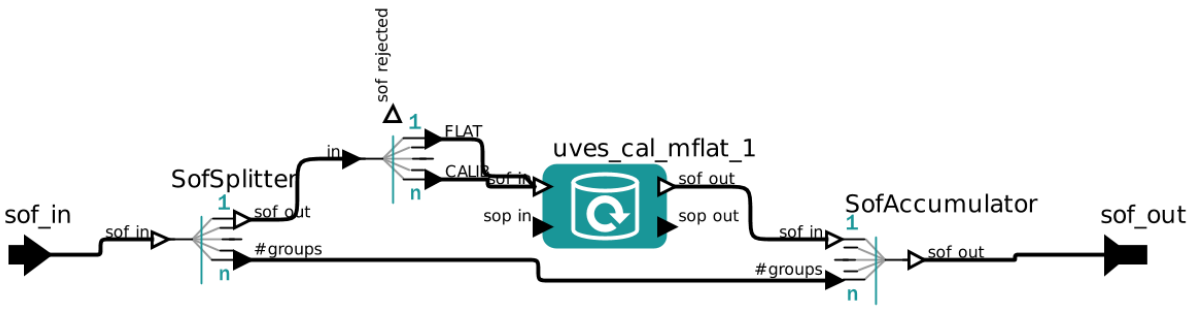


Figure 5.4: *This is the window you get when you choose* `Open Actor` *for the composite actor* `Master Flat Creation` *within* `Master Flat Creator`.

Creation" actor runs the `uves_cal_mbias` pipeline recipe). In order to configure a `RecipeExecuter`, one has to first use `Open Actor` (possibly repeatedly) to get to the level of the `RecipeExecuter` actors (see Figures 5.3 & 5.4).

In Figure 5.5 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 UVES 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 the next section.
- 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 is the default option.
- The set of parameters, which start with "recipe_param_" and end with a number, corresponds to the parameters of the relevant UVES pipeline recipe. By default in the `RecipeExecuter` actor, the pipeline

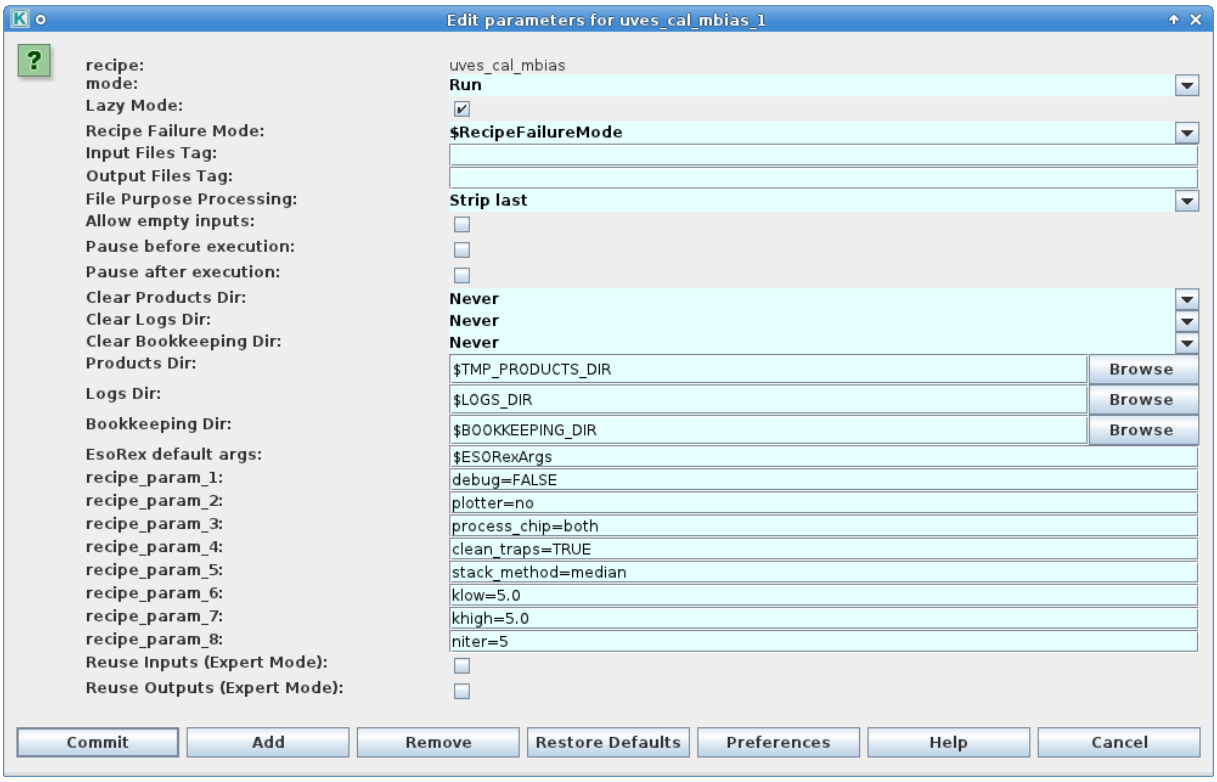


Figure 5.5: *The "Edit parameters" window for a typical* `RecipeExecuter` *actor, the* `uves_cal_mbias` *actor which runs the* `uves_cal_mbias` *pipeline recipe.*

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 UVES pipeline recipe parameters, the user should refer to the UVES pipeline user manual (Larsen 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 may be saved in the workflow by clicking the `Commit` button when you have finished.

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.

[5] Available from http://www.eso.org/sci/facilities/paranal/instruments/uves/doc/index.html

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

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

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

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>` UVES Echelle 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
  uves
```

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 UVES Echelle. 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 uves
   ```

   to launch the Product Explorer. The Product Explorer allows you to inspect the data products already reduced by the UVES Echelle 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> uves
   ```

   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 6.3.3

3. Type:

   ```
   esoreflex uves &
   ```

   to launch the UVES Echelle esoreflex workflow. The UVES Echelle 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 6.3.3 for how to configure this option).

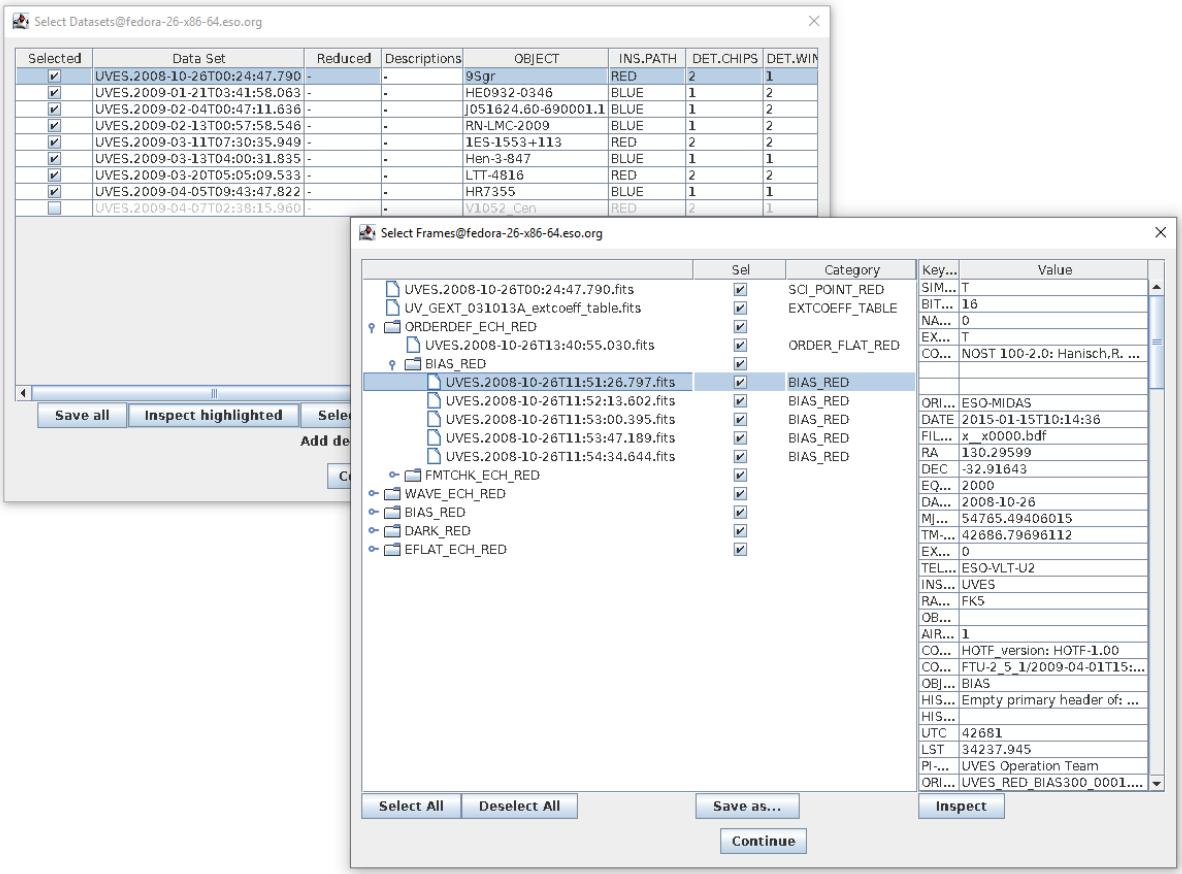


Figure 6.1: *The "Select Frames" window with a single file from the current DataSet highlighted in blue, and the corresponding FITS header displayed in the text box on the right. Hidden partially behind the "Select Frames" window is the "Select Datasets" window with the currently selected DataSet highlighted in blue.*

6.3 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.

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

[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`

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.

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.

6.3.3 The ProductExplorer

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

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

- 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.

6.3.4 Creation Of Master Calibration Files

In this step of the workflow, the following UVES recipes are executed in the order listed below. Please refer to the UVES pipeline user manual (Larsen et al. 2012: Sections 9, 10 & 11) for the details of each recipe and the algorithms employed:

1. The `Master Bias Creation` actor will execute the UVES pipeline recipe `uves_cal_mbias` in order to create a combined master bias frame from the set of raw bias frames.
2. The `Master Dark Creation` actor will execute the UVES pipeline recipe `uves_cal_mdark` in order to create a combined master dark frame from the set of raw dark frames. Note that this actor will be skipped if there are no raw dark frames in the current DataSet.
3. The `Spectrum Locator` actor will execute the UVES pipeline recipe `uves_cal_predict` in order to compute initial guesses for the wavelength solution and order positions by employing a physical model of UVES combined with the information on the atmospheric pressure, temperature, and corresponding instrument setting stored in the FITS header of the input arc-lamp frame.
4. The `Order Detection` actor will execute the UVES pipeline recipe `uves_cal_orderpos` in order to fully detect and define the order positions on the detector from an input flat-field frame.
5. The `Master Flat Creation` actor will execute the UVES pipeline recipe `uves_cal_mflat` in order to create a combined master flat frame from the set of raw flat frames.

   Note that the workflow supports the processing of deuterium lamp flats (`DPR TYPE = LAMP,DFLAT`) in addition to the usual lamp flats (`DPR TYPE = LAMP,FLAT`). The deuterium lamp flats are recommended for the spectral region shortwards of 350 nm and they are taken as supplementary calibrations for the UVES blue arm central wavelength setting of 346 nm. If deuterium flats are present in the DataSet, then they will be combined with the usual lamp flats into a combined master flat that provides much better S/N shortwards of 350 nm.

6.3.5 Wavelength And Response Calibration

In this step of the workflow, the following UVES recipes are executed in the order listed below. Please refer to the UVES pipeline user manual (Larsen et al. 2012: Sections 9, 10 & 11) for the details of each recipe and the algorithms employed:

1. The `Wavelength Calibration` actor will execute the UVES pipeline recipe `uves_cal_wavecal` in order to create a wavelength calibration solution from an input arc-lamp frame.
2. The `Instrument Response` actor will execute the UVES pipeline recipe `uves_cal_response` in order to create an instrument response curve from the observation of a standard star, which will subsequently be used to flux-calibrate the science observation. Note that this actor will be skipped if there are no observations of a standard star in the current DataSet.

Generally, UVES data are supplied with master instrument response curves which will also be included in the DataSets for a science observation. The UVES workflow will flux-calibrate the science observation using the instrument response curve derived from the standard star observation if it exists in the current DataSet, and failing this, the master instrument response curve will be used if it exists in the current DataSet. If neither a standard star observation nor a master instrument response curve exist in the current DataSet, then the science observation will not be flux-calibrated.

6.3.6 Spectrum Extraction

In this step of the workflow, the `Spectrum Extraction` actor will execute the UVES pipeline recipe `uves_obs_scired` in order to extract and merge the science spectrum from the science observation.

The object spectrum is extracted using either `optimal` extraction (default), `linear` extraction (summing the flux within the extraction slit), `average` extraction (averaging the flux within the extraction slit) or the `2d` method (resampling the full slit length to a 2-dimensional image, with wavelength along the x-axis and position along the slit along the y-axis, best suited for **extended sources**). More details can be found in the UVES pipeline user manual (Larsen et al. 2012, Sections 10 and 11).

If the spectrum is optimally extracted, cosmic rays are detected and rejected, and the sky is fitted and subtracted during the extraction. If linear/average extraction is used, then the sky is extracted in windows above and below the object and subsequently subtracted (if `reduce.skysub` is set to TRUE (default)).

The default values of the recipe parameters are optimized towards the extraction of the spectrum of one point source centered on the slit. The recipe parameters `reduce.slitlength`, `reduce.objoffset`, and `reduce.objslit` allow the user to fine-tune the extraction for other situations. The usage of the parameters depends on the selected extraction mode in a complex way, as described below.

By default (`reduce.slitlength` = -1) the full slit length for the extraction is obtained from the header keyword `HIERARCH ESO INS SLIT2 LEN` (blue arm) or `HIERARCH ESO INS SLIT3 LEN` (red arm) and the full slit is centered on the center of the order. **Warning** If the value specified for `reduce.slitlength` exceeds the value derived from the header keywords inter-order regions may be included in the extraction.

In **average/linear** extraction:

`reduce.slitlength` specifies the full slit length within which one object window and up to two sky windows will be defined

`reduce.objoffset` [`ignored if reduce.objslit <0 (default −1)`] allows the user to offset the object window away from the order center

`reduce.objslit` if >0 `reduce.objslit` specifies the size of the object window, which can then be offset from the order center via `reduce.objoffset`; if ≤0 (default), the object window size will be 1/2 of the full slit, and its position will be defined by searching the object's spectrum within the slit

In **optimal** extraction the full slit is used as extraction slit:

`reduce.slitlength` specifies the full slit length within which object and sky spectra will be extracted simultaneously

`reduce.objoff` allows the user to offset the *full slit* (= extraction slit) as specified by `reduce.slitlength`

`reduce.objslit` completely ignored, because the full slit is used as extraction slit

Key differences between the three modes are

1. In average/linear extraction the full slit is always centered on the order center, but in optimal extraction it can be offset away from the order center by the `reduce.objoffset` parameter
2. Optimal extraction always uses `reduce.objoffset` but completely ignores `reduce.objslit`, whereas average/linear extraction uses `reduce.objoffset` only if `reduce.objslit` is >0
3. In average/linear extraction, once a portion of the full slit has been reserved by the object window as described above, sky windows of size 0.5·(`slitlength−objslit`)±`objoffset` will be carved out of the remaining full slit (if any). This procedure is described in detail in the description of `uves_obs_scired` in Sect. 11 of the UVES pipeline user manual (Larsen et al. 2012).

If **more than one object trace is present** in the slit optimal extraction can be used with a suitable combination of `reduce.slitlength` and `reduce.objoffset` to adapt the slit length and position to cover only one trace and the adjacent sky regions. Obviously the data have to be processed separately for each trace. It is not possible to extract multiple traces with Linear or Average extraction.

Finally, as described in the previous section, the flux-calibration of the science spectrum will be carried out if an appropriate instrument response curve (master or otherwise) has been used as an input to this actor.

After the spectrum extraction, the actor will display an interactive window to allow the user to inspect the extracted spectrum and assess the extraction quality (see Figure 6.2).

6.3.7 Output Organisation

After having processed the input data for a DataSet, the `Data Filter` actor allows the user to inspect the results and select which files to save. By default this actor is set to skip and all products will be saved. Next the

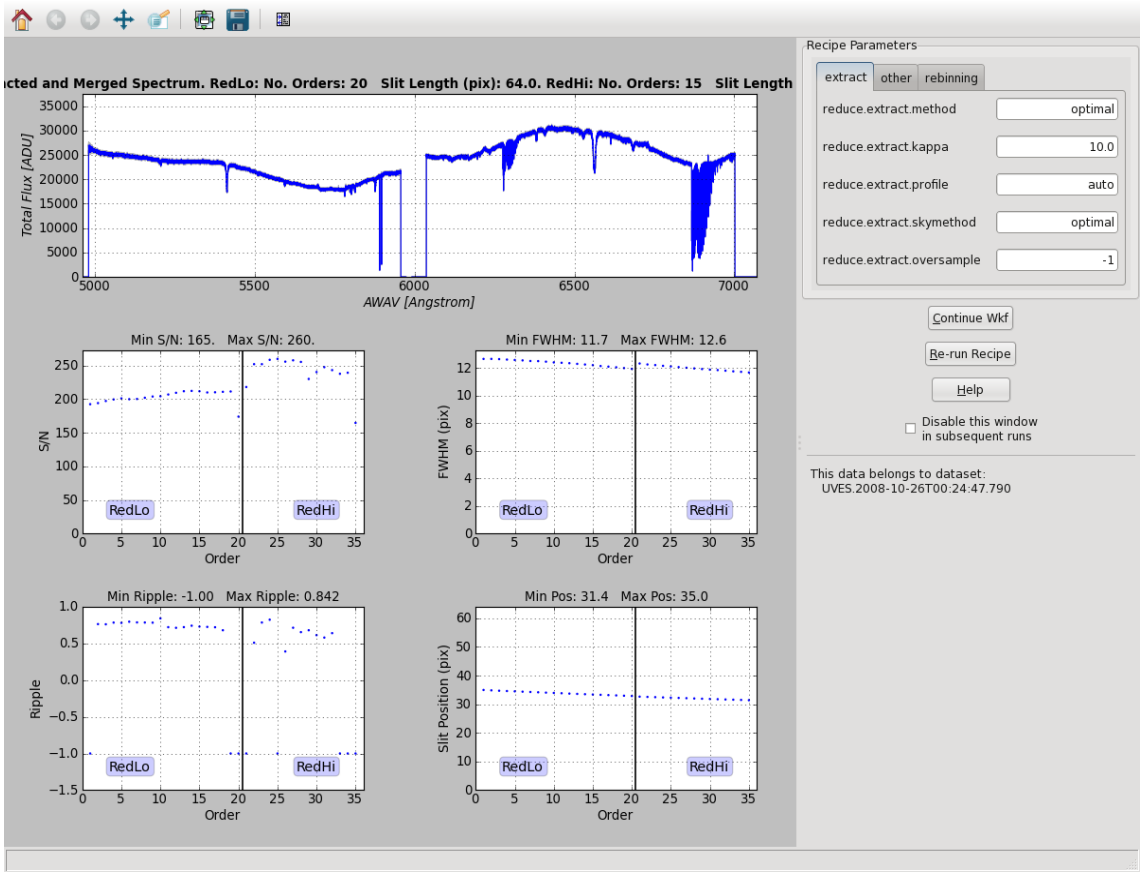


Figure 6.2: *The interactive pop-up window for the* `Spectrum Extraction` *actor and UVES pipeline recipe* `uves_obs_scired`*. The extracted and merged spectrum for the first DataSet is displayed in the top panel, and the flux-calibrated spectrum is displayed in the panel below.*

workflow highlights and executes the `Product Renamer` actor, which, by default, will copy the defined final products of the UVES pipeline recipe `uves_obs_scired` to the directory specified by `END_PRODUCTS_DIR` and rename them with names derived from the values of certain FITS header keywords. Specifically, final products are renamed by default with names of the form:

```
<HIERARCH.ESO.OBS.NAME>_<HIERARCH.ESO.PRO.CATG>.fits
```

where `<HIERARCH.ESO.OBS.NAME>` and `<HIERARCH.ESO.PRO.CATG>` represent the values of the corresponding FITS header keywords. These names are fully configurable by right-clicking on the `Product Renamer` actor and selecting `Configure Actor`, and then editing the `Rename_keywords` parameter as appropriate.

The final products that are copied and renamed are:

- `<HIERARCH.ESO.OBS.NAME>_RED_SCI_POINT_<ARM>.fits` - The extracted and merged science spectrum.

- `<HIERARCH.ESO.OBS.NAME>_ERRORBAR_SCI_POINT_<ARM>.fits` - The uncertainty on the extracted and merged science spectrum.
- `<HIERARCH.ESO.OBS.NAME>_FLUXCAL_SCI_POINT_<ARM>.fits` - The extracted, merged, and flux-calibrated science spectrum. This product is only generated if an appropriate instrument response curve (master or otherwise) was used as an input to the `Spectrum Extraction` actor.
- `<HIERARCH.ESO.OBS.NAME>_FLUXCAL_ERRORBAR_SCI_POINT_<ARM>.fits` - The uncertainty on the extracted, merged, and flux-calibrated science spectrum. This product is only generated if an appropriate instrument response curve (master or otherwise) was used as an input to the `Spectrum Extraction` actor.
- `<HIERARCH.ESO.OBS.NAME>_ORDER_EXTRACT_QC_<ARM>.fits` - A table of useful statistics from the spectrum extraction procedure which are used to make the plots in the interactive window for the `Spectrum Extraction` actor (see Figure 6.2).
- `<HIERARCH.ESO.OBS.NAME>_MERGED_SKY_<ARM>.fits` - The extracted and merged sky spectrum.

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

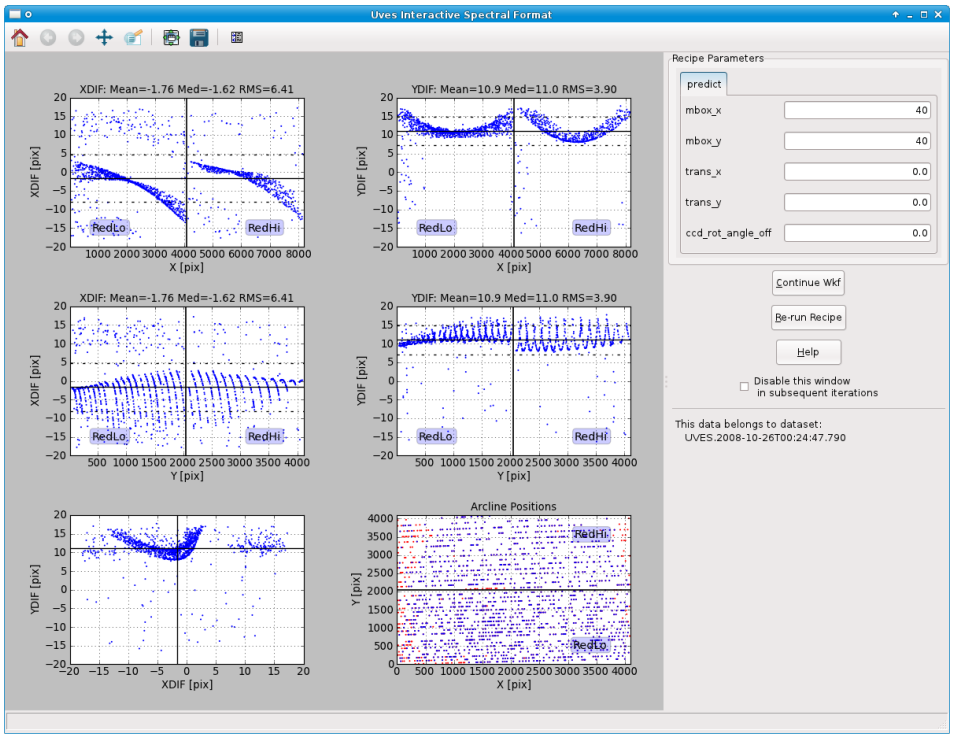


Figure 6.3: *The interactive window for the* `Spectrum Locator` *actor for the first DataSet in the data supplied with* `Reflex 2.11.5`..

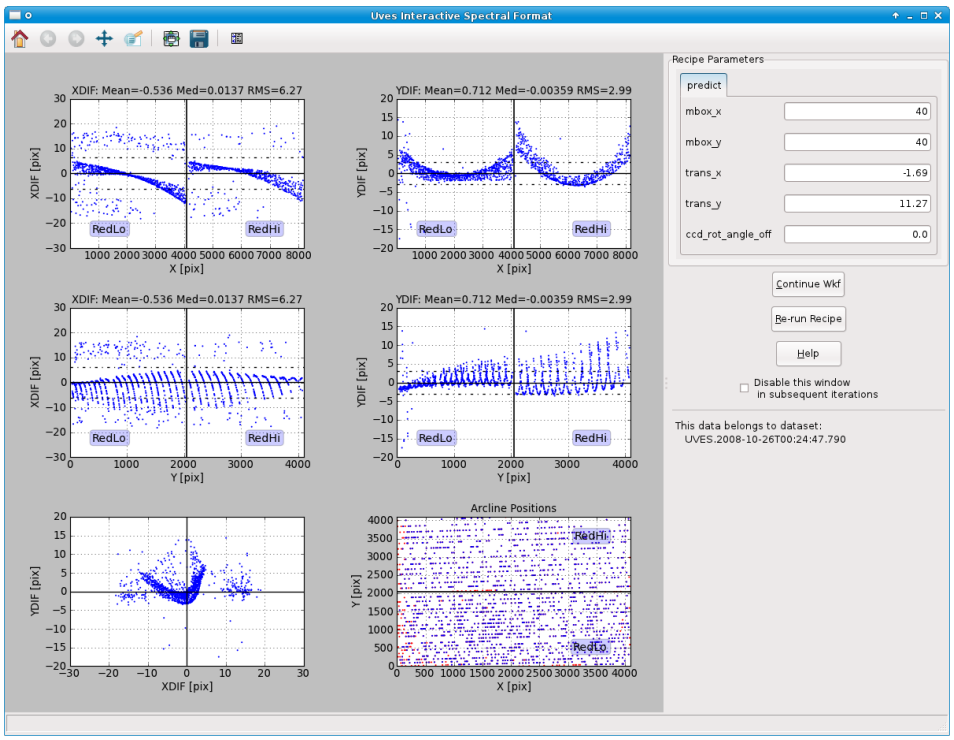


Figure 6.4: *Same as Figure 6.3 but after iteration of the corresponding* `uves_cal_predict` *pipeline recipe.*

7 Optimising Your Results Through Workflow Interaction

In this section, we use the information from previous sections along with the UVES demo data supplied with `Reflex 2.11.5` to illustrate how to optimise the scientific products in terms of quality and S/N. This is achieved by interaction with the workflow actors via interactive windows displayed at key data reduction points in the data flow, which enable iteration of certain recipes in order to obtain better results. For instance, the physical model prediction calculated during execution of the `Spectrum Locator` actor may be inaccurate and need correcting, or the wavelength solution may need tweaking, or the user may want to clip more cosmic rays from the extracted spectrum.

We recommend that the user has already carried out the reductions for all the data sets as described in Section 3, although this is not a pre-requisite to following this section. By doing this, the user will be taking advantage of the workflow Lazy mode, with minimal waiting time between various pipeline recipe executions.

Please notice that the numbers given in the rest of the section were obtained in a Fedora 15, 64 bit machine. Your values may differ slightly depending on you operating system and platform.

Please follow these steps in order to optimise the reductions for the UVES demo data supplied with `Reflex 2.11.5`:

1. Carry out the first four steps described in the quick start Section 3.

2. In the UVES workflow, the four interactive actors (`Spectrum Locator`, `Wavelength Calibration`, `Instrument Response`, `Spectrum Extraction`) are identifiable by an orange rectangle encompassing the actor name. For all of these composite actors except the `Spectrum Extraction` actor, you will need to enable their interactive mode. Do this by double-clicking on the composite actor, setting the "EnableInteractivity" parameter to `true`, and clicking `Commit` to save the changes to the workflow.

3. Click the button.

4. Click the `Continue` button in the "Select Datasets" window.

5. A "Current DataSet" window pops up that you can minimise.

6. `Spectrum Locator:` During the execution of the `Spectrum Locator` actor, an interactive window will appear as shown in Figure 6.3. The interactive window consists of six plots, five pipeline recipe parameters that may be changed, three buttons on the right, and a panel of buttons at the top-left. Help for each of these features may be accessed by leaving the mouse pointer for about one second over the plot/parameter/button for which information is required. Note that the recipe parameters are grouped via labels at the top of the "Recipe Parameters" box that may be left-clicked on to reveal the associated parameters.

   The `Spectrum Locator` actor executes the UVES pipeline recipe `uves_cal_predict`, taking as input a format-check arc-lamp frame. The purpose of the recipe is to compute an initial guess for the wavelength solution and order positions by employing a physical model of UVES combined with the information on the atmospheric pressure, temperature, and corresponding instrument setting in the FITS header of the input arc-lamp frame. For quality control purposes, this initial guess wavelength solution

may be compared with the actual detected line positions (shown in the bottom-right plot), which also allows the identification of cases where the physical model prediction is inaccurate.

The plots in the interactive window provide such a comparison by showing the differences `XDIF` and `YDIF` between the predicted and detected line positions along the x and y image axes, respectively, as functions of x, y and each other. If the physical model prediction is good, then these plots will show clear trends with sets of scattered outliers, and the statistics shown in the plot titles should report median values for `XDIF` and `YDIF` that are in the range $\pm$ 10 pix. If the physical model prediction is poor, then these plots will show only scattered points with no trends, although this happens very rarely (e.g. for data taken after an earthquake event).

7. The panel of buttons at the top-left of the interactive window may be used to manipulate the displayed plots. The buttons have the following functions:

   - - Reset all the plot ranges to their original values.
   - - Undo the last modification of the plot ranges (if possible).
   - - Redo the next modification of the plot ranges (if possible).
   - - Selecting this button allows the user to use the mouse to shift the plot ranges by left-clicking on the target plot canvas and then dragging the mouse around while keeping the left mouse button held down, and releasing when ready.
   - - Selecting this button allows the user to zoom in on each plot by left-clicking on the target plot canvas to mark the top-left corner of a rectangle and then dragging the mouse to the bottom-right corner of the rectangle and releasing. The plot ranges will then be modified to match the rectangle that was defined. The following constraints can be made by holding simultaneously a key while pressing the left mouse button:

Constrain pan/zoom to x axis	hold X
Constrain pan/zoom to y axis	hold Y
Preserve aspect ratio	hold Ctrl

   - - Clicking this button opens a "Configure subplots" window that allows the user to adjust the spacing and positioning of the individual plots.
   - - Clicking this button opens a "Save to file" window which allows the user to save a screenshot of the current interactive window.

   Use these buttons to inspect the plots in the interactive window in more detail.

8. In some cases it is worth adjusting the initial guess wavelength solution to better match the detected line positions in order for the subsequent wavelength calibration recipe to produce a more robust (and possibly better) wavelength solution, especially if the median `XDIF` and/or `YDIF` values are outside the range $\pm$ 5 pix. This may be done by adding the median `XDIF` and `YDIF` values to the values in the text boxes corresponding to the pipeline recipe parameters `trans_x` and `trans_y`, respectively, on the right-hand side of the interactive window, and then clicking the `Re-run recipe` button. The `uves_cal_predict` pipeline recipe will be executed again with the new parameter values, and the new results will appear in the interactive window. This time the median `XDIF` and `YDIF` values should be closer to zero, and the corresponding root-mean-square (RMS) values will most likely have decreased.

The adjustment of the `trans_x` and `trans_y` parameters may be iterated until the median `XDIF` and `YDIF` values are in the range $\pm$ 0.1 pix (usually two or three iterations are required).

Carry out this procedure for the current DataSet. The first values for `trans_x` and `trans_y` should be -1.62 and 11.00 pix respectively for this DataSet, with RMS statistics of 6.41 and 3.90 pix for `XDIF` and `YDIF` respectively. Further iteration should converge to values of ∼-1.69 and ∼11.27 pix for `trans_x` and `trans_y` respectively, with RMS statistics of 6.27 and 2.99 pix for `XDIF` and `YDIF` respectively. At this point, the interactive window should look like the one shown in Figure 6.4. An extreme case is shown in Figure 7.1.

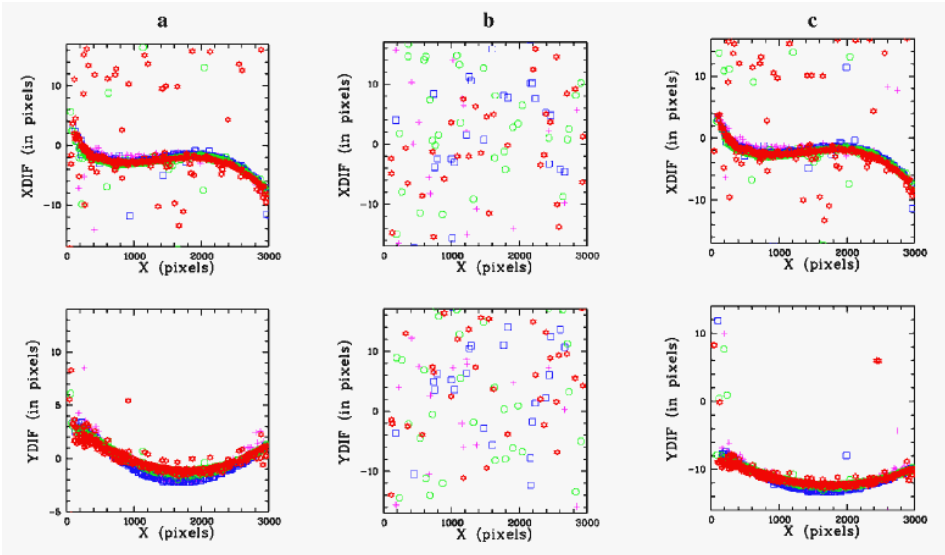


Figure 7.1: *The May 12, 2000 earthquake event as detected from the physical model control plots. The normal result obtained after successful line matching (a) produces a well concentrated distribution with mean ordinate zero. The earthquake event causes the lines matching recipe to fail (b). Adjusting the model by -10 pixels (along the cross-order direction) again matches the instrument configuration (c)..*

9. Before we move on, we note that there are three other pipeline recipe parameters on the right-hand side of the current interactive window. The parameters `mbox_x` and `mbox_y` are used to define a search box for matching the physical model line positions with the positions of the lines detected in the input arc-lamp frame. If the physical model prediction is poor, as shown by the lack of clear trends in the plots in the interactive window, then one should try larger values of `mbox_x` and `mbox_y`, such as 60 or 80 pix, in order to aid the `uves_cal_predict` recipe in determining the initial guess wavelength solution. Subsequent adjustment of `trans_x` and `trans_y` should then be performed (as described in the previous step), and then `mbox_x` and `mbox_y` should be set back to 40 pix, and one final adjustment of `trans_x` and `trans_y` should be performed to arrive at the best initial guess wavelength solution.

   The parameter `ccd_rot_angle_off` may be used to help "tighten" any less-well defined trends in the plots in the interactive window. This adjustment is usually not necessary, but if it is, then rotations of the order of 0.01-0.1 in either sense may make improvements.

10. Click the `Continue wkf` button in the interactive window.

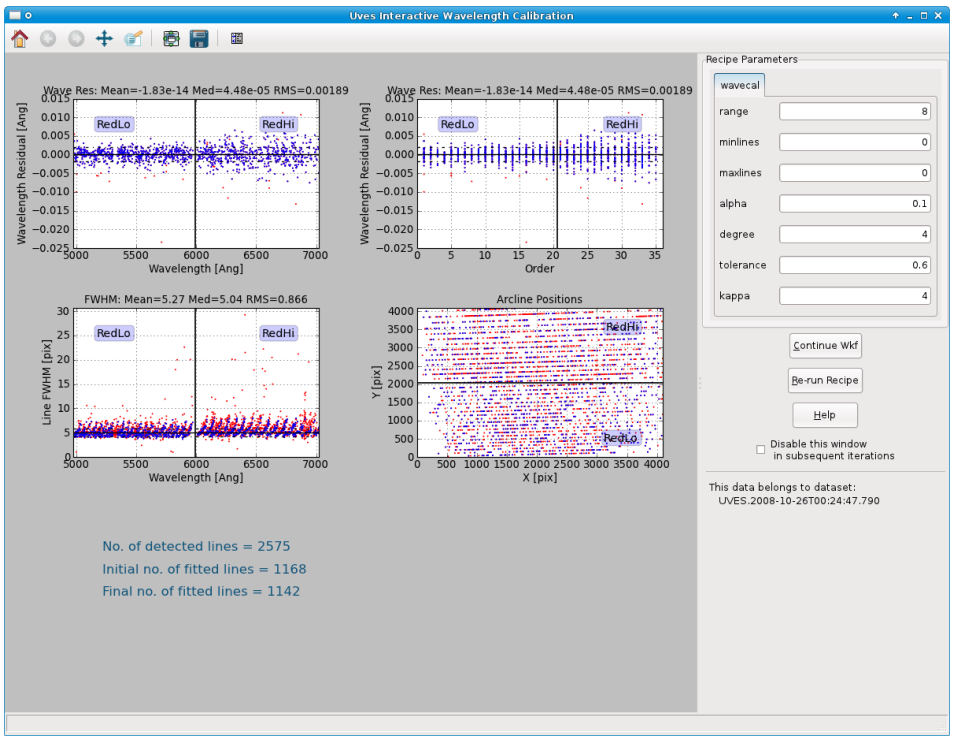


Figure 7.2: *The interactive window for the* `Wavelength Calibration` *actor for the first DataSet in the data supplied with* `Reflex 2.11.5`.

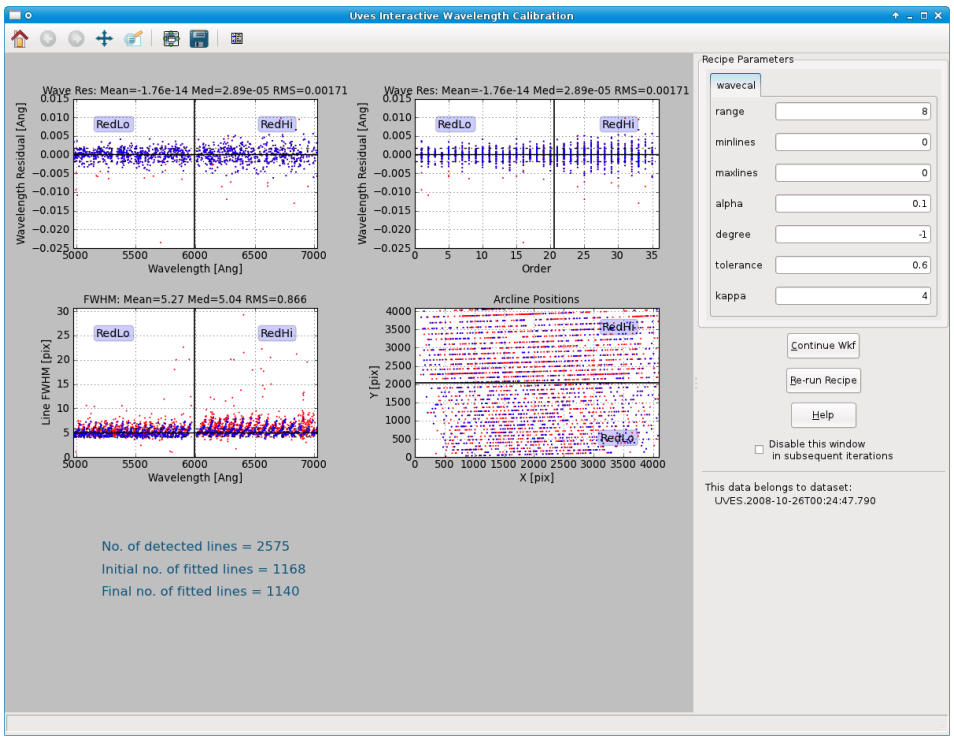


Figure 7.3: *Same as Fig. 7.2 but after iteration of the corresponding* `uves_cal_wavecal` *pipeline recipe..*

11. `Wavelength Calibration`: The next interactive window that will appear (see Figure 7.2) is associated with the execution of the `Wavelength Calibration` actor, which executes the UVES pipeline recipe `uves_cal_wavecal`. This pipeline recipe determines the wavelength calibration solution from an appropriate arc-lamp frame and a line reference catalogue. The top two plots in the interactive window show the wavelength calibration residuals for arc lines that were identified in the input line reference catalogue as a function of wavelength and spectral order. Blue points represent arc lines that were used in the final solution, and red points represent arc lines that were excluded from the final solution. The bottom-left plot shows the arc line FWHM as a function of wavelength, and the bottom-right plot displays the positions of all detected arc lines on the detector.

12. The wavelength calibration solutions produced by the UVES pipeline are generally robust and of good quality, showing very few systematic errors in the wavelength residuals and demonstrating the correct identification and clipping of outliers. The main purpose of this interactive window is therefore to allow the user to check that all is well with the wavelength solution, and to allow some further refinement if so required. Most users will simply accept the solution that is presented by clicking on the `Continue wkf` button.

    The interactive window gives access to seven pipeline recipe parameters that may be adjusted in order to improve the wavelength calibration solution, where an improvement is characterised by a smaller RMS statistic for the wavelength residuals (reported in the upper plot titles) but with a reasonable number of arc lines used in the final solution (reported as "Final no. of fitted lines"). We suggest that you attempt to improve the wavelength calibration solution for the first DataSet, based on the parameter descriptions that follow:

    - `range` - Twice the value of this parameter is the window width used for arc line detection (pix). Adjusting this parameter will influence the number of arc lines that are detected. Varying this parameter in the range 4-12 pix is sometimes useful for optimising the number of detected arc lines.
    - `minlines` - The minimum number of arc lines to be detected. This parameter should only be adjusted in conjunction with the parameter `maxlines`. Adjusting this parameter will force the pipeline recipe to find a minimum number of arc lines, if possible. This parameter is useful for controlling the number of lines used in the wavelength calibration solution.
    - `maxlines` - The maximum number of arc lines to be detected. This parameter should only be adjusted in conjunction with the parameter `minlines`. Adjusting this parameter will force a maximum on the number of arc lines the pipeline recipe finds. This parameter is useful for controlling the number of lines used in the wavelength calibration solution.
    - `alpha` - This parameter ranges from 0.0 to 1.0, and is used to control the line matches between the detected arc lines and those in the reference line catalogue by rejecting line matches where the distance to the next nearest neighbour (in the spectrum aswell as in the catalogue) is smaller than $\Delta\lambda/$`alpha`, where $\Delta\lambda$ is the wavelength residual for the current line match candidate. This parameter is best adjusted in conjunction with the parameter `tolerance`. Adjusting `alpha` in the range $\sim$0.03-0.3 is useful for controlling the number of line identifications and hence the number of lines used in the wavelength calibration solution.
    - `degree` - The degree of the polynomial model to be used in the wavelength calibration process. Usually the default value of 4 gives acceptable wavelength residuals with no systematics, but sometimes higher degrees such as 5 or 6 give better results. It is also worth trying the value -1, which instructs the pipeline recipe to determine the model degree based on the wavelength residuals.

- `tolerance` - This parameter is used to reject line identifications with wavelength residuals worse than `tolerance` from the wavelength calibration solution. This parameter is best adjusted in conjunction with the parameter `alpha`. Varying `tolerance` in the range 0.1-1.0 is sometimes useful for optimising the number of lines used in the wavelength calibration solution.
- `kappa` - The level of sigma-clipping to be performed in the final iterations of the fit of the wavelength calibration model. Suggested values for this parameter are in the range 3.0-5.0.

We find that for the first DataSet, setting `degree = -1` and leaving the other parameters at their default values, is sufficient to reduce the RMS of the residuals from 1.89×10^{-3} to 1.71×10^{-3} Ang while only reducing the final number of fitted lines from 1142 to 1140 (see Figure 7.3).

13. Click the `Continue wkf` button in the interactive window.

14. `Instrument Response:` In Figure 7.4, we show the interactive window associated with the `Instrument Response` actor, which executes the UVES pipeline recipe `uves_cal_response`. There are no response-specific recipe parameters. If the user wants to change the fit used for the response he/she has to edit the file `RESP_FIT_POINTS_CATALOG.fits`, taking care to edit the extension belonging to the specific standard star.

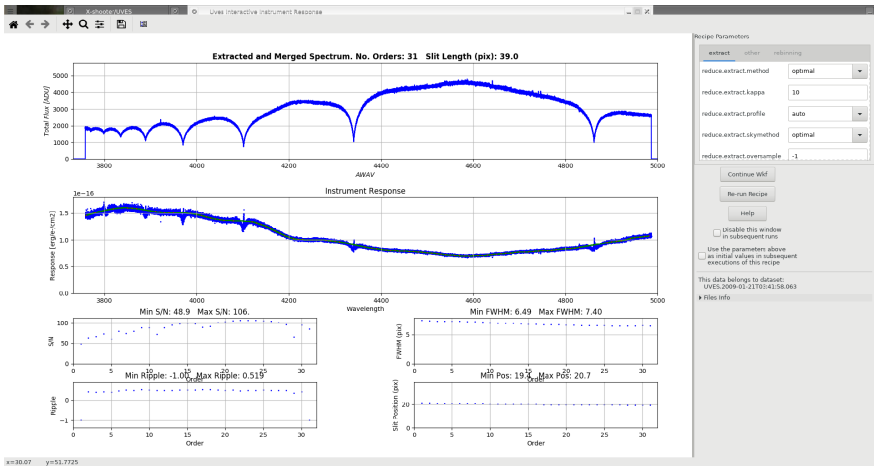


Figure 7.4: *The interactive window for the* `Instrument Response` *actor.*

15. `Spectrum Extraction:` In Figure 7.5, we show the interactive window associated with the `Spectrum Extraction` actor, which executes the UVES pipeline recipe `uves_obs_scired`. This pipeline recipe performs the extraction of the science object spectrum using optimal extraction (Horne 1986), automatically selecting an analytical spatial profile (Gaussian or Moffat) in the case of low S/N spectra (S/N $<$ 10) or an oversampled empirical spatial profile otherwise, and automatically determining the degrees of the low-order polynomials involved in the extraction process. The top plot in the interactive window shows the extracted (using optimal extraction) and merged one-dimensional spectrum of the science object (blue) with ±1-sigma uncertainties plotted as the light blue region encompassing the object spectrum (and bounded by the light grey spectra), and the plot below this shows the flux-calibrated version.

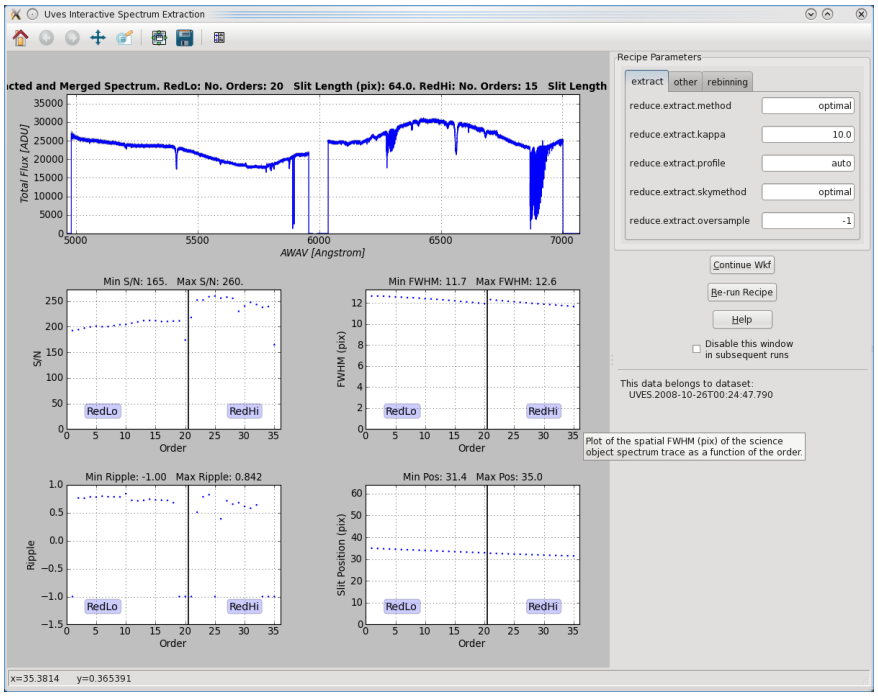


Figure 7.5: *The interactive window for the* `Spectrum Extraction` *actor for the first DataSet in the data supplied with* `Reflex 2.11.5`.

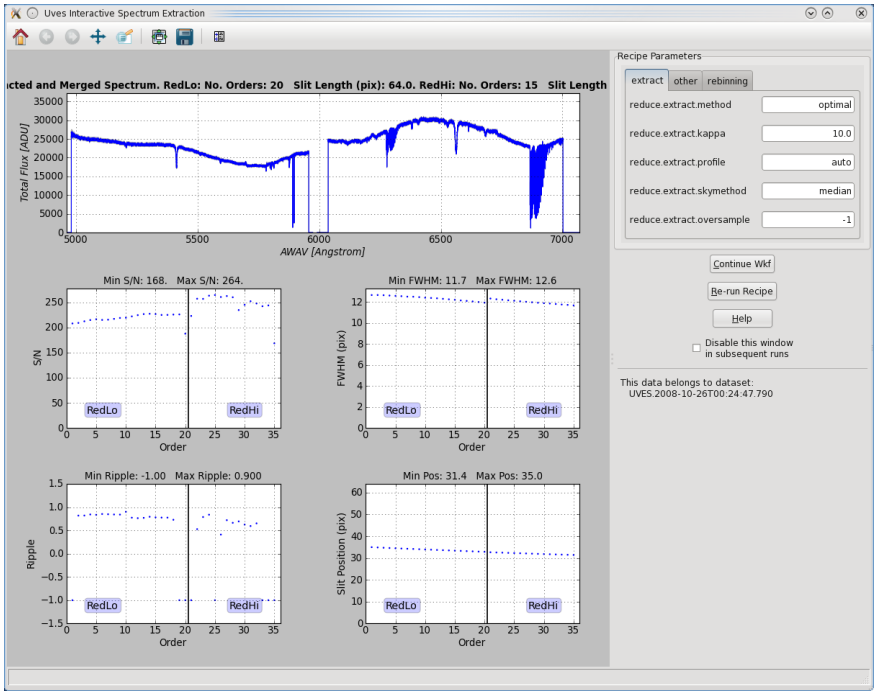


Figure 7.6: *Same as Fig. 7.5 but after iteration of the corresponding* `uves_obs_scired` *pipeline recipe.*

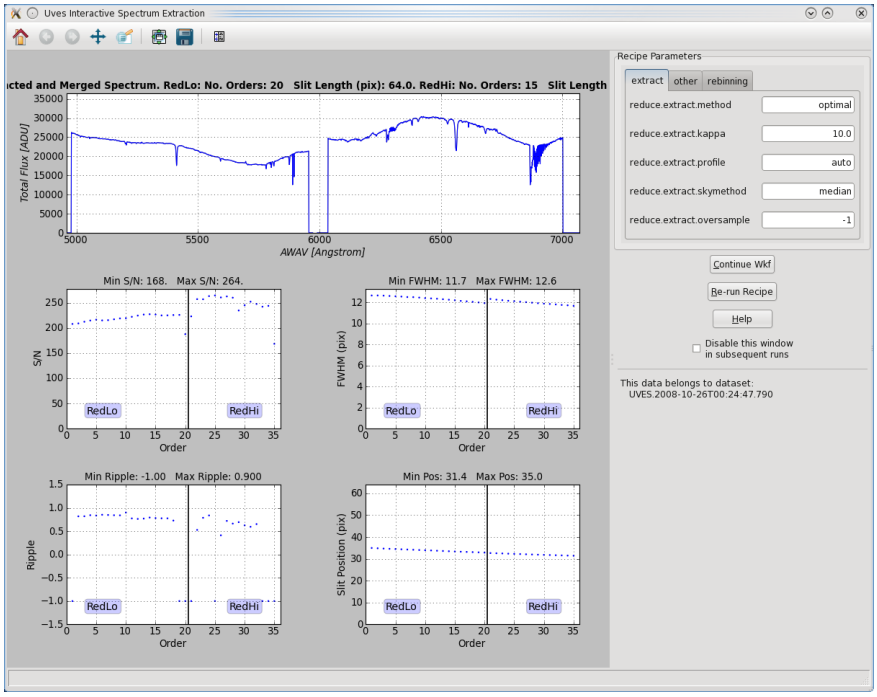


Figure 7.7: *The interactive window for the* `Spectrum Extraction` *actor for the first DataSet in the data supplied with* `Reflex 2.11.5` *after iteration of the corresponding* `uves_obs_scired` *pipeline recipe and rebinning to a resolution of 1 Ang.*

Below the plots displaying the spectra are four smaller plots which show the S/N of the extracted spectrum (upper-left), the spatial FWHM (upper-right), the "Ripple" parameter (lower-left), and the science object position on the slit (lower-right), all as a function of the spectral order. The ripple parameter is a measure of any systematic oscillations in the extracted spectrum, and should have values less than ∼5 over all orders to indicate that such oscillations are not present.

Use the buttons at the top of the interactive window to inspect these plots in detail, especially the extracted spectra. For instance, you could zoom in on the absorption feature at 5412 Ang.

16. The current interactive window provides a recipe parameter to control the application of the velocity correction (if any):

    - `velocity_correction` - Specifies which velocity correction to apply to the products, and may be set to `none` (the default), `barycentric`, or `heliocentric`. The barycentric correction value will be taken from the `HIERARCH ESO DRS BARYCORR` keyword, and the heliocentric value will be taken from the `HIERARCH ESO DRS HELICORR` keyword. Note that the application of the correction implies an adjustment of the wavelength values. Therefore if you wish to combine the spectra of the same target, you must interpolate the spectra to a common wavelength range.

    Additionally, the current interactive window provides ten (10) pipeline recipe parameters with the aim of allowing the user to optimise the extraction for quality and S/N. Try to improve the spectrum extraction for the first DataSet, based on the parameter descriptions that follow:

    - `reduce.extract.method` - The method to be used for extracting the science spectrum, and it may be set to `optimal`, `average`, or `linear` for our purposes. In most cases, the default optimal extraction is the desired method.
    - `reduce.extract.kappa` - The level of sigma-clipping to be performed when rejecting pixels suspected of being contaminated by cosmic ray hits. The default value of 10.0 should be reduced to smaller values (down to ∼3.0) when an extracted spectrum shows sharp emission spikes that should not be present
    - `reduce.extract.profile` - The spatial profile model to be used when performing optimal extraction, and it may be set to `auto`, `virtual`, `gauss`, or `moffat` for our purposes. The default parameter value `auto` instructs the pipeline recipe to automatically decide on the type of spatial profile model to employ based on the S/N of the science spectrum. Sometimes it is worth testing the other spatial profiles that are available.
    - `reduce.extract.skymethod` - The method for calculating the sky level may be set to one of two values; namely `optimal` or `median`. When set to `optimal`, the sky level is included in the optimal extraction model, and when set to `median`, the sky level is estimated via the median and subtracted from the science spectrum before extraction. Sometimes the non-default `median` option can give slightly better S/N than the default `optimal` option.
    - `reduce.extract.oversample` - The oversampling factor to be used in the empirical spatial profile. When the empirical spatial profile is employed by the pipeline recipe, the oversampling factor is set by default to 5 for S/N < 200, and to 10 for S/N > 200. Sometimes a larger value for the oversampling factor may give a higher S/N extracted spectrum, especially for very high S/N spectra.

- `reduce.slitlength` - This parameter may be used to control the slit length used to define the spatial profile in the spectrum extraction. The default negative value instructs the pipeline recipe to read the slit length from the science object FITS header. Setting this parameter to smaller values than the slit length in the header may sometimes be useful in avoiding problems at the order edges, with the disadvantage that the maximum achievable S/N will be smaller (less pixels involved in the extraction).
- `reduce.objoffset` - This parameter may be used in conjunction with the parameter `reduce.slitlength` in order to extract the spectra of more than one object observed on the slit in the same science image. Please see the section on "Optimal Extraction" in the chapter on "Algorithms And Recipe Details" in the pipeline user manual (Larsen et al. 2012) for more details.
- `reduce.objslit` - Object window size (in pixels). This must be less than the total slit length. If negative, the default value (half of full slit length) is used. The upper and lower sky windows are defined as the part of the full slit (if any) outside the object window. The center of the object window is determined by the offset parameter. This parameter does not apply to optimal extraction.
- `reduce.rebin.wavestep` - The rebinning step size used for BLUE/REDL data in wavelength units for the extracted science spectrum. The default negative value instructs the pipeline recipe to use the optimal resolution. Setting this parameter to a positive value is useful for rebinning the spectrum to a lower resolution while gaining a substantial increase in S/N. Hence this parameter is more concerned with spectrum post-processing rather than extraction optimisation.
- `reduce.rebin.wavestep_redu` - Identical to `reduce.rebin.wavestep_redu` except that it applies to REDU data only.

We find that for the first DataSet, setting `reduce.extract.skymethod = median` and `reduce.extract.oversample = 10`, and leaving the other parameters at their default values, increases the maximum spectrum S/N from 260 to 264 (see Figure 7.6).

17. For illustrative purposes, set `reduce.rebin.wavestep = 1.0 Ang` (and/or `reduce.rebin.wavestep_redu = 1.0 Ang`) for the first DataSet, and re-run the pipeline recipe. In Figure 7.7 we display the interactive window with the rebinned spectrum. Notice that the S/N has been greatly improved at the cost of a lower resolution. Since the spectrum rebinning is a post-processing operation, the S/N per order reported in the interactive window corresponds to the S/N before rebinning.

18. Click the `Continue wkf` button in the interactive window.

19. As noted in the quick start Section 3, the workflow will write out the extracted spectrum and other important products to the end products directory (specified by the parameter `END_PRODUCTS_DIR` under "Setup Directories" in the workflow canvas).

20. The workflow will now repeat the above reduction steps for the next DataSet, looping over each DataSet until all of them have been processed. The other demo DataSets are provided to allow you to experiment with optimising the reductions. We advise you to follow the above interactive procedures until you feel that you have mastered the functionality of the pipeline parameters that are provided in the interactive windows.

8 Frequently Asked Questions

- **The error window fills the whole screen - how can I get to the `Continue`/`Stop` buttons?**

  Press the Alt key together with your left mouse button to move the window upwards and to the left. At the bottom the `Continue`/`Stop` buttons will be visible. This bug is known but could not yet be fixed.

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

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

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

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

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

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas, under Setup Directories) and organised further in directories by pipeline recipe.

- **Can I use different sets of bias frames to calibrate my flat frames and science data?** Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it (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.

- **If I enter "-" into an empty integer parameter of an interactive window it is automatically completed to "-1". Why?**

  The parameters are validated for correctness according to their type (e.g. string, integer, float). In the case of an integer or float parameter "-" alone is considered an invalid input and is therefore automatically completed to "-1". This is part of the validation of input done by the WxPython library.

- **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

- **I have many DataSets in my data directory. How can I reduce them interactively without having to wait a long time between interactive windows being displayed?** Reduce all the DataSets at once with the interactive windows disabled for all interactive actors. When this reduction has finished, you should re-enable the interactive windows that you require, and run the workflow again. The workflow will run in Lazy mode and no time will be spent on pipeline reductions, unless you specifically change a parameter in one of the interactive windows.

  Note that Lazy mode will not work if the workflow parameter `EraseDirs` is set to `true`.

8.1 UVES Specific Questions

- **To obtain best calibration accuracy, should I use the response curve obtained from the actual standard star data reduction or the master response curve?** Both the response derived from the standard star observed close in time to your data and the master response should give you a good relative flux calibration. In rare cases the response derived from an individual standard star may have fit problems. So if you want to be absolutely sure that this does not happen use the master response. If you are interested in an accurate *absolute* flux calibration the response derived from a standard star observed on the same night as your science data should be better. Please note that for absolute flux calibration you need to ensure photometric conditions and observe your science target with a wide slit, because otherwise wavelength-dependent slit losses occur.

- **I have a standard star in my DataSet but it is not found in the standard star catalog. Why?** Most likely you have one of the old UVES standard stars, for which no fine sampled high resolution reference data exist. If you have to use this specific star you can find the catalog and a table with fit points at *https://ftp.eso.org/pub/dfs/pipelines/instruments/uves/uves_response_old_calib.tar.gz*. Please be aware that these data are not well suited for the response determination (see next question for details).

- **The response curve derived from my standard star looks very bad and the raw response shows huge residuals. Why?** The pipeline supports seven standard stars, for which high-resolution reference data based on model spectra exists: EG 21, EG 274, Feige 110, GD 71, GD 153, LTT 3218, and LTT 7987. For other standard stars observed with UVES the reference data were mostly derived from low-resolution observations and are sampled at coarse intervals (typically 50 Å).

  Fig. 8.1 shows an example for the standard star HR 9087 observed with the 390 setup. The left plot shows the ratio of the observed spectrum and the interpolated reference data (black) and the points used to fit the response (red). These fit points were optimized for the new high-resolution reference data, which yield raw response curves with small residuals at line cores only. Obviously they are not suitable for this observation. The right plot illustrates this problem by showing in black the reference data, in red the interpolated reference spectrum and in blue the observed spectrum, scaled to fit into the plot. The observed lines are obviously much better resolved than the reference data. So even if the fit points were revised the response curve will not be good. If you absolutely need to use such a standard star you have to refine the fit points by editing the corresponding entry in the static calibration file `RESP_FIT_POINTS_CATALOG`.

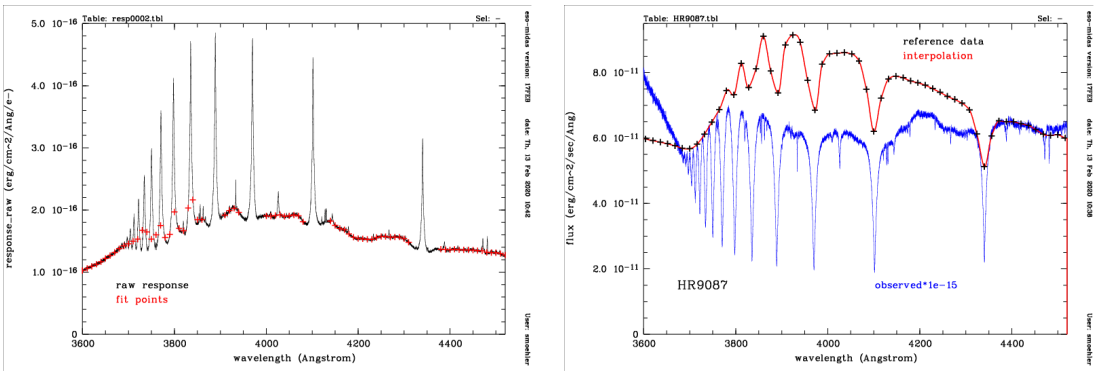


Figure 8.1: The **left** plot shows the raw response (black), together with the fit points (red) for an observation of the standard star HR 9087 with the 390 nm setup. The **right** plot shows the reference data for HR 9087 (black), the interpolation used by the pipeline (red) and for comparison a scaled version of the observed UVES spectrum.

- **I have many DataSets of extended objects. How can I reduce them?** UVES observations of extended object can be reduced with the normal workflow. We have not verified that the corresponding products are science data products.

- **I want to have the two-dimensional rectified spectrum instead of the one-dimensional extracted one. How can I get that?**
  To get two-dimensional products you have to set the `extract_method` parameter of the `uves_obs_scired` recipe to `2d`. To do so in the Reflex workflow double-click on the

`SpectrumExtraction` actor, change `INIT_EXTRACT_METHOD` from `optimal` to `2d`, and click on Commit. To keep this setting for future processing you may save the workflow via `File→Export As`. Please note that if you want **both** one- and two-dimensional spectra you have to run the workflow twice with different values for `extract_method`.

- **I have many DataSets of objects observed with the Image Slicer. How can I reduce them?** UVES observations of object observed with the Image Slicer can be reduced with the normal workflow. As the observed object is EXTENDED the user may want to keep full spatial resolution during extraction, by setting to '2d' the parameter `reduce.extract.method`. In this case Reflex does not allow to display results, thus the user should disable the interactivity for the Spectrum Extraction. We have not verified that the corresponding products are science data products.

- **I have many DataSets of objects observed with the the Iodine Cell filter. How can I reduce them?** UVES observations of object observed with the Iodine Cell filter is not yet supported.

9 Troubleshooting

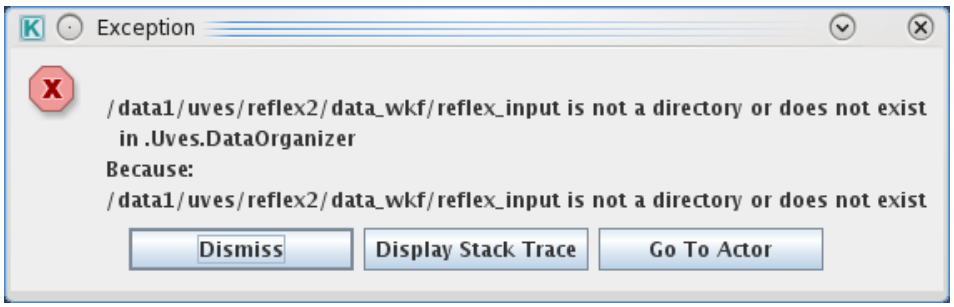


Figure 9.1: *TheDataOrganizer interactive window reports an error ":No Datasets have been created, check the DataSet and the OCA rules.".*

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

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

   (b) **it fails with error message ":No DataSets have been created, check the data set and the OCA rules."(see Figure 9.1.)**
   This error may be due to the fact that the data provided by the ESO archive are compressed (`<filename>.fits.Z`). Please remember to uncompress the data before executing `Reflex`.

   (c) **all DataSets are greyed out in the DataSets interactive window.**
   The ESO archive used with CalSelector does not always supply all static calibration files. As a consequence some/all DataSets are greyed out because they were missing such required data.
   Missing static calibration should be found by `reflex` in `<install_directory>/calib/<pipeline_version>/cal`.

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

   If a DataSet in the "Select DataSets" window is greyed out, then it means that the DataSet that was constructed is missing some key calibration(s) (i.e. the DataSet is incomplete). To find out what calibration(s) are missing from a greyed out DataSet, click on the DataSet in question to highlight it in blue, and then click on the button `Inspect Highlighted`. The "Select Frames" window that appears will report the category of the calibration products that are missing (e.g. `MASTER_FLAT_REDL_WKF`, `MASTER_FLAT_REDU_WKF`, `ORDER_GUESS_TAB_REDL_WKF`, `ORDER_GUESS_TAB_REDU_WKF`, `LINE_GUESS_TAB_REDL_WKF`, `LINE_GUESS_TAB_REDU_WKF`). From this the user has then to determine the missing raw data (in this case formatcheck and flat frames for the RED arm). If static calibrations are missing the mechanism unfortunately does not work, but should be found by `reflex` in `<install_directory>/calib/<pipeline_version>/cal`

A faster way to identify which kind of data are missing is moving the mouse over the grey DataSet (see Fig. 9.2).

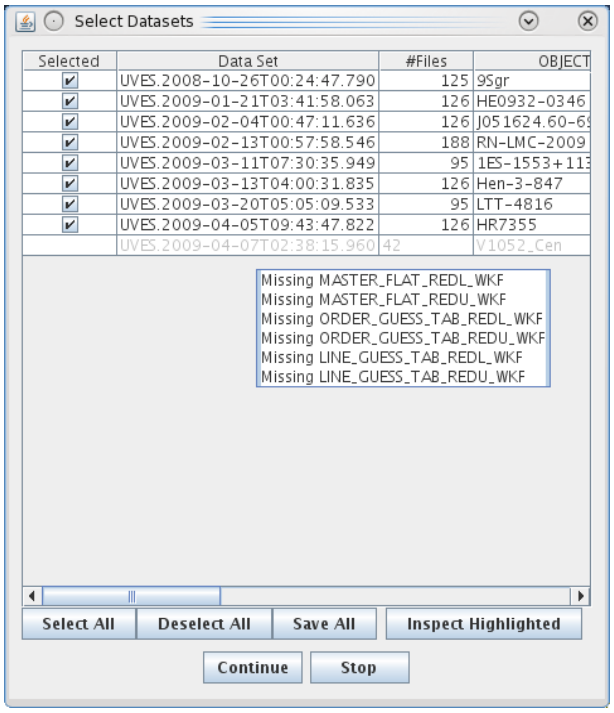


Figure 9.2: *This shows how information about missing files in an incomplete DataSet is displayed when moving the mouse over it.*

3. **I downloaded the data from the ESO archive and put them into a new directory, tried to run `Reflex` on them, but all DataSets are greyed out in the DataSets interactive window.**

   The ESO archive does not supply as default general calibration files such as the LINE_REFER_TABLE. As a consequence all DataSets are greyed out because they were missing such required data.

   The user may solve this problem remembering that any reference data file required for the data reduction is also provided by the pipeline distribution kit.

4. **I have only DFLATS for a given DataSet, which is shown as incomplete by the DataSetChooser. Is there support for this setup?**

   Yes, there is support. Use the `uves_wkf_onlydflatsd.oca` in the Data Organizer actor configuration.

5. **In some platforms, when I click on the "Configure subplots" button in the interactive window (see Section 7, step 6), the window that appears is empty. Can this be corrected?**

   This is a known problem that will addressed in future releases.

6. **Artifacts present in merged spectra of RED760 (or beyond) settings. Can this be removed?**

For data observed in the red settings (760 or beyond), we recommend to use `reduce.ffmethod = "pixel"` (together with `reduce.merge.delt1=14` and `reduce.merge.delt2=4`). Thereby the user can avoid artefacts from residual flat-field fringes that have a noise-like structure and might dominate the spectral structure and compromise the signal-to-noise ratio. The drawback of that method are short spectral gaps between the orders in the REDU range.

7. **Reducing EXTEND data the Spectrum Extraction interactive display reports missing expected inputs. Can this be removed?** The reflex interactive workflow python GUI has been designed to support display of products of point like objects. Reduction of extended sources has only a basic support. We recommend in this case to set the "EnableInteractivity" parameter to `false`.

8. **The Wavelength Calibration fails. How I can recover this problem?** Occasionally walelength calibration may fail witb error like (on purpose to make the error message readable we wrapped around long text messages):

```
[null] Standard error:
[null] [ ERROR ] WaveCal: Dumping all 6 error(s):
[null] [ ERROR ] WaveCal:   [1/6] 'Illegal input: For poly
       regression you need at least 9 points. The table with
       column to compute regression has 4 rows!' (14) at
       uves_polynomial_regression_2d:uves_utils.c:2899
[null] [ ERROR ] WaveCal:   [2/6] 'Illegal input: Error fitting polynomial.
       Possible cause: too few (29) line identifications' (14) at
       calibrate_global:uves_wavecal_identify.c:1205
[null] [ ERROR ] WaveCal:   [3/6] 'Illegal input: Could not perform global calibration'
       (14) at uves_wavecal_identify:uves_wavecal_identify.c:357
[null] [ ERROR ] WaveCal:   [4/6] 'Illegal input: Could not calibrate orders' (14) at
       uves_wavecal_process_window:uves_wavecal_body.c:722
[null] [ ERROR ] WaveCal:   [5/6] 'Illegal input: Wavelength calibration failed' (14) at
       uves_wavecal_exe_body:uves_wavecal_body.c:1914
[null] [ ERROR ] WaveCal:   [6/6] 'Illegal input: WaveCal execution failed' (14) at
       uves_cal_wavecal:uves_wavecal_impl.c:211
[null] [ ERROR ] esorex: Execution of recipe 'uves_cal_wavecal' failed, status = 14
```

A possible reason may be a large shift in the spectral format (mainly along cross order direction). In this case the physical model plot provided by the spectrum locator actor usually shows a widely scattered distribution of residuals instead of a clear correlation (see Fig. 7.1 for an example). In this case we recommend to get format-check and wavelength calibration frames from previous dates (up to a few days before the problematic one) and compare them with the problematic one to derive the X and Y shifts. Then re-reduce the data by setting the `trans_x` and `trans_y` to the found values (with sign) of the shift of the actual to the reference frame, and optimise the resuls of the model prediction as described at page 8. This should allow to solve the problem.

9. **Reducing science data the extracted spectrum in the interactive window appears as 0 with one (or more) spikes. Can this be solved?** Occasionally extraction may still containe one of few spikes with intensity of several order of magnitude different from the one of the extracted spectrum. Due to the way the interactive plot min/max values are set, this may imply the spectrum may appear flattened at "0". In this case we recommend the use to rescale the spectrum zooming in, or simply plot the spectrum alternatively and set reasonable Y plot ranges.

10. **Science spectra not flux calibrated**. Due to increased proper motion of the standard star, the recipe uves_cal_response does not find the reference standard star in the input FLUX_STD_TABLE catalog,

and consequently cannot compute the instrument response and flux calibrate the science data. In which case we recommend to increase the value of the parameter **paccuracy** for example from 60 to 70 [arcsec].

Acknowledgements

The `Reflex` team in alphabetical order consists of Pascal Ballester, Daniel Bramich, Vincenzo Forchì, Wolfram Freudling, César Enrique García, Maurice Klein Gebbinck, Andrea Modigliani, Sabine Moehler & Martino Romaniello. Our thanks go to R. Figuera Jaimes for volunteering to be our first external test user.

Forchì V., 2012, Reflex User Manual, VLT-MAN-ESO-19000-5037, Issue 3.4 15, 21, 46, 47

Horne K., 1986, PASP, 98, 609 41

Kaufer A. et al., UV-Visual Echelle Spectrograph User Manual, VLT-MAN-ESO-13200-1825, Issue 86 7

Larsen J.M., Modigliani A. & Bramich D.M., UVES Pipeline User Manual, VLT-MAN-ESO-19500-2965, Issue 6.6.1 7, 21, 29, 30, 31, 45