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 Sinfoni Tutorial

VLT-MAN-ESO-19500-XXXX

Issue 1.8

Date 2017-04-19

Prepared:	Andrea Modigliani	2017-04-19	
	Name	Date	Signature
Approved:	W.Freudling		
	Name	Date	Signature
Released:	M. Sterzik		
	Name	Date	Signature

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	25/10/2013	All	First release
1.1	25/03/2014	Update	Updated to sinfo-2.5.2
1.2	24/02/2015	Update	Updated to sinfo-2.6.0
1.3	20/07/2015	Update	Updated to Reflex 2.8 and sinfo-2.6.5
1.4	14/09/2015	Update	Updated to sinfo-2.6.8
1.5	26/10/2015	Update	Updated to sinfo-2.7.0
1.6	03/03/2016	Update	Updated to sinfo-2.9.0
1.8	19/04/2017	Update	Updated to sinfo-3.0.0

This page was intentionally left blank

Contents

1 Introduction And Scope

`Reflex` is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine 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 contact *usd-help@eso.org* for further support.

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

A workflow accepts science and calibration data, as 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 SINFONI Integral Field Unit observations only via the SINFONI workflow. The user is referred to the SINFONI web page (http://www.eso.org/sci/facilities/paranal/instruments/sinfoni/) for more information on the instrument itself, and the SINFONI pipeline user manual for the details of the pipeline recipes (http://www.eso.org/sci/software/pipelines/).

The workflow uses association rules known to work with files downloaded from the ESO archive with the CalSelector tool (from year 2009 onwards). For older datasets where the data were directly delivered to the PI (e.g. in DVDs) see Section 7.

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

2 Software Installation

Reflex 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 `macports` repositories support OS X, while the `rpm/yum` repositories support Fedora 20/21/22/23. For any other operating system it is recommended to use the `install_esoreflex` script.

2.1 Installing Reflex workflows via `macports`

This method is supported for the OS X operating system. It is assumed that macports (*http://www.macports.org*) and java are installed. If you have any problem with this installation method, please read the full documentation at
http://www.eso.org/sci/software/pipelines/installation/macports.html.

For a quick installation, the following steps will install the ESO pipeline macports repository, the SINFONI pipeline, including the Reflex workflow support and Reflex itself:

- Set up the repository:

```
# curl ftp://ftp.eso.org/pub/dfs/pipelines/repositories/macports/setup/Portfile -o Portfile
# sudo port install
# sudo port sync
```

- Install the SINFONI pipeline:
  `# sudo port install esopipe-sinfo-all`

2.2 Installing Reflex workflows via `rpm/yum`

This method is supported for Fedora 20/21/22/23 operating systems. If you have any problem with this installation method, please read the full documentation at
http://www.eso.org/sci/software/pipelines/installation/rpm.html.

For a quick installation, the following steps will install the ESO pipeline rpm repository, the SINFONI pipeline, including the Reflex workflow support and Reflex itself:

- Set up the repository for Fedora 20/21:

```
# sudo yum install yum-utils
# sudo yum-config-manager \
  --add-repo=ftp://ftp.eso.org/pub/dfs/pipelines/repositories/fedora/esorepo.repo
```

- Set up the repository for Fedora 22/23:

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

- Install the SINFONI pipeline (Fedora 20/21):
  `# sudo yum install esopipe-sinfo-all`
- Install the SINFONI pipeline (Fedora 22/23):
  `# sudo dnf install esopipe-sinfo-all`

2.3 Installing Reflex workflows via `install_esoreflex`

The software pre-requisites for `Reflex 2.9` may be found at:
http://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.9` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

   ```
   wget ftp://ftp.eso.org/pub/dfs/reflex/install_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. You will be asked whether you want to use your Internet connection. Unless you want to reuse already downloaded packages (only advanced users), use the default Yes.

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

6. For the pipelines to be installed you will be prompted for the demo data sets to be installed. Type "A" for all demo datasets. Take into account that if you are installing in a directory that already contains data, it won't be removed.

7. The script will also detect whether previous versions of the workflows or Reflex were installed and in this case you have the option to update links or remove obsolete cache directories. It is advised to use the defaults.

8. 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.

3 Demo Data

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

Note that you will need a minimum of ∼2.0 GB, ∼0.6 GB and ∼3.8 GB of free disk space for the directories `<download_dir>`, `<install_dir>` and `<data_dir>`, respectively. The SINFONI demo data have been retrieved with the CalSelector tool[2].

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

4 Quick Start: Reducing The Demo Data

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the SINFONI demo data set supplied with the `Reflex 2.9` release. By following these steps, the user should have enough information to perform a reduction of his/her own data without any further reading:

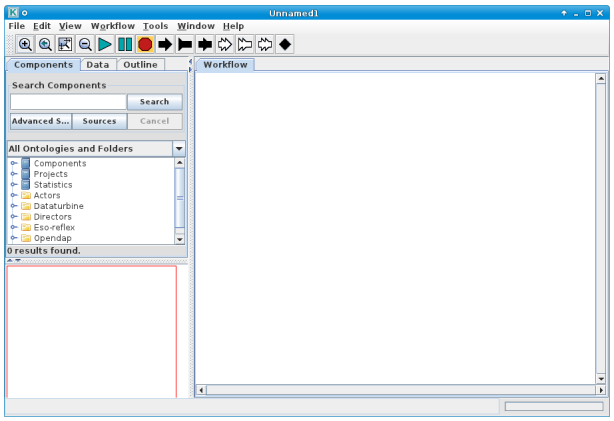


Figure 4.0.1: *The empty* `Reflex` *canvas.*

1. Start the `Reflex` application:

   ```
   esoreflex &
   ```

   The empty `Reflex` canvas as shown in Figure 4.0.1 will appear.

2. Now open the SINFONI workflow by clicking on `File -> Open File`, selecting first `sinfo-3.0.0` and then the file `sinfo.xml` in the file browser. You will be presented with the workflow canvas shown in Figure 4.0.2. Note that the workflow will appear as a canvas in a new window.

3. To aid in the visual tracking of the reduction cascade, it is advisable to use component (or actor) highlighting. Click on `Tools -> Animate at Runtime`, enter the number of milliseconds representing the animation interval (100 ms is recommended), and click `OK`.

4. Under "Setup Directories" in the workflow canvas there are seven parameters that specify important directories (green dots). Changing the value of `ROOT_DATA_DIR` and/or `RAW_DATA_DIR` is the only necessary modification if you want to process data other than the demo data[3], since the value of this parameter specifies the working directory within which the other directories are organised. Double-click on the parameter `ROOT_DATA_DIR` and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the `Browse` button to select the directory from a file browser. When you have finished, click `OK` to save your changes.

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

5. Click the  button to start the workflow

6. The workflow will highlight the `Data Organiser` actor which 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 `Reflex` cannot decide which one to use.

7. The `Data Set Chooser` actor will be highlighted next and will display a "Select Datasets" window (see Figure 4.0.3) that lists the DataSets along with the values of a selection of useful header keywords[4]. 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.

8. Click the `Continue` button and watch the progress of the workflow by following the red highlighting of the actors. A window will show which DataSet is currently being processed.

9. When the reduction of the current DataSet finishes, a pop-up window called *Product Explorer* will appear showing the datasets which have been so far 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 4.0.4 shows the Product Explorer window.

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

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

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

[4]The keywords listed can be changed by right-clicking on the `DataOrganiser` Actor, selecting `Configure Actor`, and then changing the list of keywords in the second line of the pop-up window.

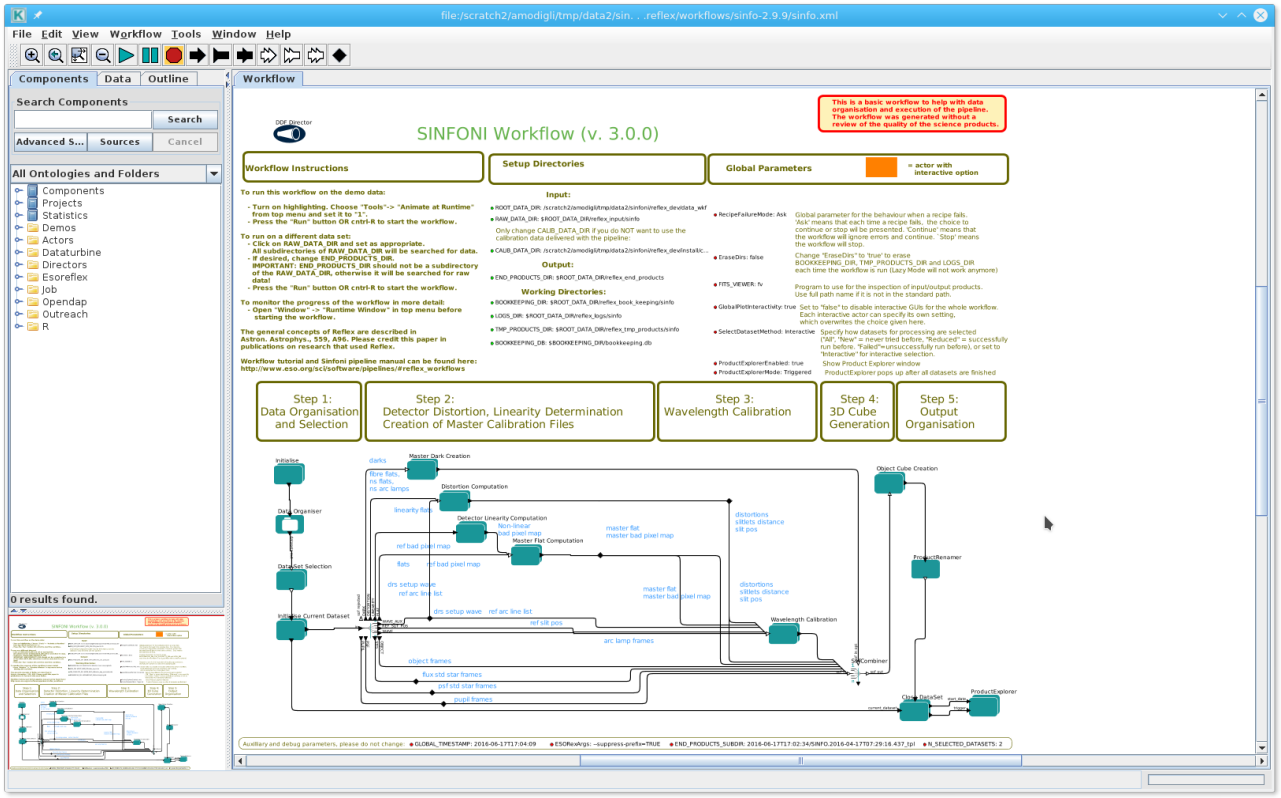


Figure 4.0.2: *SINFONI workflow general layout.*

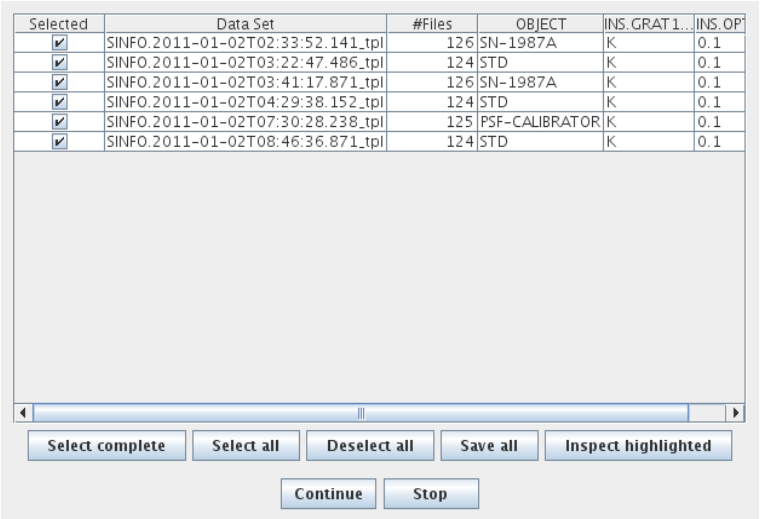


Figure 4.0.3: *The "Select Datasets" pop-up window.*

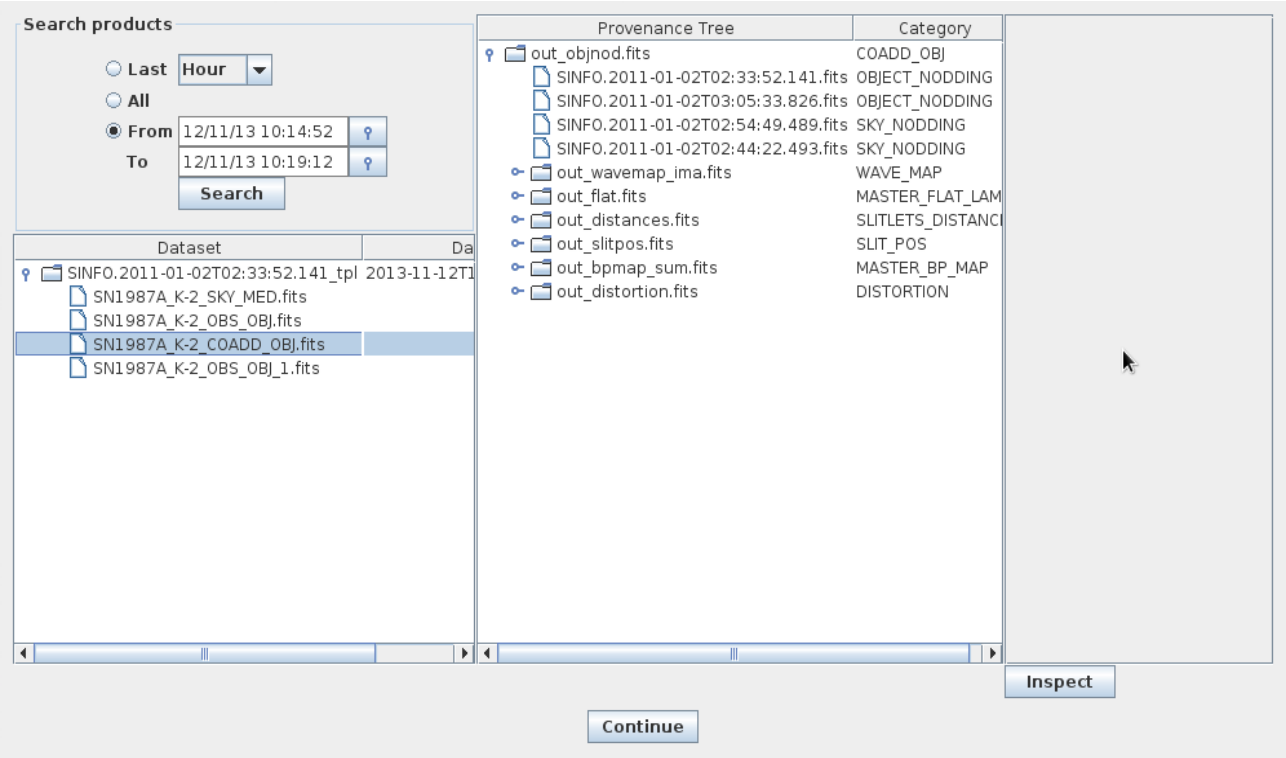


Figure 4.0.4: *The Provenance Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products.*

5 About The Reflex Canvas

5.1 Saving And Loading Workflows

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

5.2 Buttons

At the top of the `Reflex` 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.

5.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the  ,  , and  buttons, respectively. A workflow is executed by clicking the  button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the  button, or the workflow may be paused by clicking the  button which will allow the current actor/recipe to finish execution before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the  button again.

6 The SINFONI Workflow

The SINFONI workflow canvas is organised into a number of areas. From top-left to top-right you will find general workflow instructions, directory parameters, and global parameters. In the middle row you will find five boxes describing the workflow general processing steps in order from left to right, and below this the workflow actors themselves are organised following the workflow general steps.

6.1 Workflow Canvas Parameters

The workflow canvas displays a number of parameters that may be set by the user. Under "Setup Directories" the user is only required to set the `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.

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 6.2.4), reusing the previous pipeline recipe outputs where input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the `EraseDirs` parameter to `true`, which forces the workflow to recursively delete the contents of the directories specified by `BOOKKEEPING_DIR`, `LOGS_DIR`, and `TMP_PRODUCTS_DIR`. This is useful for keeping disk space usage to a minimum and will force the workflow to fully rereduce the data each time the workflow is run.

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`, `Disabled` and `Triggered`. The latter, recommended, means that the `ProductExplorer` actor will be shown only at the end of the workflow execution.

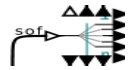
6.2 Workflow Actors

6.2.1 Simple Actors

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

- 

- The `Data Organiser` actor.
- 

- The `Data Set Chooser` actor (inside a composite actor).
- 

- The `Fits Router` actor
- 

- The `Product Renamer` actor.
- 

- The `Product Explorer` 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).

6.2.2 Composite Actors

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

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

6.2.3 Recipe Execution within Composite Actors

The SINFONI workflow contains Composite Actors to run pipeline recipes. This is in the simplest case due to the `SoF Splitter`/`SoF Accumulator`[5], which allow to process calibration data from different setting within one given DataSet (e.g. lamp frames taken with different preoptics/band-filters). More complex Composite Actors contain several actors (e.g. `Recipe Executer`).

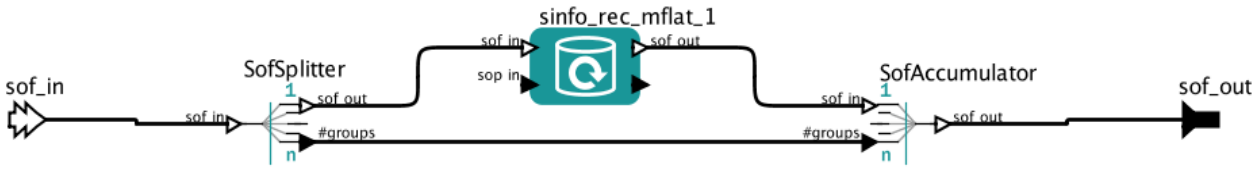


Figure 6.2.1: *This is the window you get when you choose* `Open Actor` *for the Composite Actor* `Master Flat Computation`. *This is the simplest case for a Composite Actor. Using* `Configure Actor` *on* `sinfoni_flat_1` *gives you Fig. 6.2.2.*

The central elements of any Reflex workflow are the `RecipeExecuter` actors that actually run the recipes. One basic way to embed a `RecipeExecuter` in a workflow is shown in Fig 6.2.1, which is the simplest version of a Composite Actor. The `RecipeExecuter` is preceded by an `SofSplitter`, and followed by an `SofAccumulator`. The function of the `SofSplitter` is to investigate the incoming SoFs, sort them by "purpose", and create separate SoFs for each purpose. The `RecipeExecuter` then processes each of the SoFs independently. Note that the lazy mode may be triggered if two sof are the same. Finally, the `SofAccumulator` packs all the results into a single output SoF. The direct relation between the `SofSplitter` and `SofAccumulator` is used to communicate the number of different SoFs created by the `SofSplitter`. A workflow will only work as intended if the "purpose" of all the files a recipe needs as input is identical. The only exception to this rule is that a purpose can also be "default". In this case, the file is included in any output SoF created by the `SoFsplitter` and `SofAccumulator`.

The reason for this scheme is best explained by an example. For a complex DataSet, the `Data Organiser` might have selected a large number of individual raw lamp frames (arc and flat field). The different lamp frames are to be used to calibrate different frames, e.g. the science frames and the standard star frames. The `Data Organiser` determines and records this "purpose" of each lamp frame, and this information is included in the DataSet and each SoF created from this DataSet. The `FitsRouter` directs all raw lamp frames to the calibration Composite Actor. The `SofSplitter` then creates SoFs, one for the lamp frames to be used for the science frames, and (probably) separate ones for the lamp frames to be used for the standard star observations. The calibration recipe creates one master flat field (and other products) for each SoF, and the `SofAccumulator` then creates a SoF that contains all the products.

A `RecipeExecuter` actor is used in the workflow to run a single SINFONI pipeline recipe (e.g: in the `Master Flat Computation` actor the recipe `sinfo_rec_mflat` is executed). In order to configure

[5]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. `FLAT`).

the `RecipeExecuters`, one has to first use `Open Actor` to get to the level of the recipe executors (see Fig. 6.2.1).

In Figure 6.2.2 we show the "Edit parameters" window for a typical `RecipeExecuter` actor, which can be displayed by right-clicking on the actor and selecting `Configure Actor`. In the following we describe in more detail the function of some of the parameters for a `RecipeExecuter` actor:

- The "recipe" parameter states the SINFONI pipeline recipe which will be executed.
- The "mode" parameter has a pull-down menu allowing the user to specify the execution mode of the actor. The available options are:
  - `Run`: The pipeline recipe will be executed, possibly in Lazy mode (see Section 6.2.4). This option is the default option.
  - `Skip`: The pipeline recipe is not executed, and the actor inputs are passed to the actor outputs.
  - `Disabled`: The pipeline recipe is not executed, and the actor inputs are not passed to the actor outputs.
- The "Lazy Mode" parameter has a tick-box (selected by default) which indicates whether the `RecipeExecuter` actor will run in Lazy mode or not. A full description of Lazy mode is provided in Sect. 6.2.4.
- The "Recipe Failure Mode" parameter has a pull-down menu allowing the user to specify the behaviour of the actor if the pipeline recipe fails. The available options are:
  - `Stop`: The actor issues an error message and the workflow stops.
  - `Continue`: The actor creates an empty output and the workflow continues.
  - `Ask`: The actor displays a pop-up window and asks the user whether he/she wants to continue or stop the workflow. This option is the default option.
- The set of parameters which start with "recipe param" and end with a number or a string correspond to the parameters of the relevant SINFONI pipeline recipe. By default in the `RecipeExecuter` actor, the pipeline recipe parameters are set to their pipeline default values. If you need to change the default parameter value for any pipeline recipe, then this is where you should edit the value. For more information on the SINFONI pipeline recipe parameters, the user should refer to the SINFONI pipeline user manual (Modigliani et al. 2013[6]).

The description of the remainder of the `RecipeExecuter` actor parameters are outside the scope of this tutorial, and the interested user is referred to the Reflex User Manual for further details (Forchì 2012). Any changes that you make in the "Edit parameters" window must be saved in the workflow by clicking the `Commit` button when you have finished to take effect. If you want to reuse the parameters you have to save the workflow with the saved parameters.

[6] Available at *ftp://ftp.eso.org/pub/dfs/pipelines/sinfoni/sinfo-pipeline-manual-16.0.pdf*

Table 6.2.0: The SINFONI pipeline actors and their contents

actor	recipe	description
Master Dark Computation	`sinfo_rec_mdark`	create master dark
Distortion Computation	`sinfo_rec_distortion`	compute distortions
Detector Linearity Computation	`sinfo_detlin`	determine detector linearity
Master Flat Computation	`sinfo_rec_mflat`	create master flat
detector's distortions and slitlets distances		
Wavelength Calibration	`sinfo_rec_wavecal`	determine wavelength calibration
solution, wave map, slit pos table.		
Object Cube Creation	`sinfo_rec_jitter`	reduce psf standard stars,
		flux standard starts and object data

6.2.4 Lazy Mode

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

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

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

6.3 Workflow Steps

6.3.1 Step 1: Data Organisation And Selection

On clicking the  button on the `Reflex` canvas, the workflow will highlight and execute the `Initialise` actor, which among other things will clear any previous reductions if required by the user (see Section 6.1).

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

   A DataSet lists three different pieces of information for each of its files, namely 1) the file name (including the path), 2) the file category, and 3) a string that is called the "purpose" of the file. The DO uses OCA[7] rules to find the files to include in a DataSet, as well as their categories and purposes. The file category identifies different types of files. A category could for example be `FLAT_LAMP`, `WAVE_LAMP` or `OBJECT_NODDING`. The purpose of a file identifies the reason why a file is included in a DataSet. The syntax is `action_1.action_2.action_3. ... .action_n`, where each `action_i` describes an intended processing step for this file. The actions are defined in the OCA rules and contain the recipe together with all file categories required to execute it (and predicted products in case of calibration data). For example, a workflow might include two actions `FLAT` and `WAVE_LAMP`. The former creates a master flat from raw flat frames, and the later creates (among other products) a wavelength calibration solution frame. The `WAVE_LAMP` action needs raw arc lamp frames and the master flat as input. In this case, these flats will have the purpose `FLATS.WAVE_LAMP`. The same DataSet might also include biases with a different purpose, e.g. `FLATS.OBJECT_NODDING`. Irrespective of their purpose the file category for all these biases will be `FLATS`.

2. Next the `DataSet Chooser` displays the DataSets available in the "Select Data Sets" window[8], activating a vertical scroll bar on the right if necessary (see Figure 4.0.3). Sometimes you will want to reduce a subset of these DataSets rather than all DataSets, and for this you may individually select (or de-select) DataSets for processing using the tick boxes in the first column, and the buttons `Select All` and `Deselect All` at the bottom left.

   You may also highlight a single DataSet in blue by clicking on the relevant line. If you subsequently click on `Inspect Highlighted`, then a "Select Frames" window will appear that lists the set of files that make up the highlighted DataSet including the full filename and path for each file (hovering the mouse over the filename), the file category (from the FITS header), and a selection tick box in the right column (see Figure 6.3.1). The tick boxes allow you to edit the set of files in the DataSet which is useful if it is known that a certain calibration frame is of poor quality (e.g: a poor raw flat-field frame). The list of files in the DataSet may also be saved to disk as an `ASCII` file by clicking on `Save As` and using the file browser that appears.

   By clicking on the line corresponding to a particular file in the "Select Frames" window, the file will be highlighted in blue, and the file FITS header will be displayed in the text box on the right (see Figure 6.3.1), 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

[7] 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`

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

FITS_VIEWER.

To exit from the "Select Frames" window, click Continue, and 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.

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

6.3.2 Step 2: Recipe execution

Once the datasets to be reduced are selected, press the Continue button on the dataset organised window to proceed with the data reduction. The workflow will automatically execute the pipeline recipes and construct the .sof files to feed the pipeline recipes with. Each sof file will be saved in the BOOKKEPING_DIR directory (and subdirectory within it), depending on the recipe it is associated to and the execution time. The pipeline parameters can be changed as shown in figure 6.2.2.

6.3.3 Step 3: Final products

Once a dataset is reduced (i.e. when the sinfo_rec_jitter recipe is terminated), a window containing the list of science product pops up. Each file can be inspected with the selected fits viewer. Final science products will be stored in the END_PRODUCTS_DIR, and sorted by execution time and dataset identifier (i.e. the name of the science frame the dataset is for). Default names for the science products are:
<TARG name>AO_PERFORMANCE.fits, <TARG name>_COADD_OBJ.fits,
<TARG name>_ENC_ENERGY.fits, <TARG name>_OBS_OBJ.fits, and
<TARG name>_STD_STAR_SPECTRA.fits, and <TARG name>_SKY_MED.fits. We refer the user to the SINFONI pipeline manual for the description of these files.

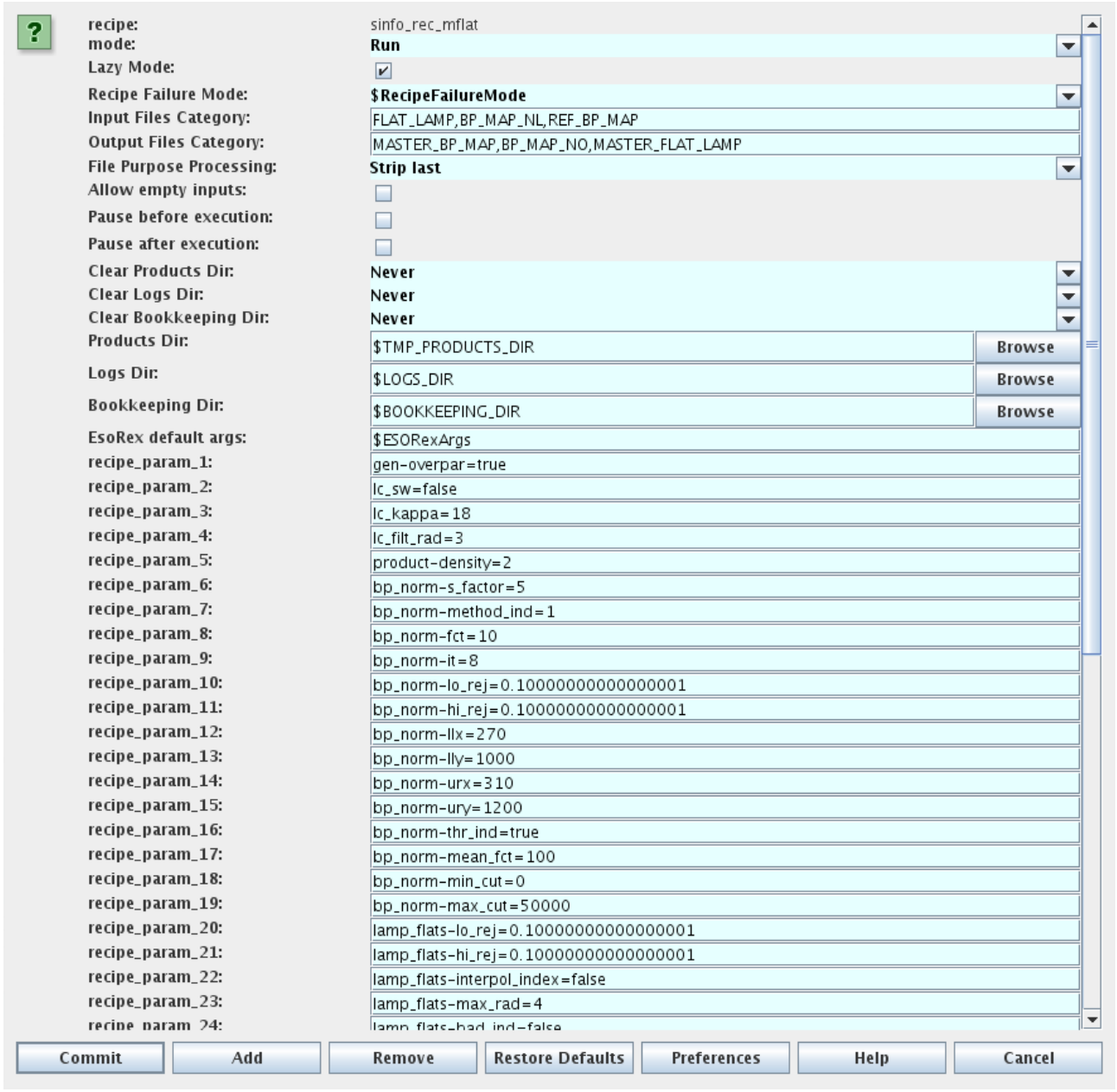


Figure 6.2.2: *The "Edit parameters" window for a typical* `RecipeExecuter` *actor, the* `sinfo_rec_mflat_1` *actor which runs the* `sinfo_rec_mflat` *pipeline recipe.*

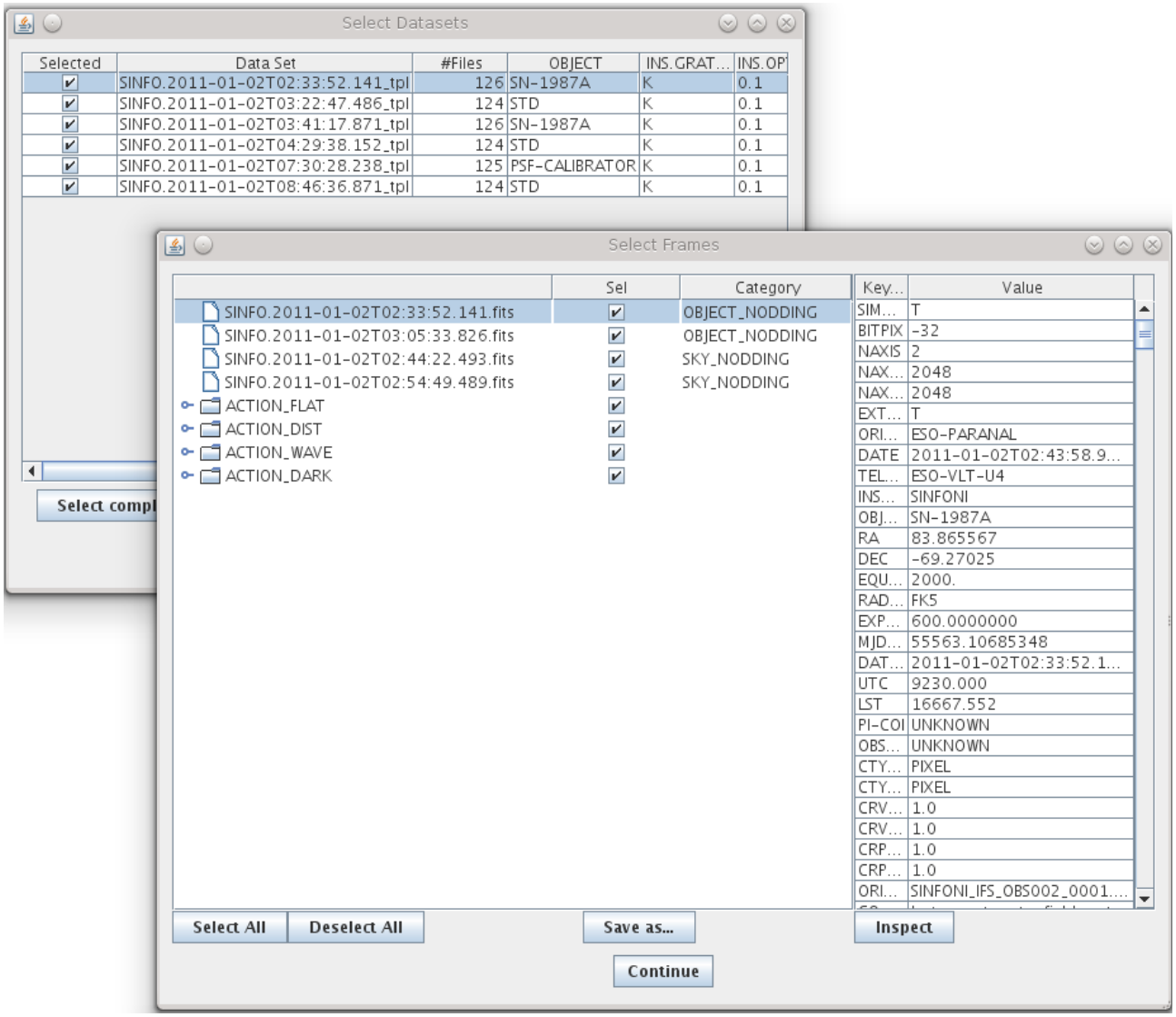


Figure 6.3.1: *The "Select Frames" window with a single file from the current Data Set highlighted in blue, and the corresponding FITS header displayed in the text box on the right. Hidden partially behind the "Select Frames" window is the "Select DataSets" window with the currently selected DataSet highlighted in blue.*

7 Troubleshooting

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

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

   This directory is setup automatically to point to the calibration database provided with the pipeline and in principle shouldn't be changed. However, if static calibration data are present in the RAWDATA_DIR (e.g. calibrations are downloaded from the archive, or copied from ESO-DVD distribution) [9], then you might have to set this directory equal to RAWDATA_DIR (otherwise an obsolete static calibration file may be selected instead of the most appropriate one).

[9] note that the data distributed over DVDs are also available via the ESO science archive, with calibraiotns selected via the CalSelector)

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.

- **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 automaticaly 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.

- **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 at the end of the reduction will list all the previous reductions of a particular data set. 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.

Forchì V., 2012, Reflex User Manual, VLT-MAN-ESO-19000-5037, Issue 0.7, *ftp://ftp.eso.org/pub/dfs/reflex/ReflexUserManual-3.1.pdf* 17, 20, 27, 28

SINFONI Pipeline User Manual, VLT-MAN-ESO-19500-3600, Issue 16.020

The SINFONI Pipeline, proceedings ADA-2006, *http://www.eso.org/sci/software/pipelines/*.