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

UVES EDPS-GUI tutorial

1 rev. 20260529

Issue 1

Date May 29 2026

Prepared:	Burkhard Wolff	May 29 2026	.....
	Name	Date	Signature
Approved:			.....
	Name	Date	Signature
Released:			.....
	Name	Date	Signature

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1	29/05/2026	all	First public release for 6.5.3

This page was intentionally left blank

Contents

1 Introduction

1.1 Scope

This document describes how to reduce UVES data with the `edps-gui` (Graphic User Interface), the dashboard of the ESO Data Processing System (`EDPS`), which the recommended interface to reduce data from ESO telescopes. Details on the UVES data reduction stream and how to configure the reduction to meet specific scientific needs are also given.

For a more extensive documentation on the `edps-gui` itself, consult the dedicated manual here.

For a description of the UVES pipeline itself, consult the pipeline manual available at:
https://www.eso.org/sci/software/pipe_aem_table.html.

Note: this tutorial refers to:

- UVES instrument pipeline named `uves`, version 6.5.3.
- UVES workflow: `uves.uves_wkf`
- `EDPS` version 1.5.7.
- `edps-gui` version 1.0.

1.2 What is `EDPS`?

The ESO Data Processing System (`EDPS`) is a framework to run ESO's data processing pipelines and it is meant to eventually replace the previous ESOReflex environment. The general principles of `EDPS` have been described by Freudling, Zampieri, Coccato et al. [2024, A&A, 681, A93]. Please refer to that paper if you have used `EDPS` for research resulting in a scientific publication.

Each of ESO's data processing pipeline consist of a series of standalone programs called *recipes*. Each recipe is designed to process certain type(s) of input data. The processing of these input data typically requires a range of auxiliary files such as calibration files. `EDPS` is designed to select appropriate input data for the different recipes of a pipeline, and execute them in sequence. This is done by specifying for each pipeline the workflow for organizing data and executing the recipes. This workflow can the used to process a set of data fully automatically.

1.3 Main concepts

`EDPS` is an environment designed to execute the recipes of an instrument pipeline according to a series of instructions. The main concepts in `EDPS` are:

- **Workflow** and **reduction cascades**. A workflow is a series of instructions designed to reduce data with an instrument pipeline in potentially multiple ways, by carrying on a sequence of tasks. Each workflow can define multiple reduction cascades, depending on the scientific needs. For example, the same workflow

can be used to process data following different strategies that trigger different reduction steps (e.g. in one strategy flux calibration can be omitted) or different end-points (e.g., combine different science exposures, or stop after the reduction of individual exposures without combining them). Each of these "strategies" defines a "reduction cascade".

- **Task**, **jobs**, and **recipes**. A task is an element in the workflow that performs a given step of the data reduction cascade. Tasks are often associated to a recipe of the underlying instrument pipeline. A jobs is a work unit in a processing environment, that runs a recipe on a set of input data with a set of recipe parameters. A single task can generate several jobs: for example, a "bias" task, can generate multiple jobs, each of the running the bias recipe on a different set of input files.

- **Dataset**. A dataset is a collection of files, that are needed to perform the data reduction as specified by the workflow. It consists, for example, of one or more science files plus the calibrations needed to process them. In `EDPS`, datasets have an hierarchical structure, which highlights the connections between the various files and tasks (e.g., task A is an input to task B).

- **Target** and **Target category**. The "target", or the "target task" is the end point of the reduction cascade. When specifying a target, `EDPS` will process all and only the files needed to execute it. For example, if my target is "science", and the science files need the bias files, `EDPS` will process only the biases that have been selected to process those science files; then it processes the science using the product of the bias reduction. However, if my target is bias, then `EDPS` will process all and only the bias files, regardless they are not used by any science. In this case, `EDPS` does not processes the science, as it has already reached the end reduction point (e.g., process all biases). The "Target category" is a group of targets that have similar purposes. For example, the target category "science", includes all the tasks that deliver final scientific products, the target category "qc1calib" includes all and only the tasks that processes calibrations (e.g., bias, flat fields, standard stars).

1.4 Installation

1.4.1 Prerequisites

Prerequisites for a well functioning installation of `EDPS` and `EDPS`-gui are:

- Recent Firefox or Chrome browser, Python 3.11 or higher (but there are issues with Python 3.14).

- At least one ESO pipeline with EDPS workflow should be in your system. To install the desired ESO pipelines, follow the instructions in the ESO pipelines pages. NOTE: the `apptainer` installation method is currently not supported. After the installation, the esorex command must be in the path. To test whether the installation was successful, type

  ```
  esorex --recipes
  ```

  A list of available recipes should appear.

- Install `graphviz`, `fv`, and `ds9`, which have to be included in the system path (definying aliases not enough). On linux, `Graphviz` can be easily installed via:

```
sudo apt install graphviz (Debian, Ubuntu)
sudo dnf install graphviz (Fedora)
```

Check the , `Graphviz` webpage for installation instructions for other OS.

`fv` and `ds9`, are optional. To install them, follow the instructions in corresponding webpages. You can test whether these three packages are installed and their path are correctly set by typing on a terminal:

```
dot -V
fv  -version
ds9 -version
```

1.4.2 Installation steps

To install EDPS follow these steps:

- Create a new Python virtual environment and activate it:

  ```
  python3 -m venv edpsgui
  . edpsgui/bin/activate
  ```

  Make sure the python3 version is 3.11 or higher, but not 3.14.

- Install the required packages:

  ```
  pip install --extra-index-url \
      https://ftp.eso.org/pub/dfs/pipelines/repositories/stable/src \
      edps edpsgui edpsplot adari_core
  ```

To run the `edps-gui` type from a terminal (with the active environment):

```
edps-gui
```

Important note. The first time `edps-gui` is executed, you will be asked to specify the directory where the reduction products (fits files and quality plots) will be stored. The default location is `$HOME/EDPS_data`. During the first execution, a configuration file named `application.properties` will also be saved in the directory (newly created) `$HOME/.edps`.

2 Reducing demo data

Follow this procedure to quickly reduce UVES demo data. We assume that the `EDPS`, `edps-gui`, the UVES pipeline and its associated demo data are installed in your system. For general instructions on how to install `EDPS` and the pipeline, see Section 1.4 or please visit: https://www.eso.org/sci/software/pipe_aem_main.html.

2.1 Setting the workflow

Proceed as follows:

1. If not done already, activate the `EDPS` virtual environment, defined during installation (Sect. 1.4).
2. Start the `edps-gui` dashboard by typing:

   ```
   edps-gui
   ```

   The `edps-gui` dashboard will start in a browser window (Figure 1).

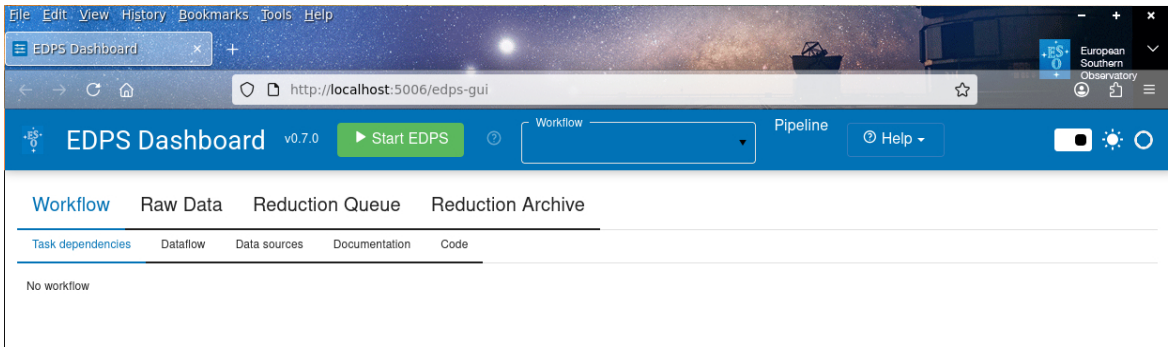

Figure 1: The empty `edps-gui` Dashboard; the underlying `EDPS` engine has not yet been started and no workflow has been loaded.

3. Optionally, before starting `EDPS`, one can specify new settings by pressing Help –> Settings (Figure 2).

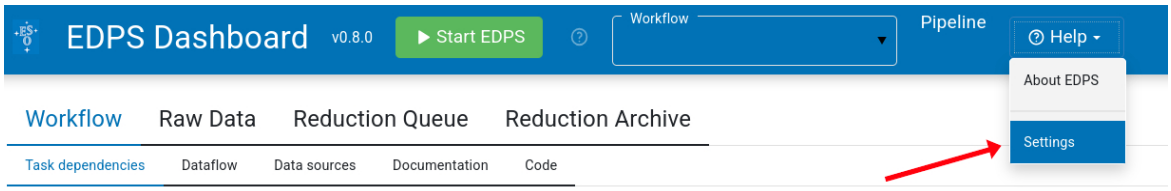

Figure 2: The “Help” –> “Settings” menu.

4. On the browser window with the dashboard, press the button ‘Start EDPS‘.
5. Choose the `uves.uves_wkf` workflow from the list in the ’Workflow’ field. The workflows offered in this selector depend on the installed pipelines. The graphic workflow representation will appear as in Figure 4.

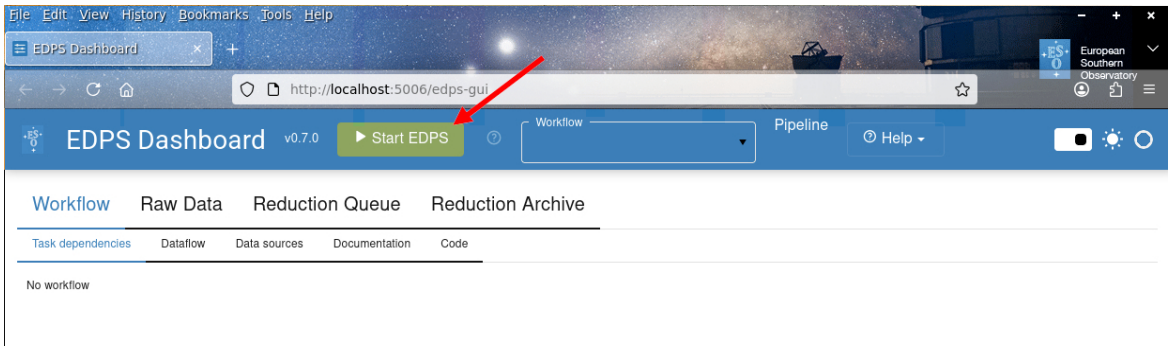

Figure 3: The “Start EDPS” button.

2.2 Selecting the input data

1. Press ‘Raw Data‘ to enter the corresponding tab, as in Figure 5.

2. Press ‘Select Inputs‘. A selection window will appear that allows to select data that are stored on a local disk (Figure 6).

3. (Optional). Select the reduction target, configure the workflow parameter and specify the association preferences. These steps are optional. For more information see Section 6.

4. Press ‘Create Datasets‘. A list of datasets appears, one line for each set of science data (Figure 7).

5. Choose the datasets that should be processed (Figure 8)

   and send them to the data reduction queue by pressing ‘Submit to Reduction Queue‘. Note that this action does not start the reduction automatically.

2.3 Start the reduction

1. Press the ‘Reduction Queue‘ tab (Figure 9).

2. Select the datasets you’d like to reduce.

3. (Optional). Configure the workflow and recipe parameters by pressing the wheel button  to open the configuration editor. See Section 6.2 for more information on the configuration editor.

4. Press the ‘Reduce‘ button (Figure 10). The selected data will now be processed with the configured parameters.

Congratulations! You reduced your first data with the `EDPS` dashboard! All the reduced data are saved in the EDPS_data directory specified when executing `edps-gui` for the first time.

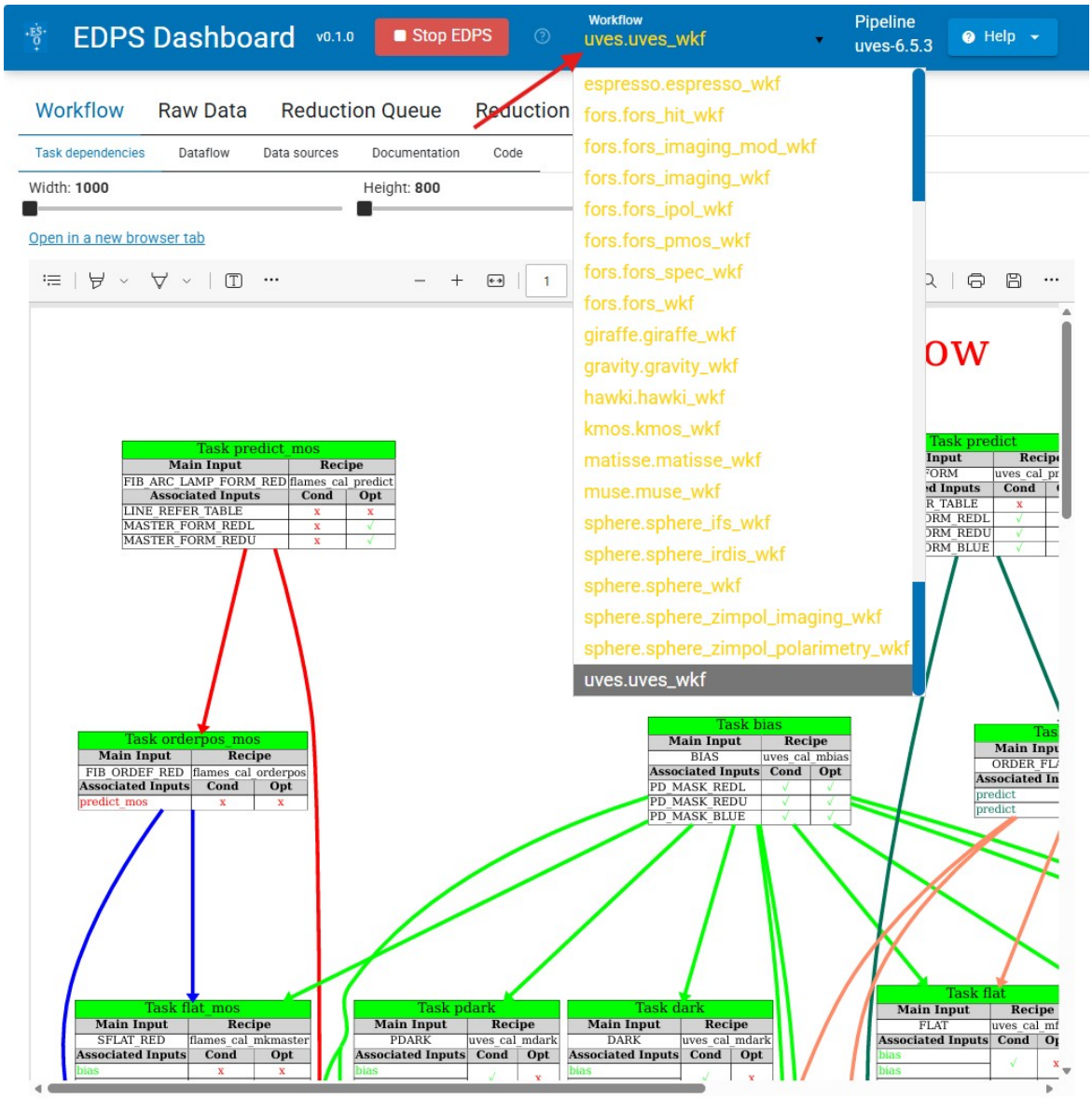

Figure 4: The `edps-gui` with the UVES workflow loaded.

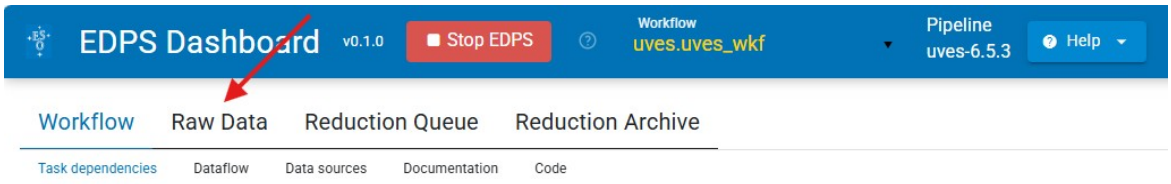

Figure 5: How to select RAW data Tab.

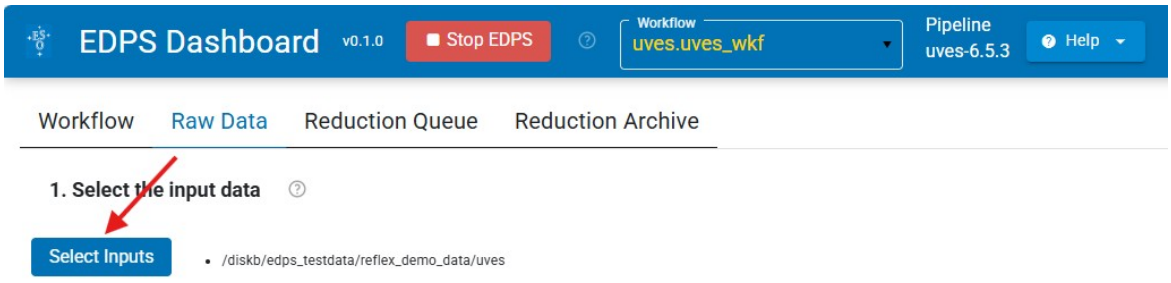

Figure 6: How to select input data.

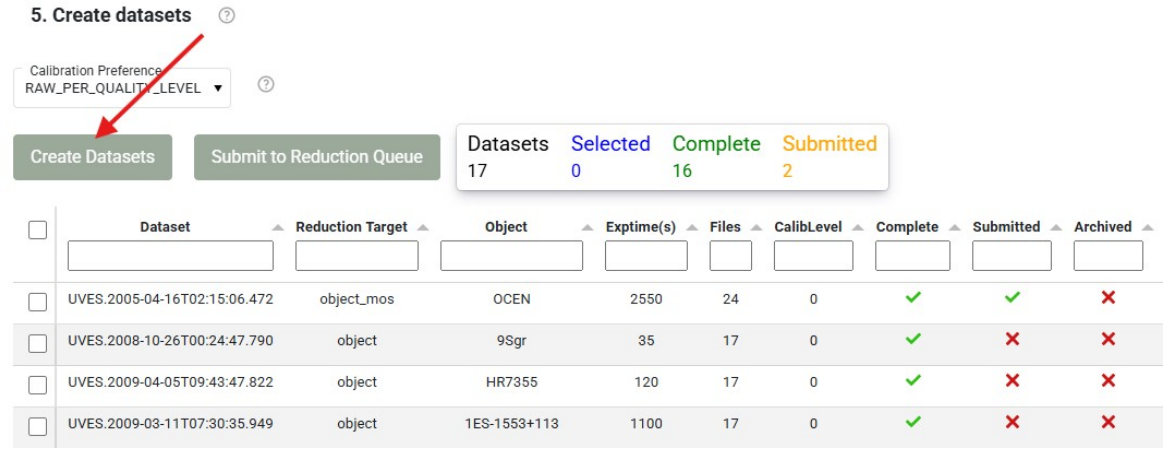

Figure 7: How to inspect the input data directory to create datasets.

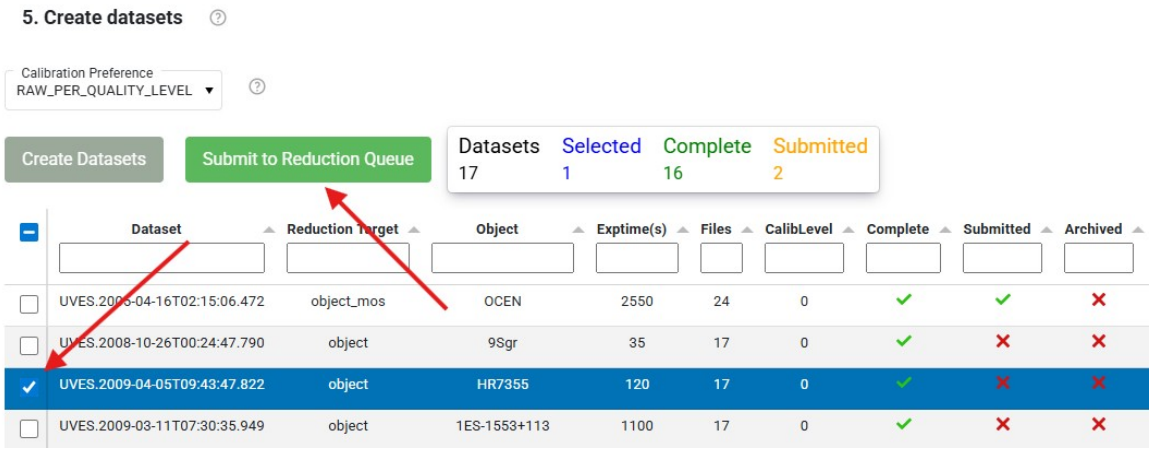

Figure 8: How to send the selected datasets to the Reduction Queue for processing.

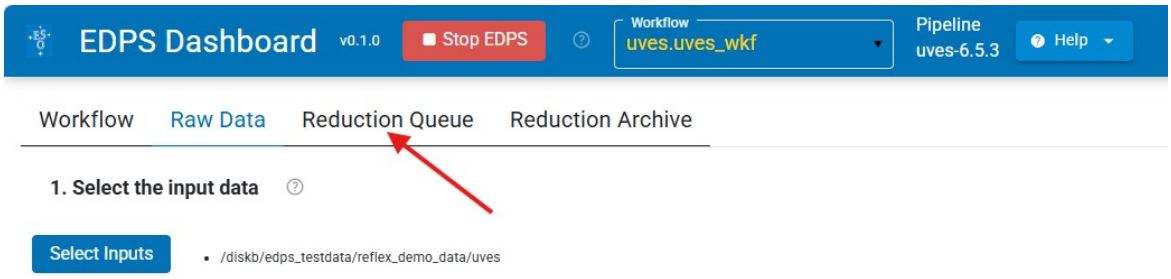

Figure 9: How to select Reduction Queue tab.

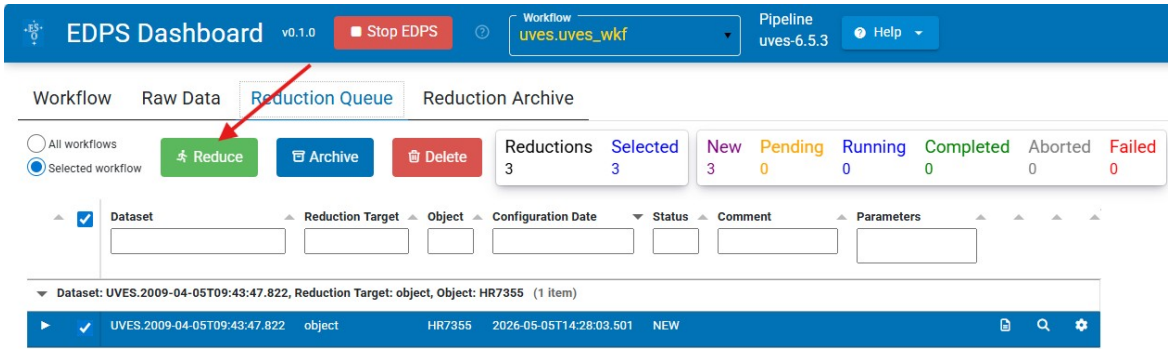

Figure 10: Reduce.

2.3.1 Quality plots

Almost all processing tasks can display the input raw frames and the products in the so called "quality plots", which can be inspected from the ‘Reduction Queue‘ window. Those associated for the main product can be inspected by pressing the magnifying glass symbol at the right side of each dataset. To inspect those associated to each individual job (if created),

- Expand the desired dataset by pressing the black arrow on its left. The list of jobs will appear with the associated status (COMPLETED, RUNNING, PENDING, MISSING, ABORTED, FAILED)
- Press the magnifying glass symbol at the right side of the job you want to inspect. Only plots for completed jobs can be inspected.

2.4 Exporting the final products

Completed reductions can be ’Archived’ (i.e. declared ’completed’ because no more work is needed) and removed from the Reduction Queue. Additionaly, even if all products for all tasks are saved in the EDPS_data directory, the most important producs can be ‘exported’ to a desired location.

To do so, proceed as follows:

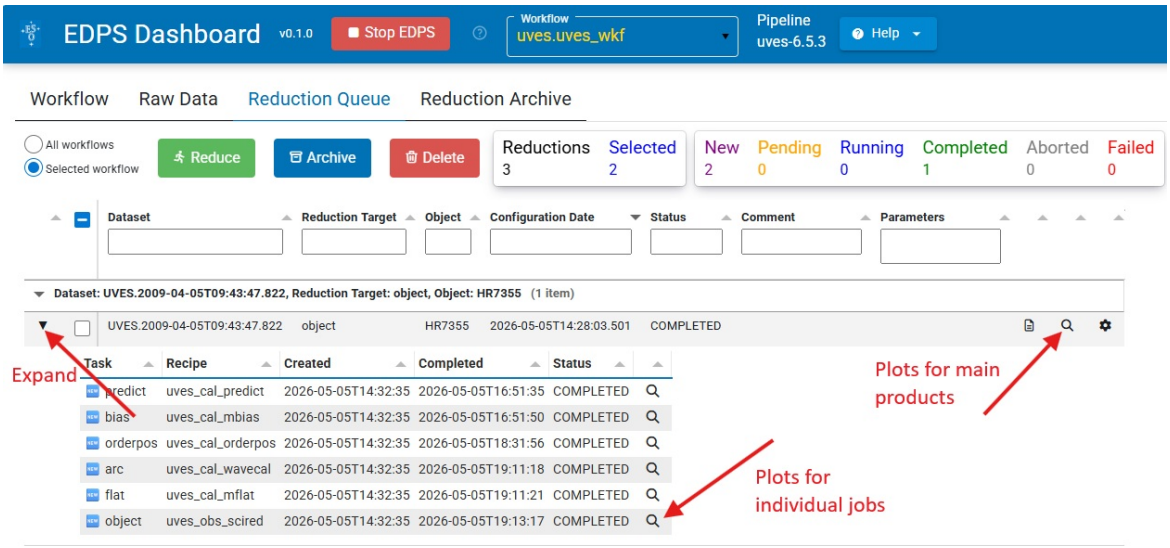

Figure 11: How to look for quality plots from the Reduction Queue tab.

1. In the ‘Reduction Queue‘ tab, select the dataset and the dataset for which you want to export the final products, and press the ’Archive’ button.

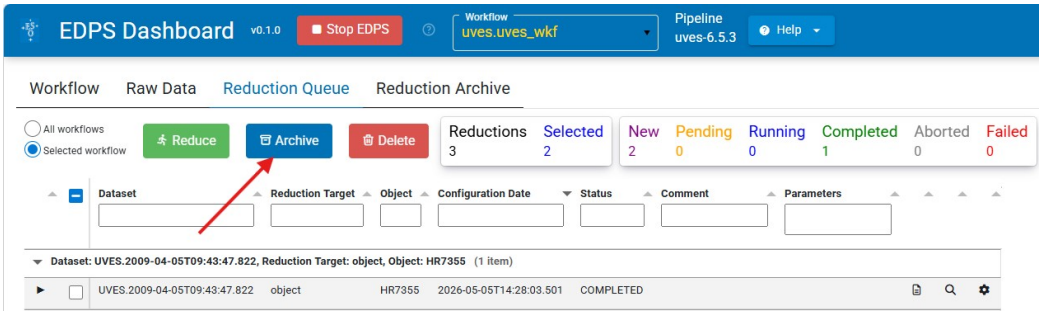

Figure 12: How to archive a completed reduction from the Reduction Queue tab.

2. Go in the Reduction Archive tab and click on the ’Export’ button. A new tab window appear where you can indicate the directory you want to copy your final products; finally press "Export" to copy the data.

Exported products are organized by ‘DATASET‘ (named as the first scientific exposure of the dataset), and ‘TIMESTAMP‘ (time of start of reduction)

The final products saved in the specified directory are:

- The 1D extracted spectra. One file for the blue spectroscopic arm, file name starting with ‘SPECTRUM_1D_BLUE_’. And two files for the red spectroscopic arm, one file per chip, file names starting with ‘SPECTRUM_1D_REDL_’ and ‘SPECTRUM_1D_REDU_’, respectively.
- If individual exposures of the same target have been combined, then the combined spectra have file names starting with ‘SPECTRUM_COMBINED_’.

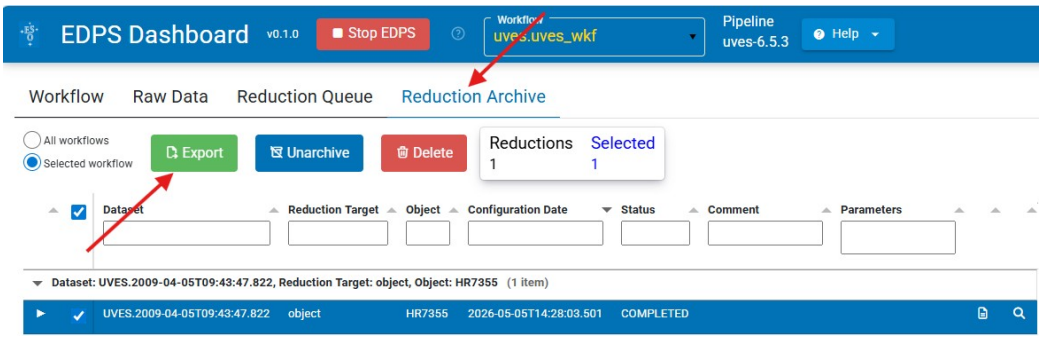

Figure 13: The reduction archive tab. This table contains all the different configurations of datasets that are declared "finished" and removed from the Reduction Queue. From this page, the user can export the most important files into a desired local directory.

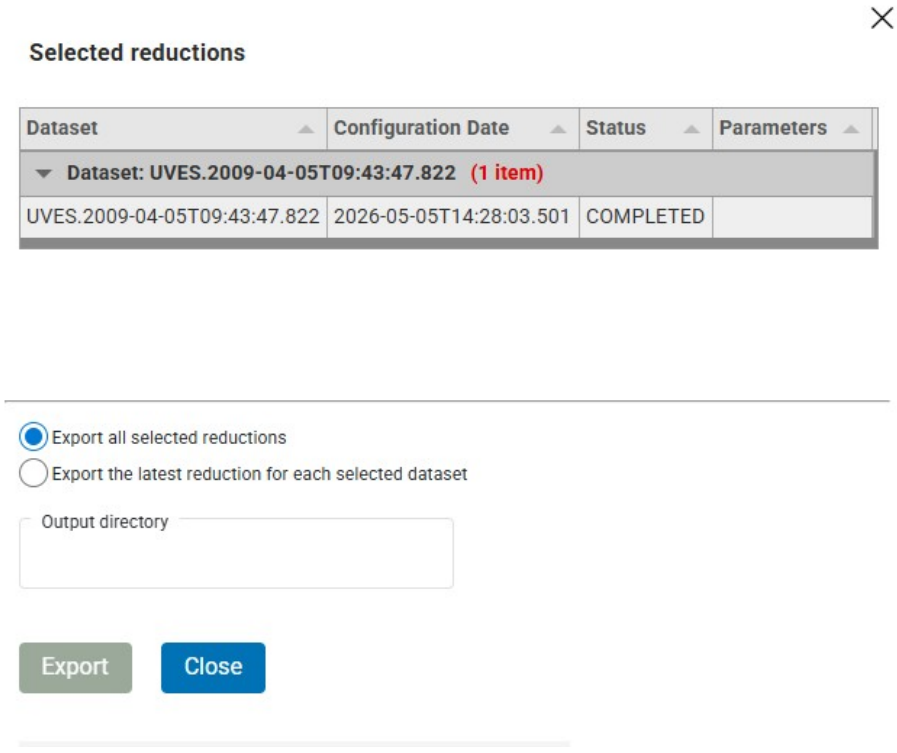

Figure 14: The EXPORT dialogue window, where the user can decide which reduced configuration to save and where.

3 The UVES data reduction flow.

The overall data flow of the UVES pipeline is displayed in Figure 15.

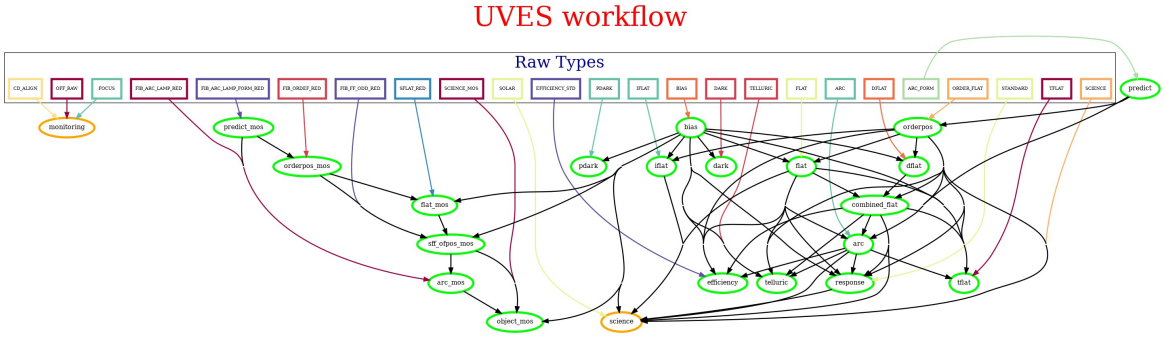

Figure 15: The data reduction cascade of the UVES workflow.

The reduction cascade is organized in tasks, which represent well-defined steps in the process. Tasks can be grouped inside sub-workflows. Each task runs a recipe; the detailed description of the algorithms, input, outputs and recipe parameters used in each recipe are available in the pipeline manual. Here, we present only the description of most important features.

The `uves.uves_wkf EDPS` workflow is designed to execute the tasks that deliver the final reduced data cube for each dataset. It can be either the product of a single exposure, or the combination of multiple exposures. Only calibrations needed by the selected the scientific exposures are processed.

It is possible to set `EDPS` to perform the data reduction until a certain step of the reduction chain (e.g. to reduce only standar stars, or only flat fields). This is done by specifying the desired tasks in the field **Select reduction target** of the **Raw Data** tab.

The reduction steps of the `uves.uves_wkf` workflow are listed below. Before starting the reduction, the parameters of the recipes associated to each task can be configured by pressing the button  close to each dataset configuration. See for more info on the configuration editor 6.2

UVES can be operated standalone or together with the FLAMES instrument. If a destinction between these two operational modes is needed in this tutorial, then the former is referred to as ‘UVES (Slit)’ and the latter as ‘UVES-Fibre’.

The two spectroscopic arms of UVES, i.e., the blue and the red arm, are processed separately. For the blue arm, the pipeline recipes create one product per product category whereas for the red arm, usually two products are created, one for the ‘lower’ and one for the ‘upper’ chip. These different products can be distinguished by the tags `_BLUE`, `_REDL`, and `_REDU` in their `PRO CATG` header keywords. For example, a complete set of raw blue and red bias calibrations gives three master calibrations having `PRO CATG` values of `MASTER_BIAS_BLUE`, `MASTER_BIAS_REDL`, and `MASTER_BIAS_REDU`, respectively. In the following description, `<CHIP>` can be one of `BLUE`, `REDL`, or `REDU`.

4 UVES (Slit) data reduction

4.1 Master bias

Task: `bias`. Recipe: `uves_cal_mbias`. Output: `MASTER_BIAS_<CHIP>`.

This task creates a master bias for each chip. A minimum of three input frames is required.

4.2 Initial order and wavelength solution

Task: `predict`. Recipe: `uves_cal_predict`. Output: `LINE_GUESS_TAB_<CHIP>` and `ORDER_GUESS_TAB_<CHIP>`.

This task creates the initial guess for the order and wavelength identification.

4.3 Order positions

Task: `orderpos`. Recipe: `uves_cal_orderpos`. Output: `ORDER_TABLE_<CHIP>`.

This task creates the final identification of the Echelle orders.

4.4 Master flat field

Task: `flat`. Recipe: `uves_cal_mflat`. Main output: `MASTER_FLAT_<CHIP>`.

This task creates the master flat for the 'regular' lamp flat fields (which have `DPR TYPE = LAMP,FLAT`). The workflow also supports the usage of flat fields taken with the deuterium lamp (`DPR TYPE = LAMP,DFLAT`) and of flats where the iodine cell is inserted into the light path (`DPR TYPE = LAMP,IFLAT`).

The deuterium lamp flats are recommended for the spectral region shorter than 350 nm. They are taken as supplementary calibrations for the UVES blue arm central wavelength setting of 346 nm. If deuterium flats are present in the data set, then they are combined with the regular lamp flats into a combined master flat that provides much better S/N shorter than 350 nm. The tasks `dflat` and `combined_flat` are executed in addition to the `flat` task.

The iodine cell flats are provided only for the red 600 nm setting. They contain the the iodine cell absorption lines superimposed on the flat-field spectrum. The task `iflat` is executed then.

4.5 Wavelength calibration

Task: `arc`. Recipe: `uves_cal_wavecal`. Output: `LINE_TABLE_<CHIP>`.

The wavelength calibration is determined from Th-Ar arc-lamp exposures.

4.6 Response curve

Task: `response`. Recipe: `uves_cal_response`. Output: `INSTR_RESPONSE_FINE_<CHIP>`.

Flux calibration can be obtained either by processing an observation of a flux standard or by applying a master response curve. If a raw standard star calibration is present in the data set, then this observation is processed by the `response` task. The resulting response curve is then used during processing of the science data. Master response curves are available for a number of setups. If a raw standard star observation is not present but a matching master response is found, then the latter is used for the flux calibration. If neither a raw standard nor a master response is present in the data set, then the extracted science spectrum is not flux calibrated.

4.7 Spectrum extraction

Task: `object`. Recipe: `uves_obs_scired`.

This task extracts the spectrum and wavelength calibrates it. If a response curve is available, then the spectrum is also flux-calibrated. The main output files are listed in Table 1.

Product	**Description**
`RED_SCI_<TYPE>_<CHIP>`	extracted, flat-fielded, wavelength-calibrated, order-merged, and sky-subtracted science spectrum
`ERRORBAR_SCI_<TYPE>_<CHIP>`	uncertainty on `RED_SCI_<TYPE>_<CHIP>`
`FLUXCAL_SCI_<TYPE>_<CHIP>`	flux-calibrated version of `RED_SCI_<TYPE>_<CHIP>`
`FLUXCAL_ERRORBAR_SCI_<TYPE>_<CHIP>`	uncertainty on `FLUXCAL_SCI_<TYPE>_<CHIP>`

Table 1: Main output products from the `object` task. For the values of `<TYPE>`, see Table 2

The `PRO CATG` depends on whether or not the image slicer and/or the absorption cell has been used. See Table 2.

<TYPE>	**Description**
`POINT`	Regular slit observation
`SLICER`	Observation using the image slicer
`ABSCELL`	Observation using the absorption cell
`SLICABSCELL`	Observation using the image slicer and the absorption cell

Table 2: Possible values for `<TYPE>`

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). In the latter case, output products have an additional tag '`2D`' in their `PRO CATG` values. The recipe parameter that controls the extraction method is `reduce.extract.method`. More details on spectrum extraction can be found in Sect. 6.3.

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 the recipe parameter `reduce.skysub` is set to the default value TRUE).

4.8 Phase 3 compatible science product

Task: `science_slit`. Recipe: `uves_utl_idp`.
Output: `RED_SCI_<TYPE>_BLUE` or `RED_SCI_<TYPE>_RED` or `FLUXCAL_SCI_<TYPE>_BLUE` or `FLUXCAL_SCI_<TYPE>_RED`.

This task takes the output from `object` and converts it into a single product that is compatible with the ESO Phase 3 standard. The files are converted from 1D image format into a table with columns for the wavelength, the flux, the error, etc. Also, the two spectra for the red arm are merged into a single file.

4.9 Spectrum combination

Task: `combine_spectra`. Recipe: `esotk_spectrum1d_combine`.
Output: `ESOTK_SPECTRUM_IDP_FORMAT`.

This task stacks two or more spectra from the same target (and with the same setup). It requires as input the output from `science_slit`.

5 UVES-Fibre data reduction

In case of UVES-Fibre, <CHIP> can be 'REDL' or 'REDU'.

5.1 Master bias

Task: `bias`. Recipe: `uves_cal_mbias`. Output: `MASTER_BIAS_<CHIP>`.

This task creates a master bias for each chip. A minimum of three input frames is required.

5.2 Initial order and wavelength solution

Task: `predict_mos`. Recipe: `flames_cal_predict`.
Output: `FIB_LIN_GUE_<CHIP>` and `FIB_ORD_GUE_<CHIP>`.

This task creates the initial guess for the order and wavelength identification.

5.3 Order positions

Task: `orderpos_mos`. Recipe: `flames_cal_orderpos`.
Output: `FIB_ORD_TAB_<CHIP>` and `FIB_ORDEF_<CHIP>`.

This task creates the final identification of the Echelle orders.

5.4 Master flat field

Task: `flat_mos`. Recipe: `flames_cal_mkmaster`. Main output: `MASTER_SFLAT_<CHIP>`.

This task creates the (slit) master flat fields for three different positions on the chip. Therefore, there are three files for each category.

5.5 Fibre flat field

Task: `sff_ofpos_mos`. Recipe: `flames_cal_prep_sff_ofpos`.
Main output: `FIB_ORDEF_TABLE_<CHIP>`.

This task is used to determine the fibre order table and to construct several master calibrations needed to extract the science fibres.

5.6 Wavelength calibration

Task: `arc_mos`. Recipe: `flames_cal_wavecal`. Output: `FIB_LINE_TABLE_<CHIP>`.

The wavelength calibration is determined from Th-Ar arc-lamp exposures.

5.7 Spectrum extraction

Task: `object_mos`. Recipe: `flames_obs_scired`.

This task extracts the spectra for each fibre. The main output files are listed in Table 3.

Product	Description
`MWXB_SCI_<CHIP>`	extracted, flat-fielded, wavelength-calibrated, order-merged, and fibre throughput corrected science spectrum; one file for each fibre
`ERR_MWXB_SCI_<CHIP>`	error for `MWXB_SCI_<CHIP>`

Table 3: Main output products from the `object_mos` task

6 Overview of all the data reduction configuration options

6.1 Selection of most appropriate calibrations

By default, EDPS associates raw calibrations to the reduction process. It is also possible to use pre-processed calibrations (a.k.a. master calibrations) if available, in order to speed up the reduction. The preference can be specified in the Raw Data tab, before creating the datasets.

Possible values of the Calibration Preferences are:

- **raw_per_quality_level**: At equal quality of reduction, association of raw calibrations is preferred. This is the default.
- **master_per_quality_level**: At equal quality of reduction, association of master calibrations is preferred.
- **raw**. Association of raw calibration is preferred, despite the quality of results.
- **master**. Association of master calibration is preferred, despite the quality of results.

When master calibrations are used, the reduction step needed to process raw calibrations are not executed. The reduction then moves directly to the process of scientific exposures.

For example, if reduction speed for a quick check is preferred over a high quality reduction, one can select "master". In this case, old master calibrations are associated even if there are raw calibrations closer in time (and therefore more likely to ensure better quality products).

The quality level that the selected calibrations deliver is indicated close to each dataset in the ‘Raw input‘ tab, under the colum ‘CalibLevel‘. CalibLevel=0 indicates that calibrations that follow the rules of the instrument calibration plans have been selected. The higher the number, the poorer the quality of the products.

6.2 Configuration of parameters: the configuration editor

The data reduction of each dataset can be configured according to the scientific needs using an appropriate configuration editor.

The EDPS workflows contain two types of parameters and they both have default values that can be modified to improve the data reduction.

- **Workflow parameters** (for some workflows only) are global and they are applied to the entire workflow. They are accessible both in the ‘Raw Data‘ tab, prior to the creation og a dataset, and in the ‘Reduction Configuration‘ editor, in the ‘Reduction queue‘ tab. Note: some workflow parameters were already configured before creating the dataset and sending it to the reduction queue. Here, they can be changed again. Please, note that the parameters have an effect only on the files that are already in the dataset. If one specifies a parameter that should include extra files in the dataset (e.g., the inclusion of more calibrations), files are not added and the reduction might fail. If you need to change a parameter that modifies the dataset content, please go back to the Raw data tab and create a new dataset.

- **Recipe parameters** are specific to the individual recipes and can be configured per task. They are accessible in the ‘Reduction Configuration‘ editor, in the ‘Reduction queue‘ tab.

This editor allows to configure the data reduction for a given dataset by specifying workflow and recipe parameters.

Note: some workflow parameters were already configured before creating the dataset and sending it to the reduction queue. Here, they can be changed again. Please, note that the parameters have an effect only on the files that are already in the dataset. If one specifies a parameter that should include extra files in the dataset (e.g., the inclusion of more calibrations), files are not added and the reduction might fail. If you need to change a parameter that modifies the dataset content, please go back to the Raw data tab and create a new dataset.

To open the editor, click on the wheel button  next to the dataset you desire to configure the reduction for. A window with the configuration editor appears as shown Figure 16.

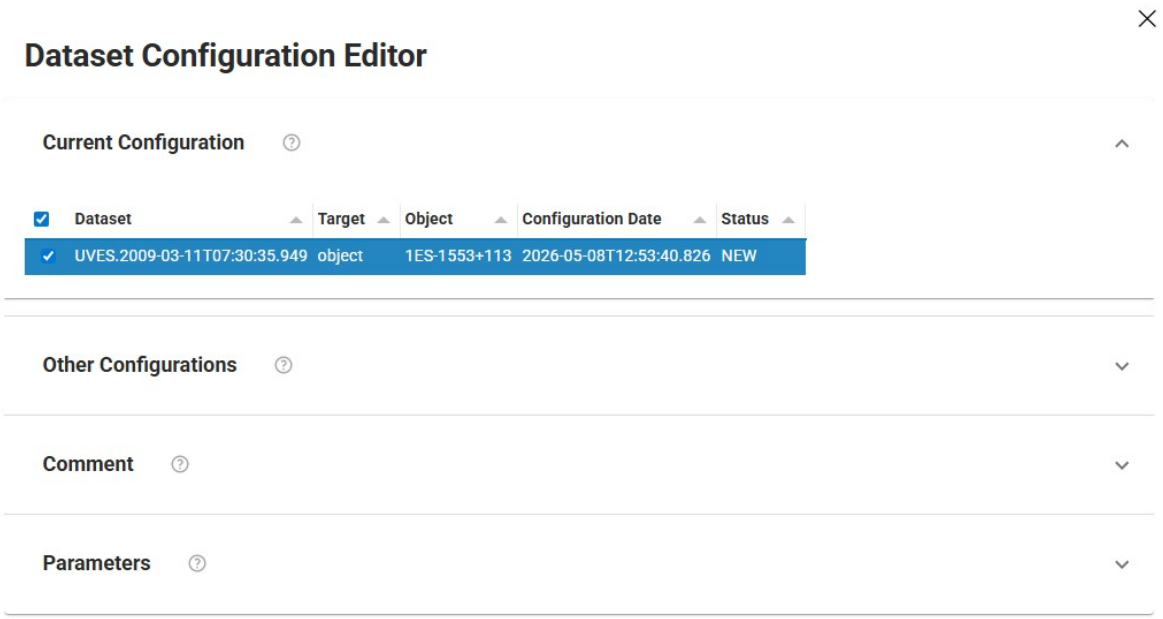

Figure 16: The Reduction Configuration editor. It contains 4 sections, that indicate the current configuration, list of other configurations to set, comments to insert, and the parameters to modify.

The editor is divided into 4 parts, which can be accessed pressing the corresponding expansion arrow.

- **Current configuration.** It indicates the name of the selected configuration for a given dataset (Figure 17).
- **Other configurations.** It allows to specify other configurations, to which the changes shall be copied to (Figure 18.
- **Comment** It allows to specify a comment to describe the configuration. It is possible to append or replace

a comment (Figure 19). Comments can be changed on all configurations. It is possible to save the comment for the current configuration only, or for all the selected configurations.

- **Parameters.** A window as in Figure 20 appears.

  The window allows to:

  – Select the parameter set. A pre-determined list of workflow parameters and recipe parameters for a given use case. For the majority of the cases, the "science" parameter set can be used.

  – Edit the workflow parameters. These are parameters that regulates the reduction strategy, e.g. whether to use a given calibration or not, or to trigger a certain reduction step. Note that if the changes imply that some files not in the dataset are needed, the reduction might fail. In case, go back to the raw data tab, edit the workflow parameters there, and recreate the datasets.

  – Edit the recipe parameters. These are parameters associated to the recipe of a given task. Note: the same recipe parameters can be configured differently for the tasks that run the same recipe. Default parameters are shown (albeit some parameters can be dynamic, e.g. 'EDPS' changes their value depending on the type of input data).

  Change the values according to the needs and then select whether to save it to the current or the selected configurations. Note, complete configurations cannot be modified, new configurations will be automatically created instead.

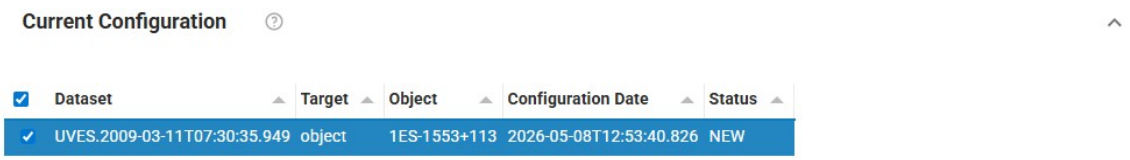

Figure 17: The first part of the Reduction Configuration Editor, that indicates the selected configuration.

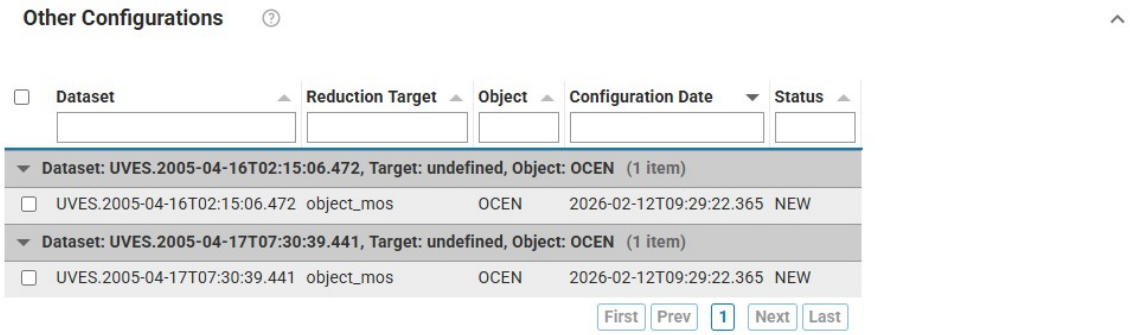

Figure 18: The second part of the Reduction Configuration Editor, that indicates other configurations for which we'd like to apply the changes.

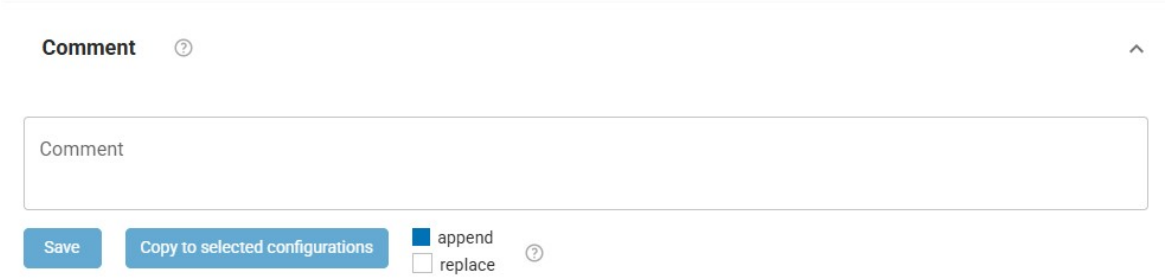

Figure 19: The third part of the Reduction Configuration Editor, that allows to specify a comment to the selected configurations.

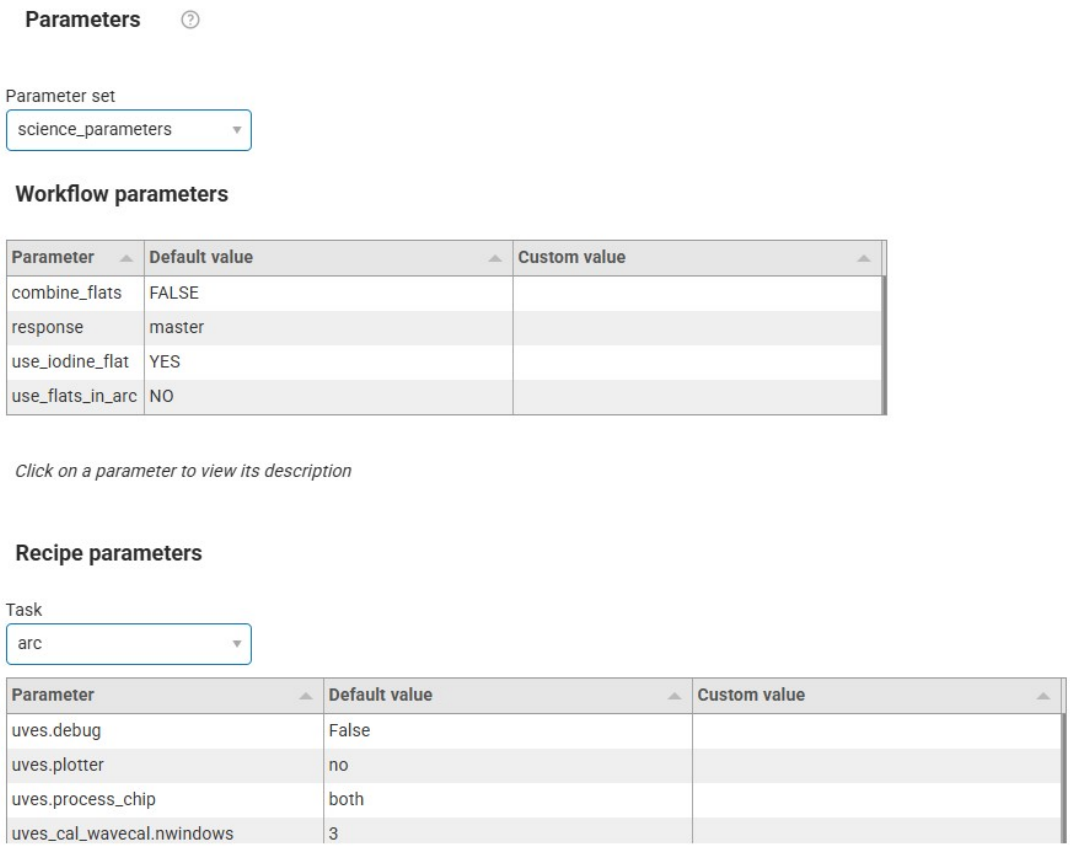

Figure 20: The fourth part of the Reduction Configuration Editor, that allows to specify the parameters sets and the recipe parameter per task. These settings can be applied to the "Selected Configuration" (Fig. 17) or to the "Other Configurations (Fig. 18).

6.3 Optimizing spectrum extraction

The default values of the recipe parameters for the spectrum extraction in task `object` (pipeline recipe `uves_obs_scired`) are optimized towards the extraction of the spectrum of one point source centered on

the slit. The parameters `reduce.slitlength`, `reduce.objoffset`, and `reduce.objslit` allow 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.

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

`reduce.objslit` is 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, whwereas 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 $0.5 \times (\texttt{slitlength} - \texttt{objslit}) \pm \texttt{objoffset}$
will be carved out of the remaining full slit (if any).

If **more than one object trace is present**, the optimal extraction can be used with a suitable combination of `reduce.slitlength` and `reduce.objoffset` to adapt the slit length and position for covering 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.

6.4 Optimizing the initial order and wavelength solution

The initial solution for the order tracing and wavelength calibration as calculated by the task `predict` is usually determined in a very robust way by fitting a physical model of the spectrograph. In rare cases, e.g., after an earthquake, it may be necessary to adjust the model.

A diagnostic graphical report is provided within the EDPS workflow. It can be obtained by clicking on the magnifying glass for the `predict` job and then by clicking on "Graphical reports". If the reported has not been created before, it can be created by checking 'Reduced data' and clicking on 'Create reports'.

The most important plot in the report shows the differences in pixels between the predicted and actual line positions for X and Y. An example is shown in Fig. 21.

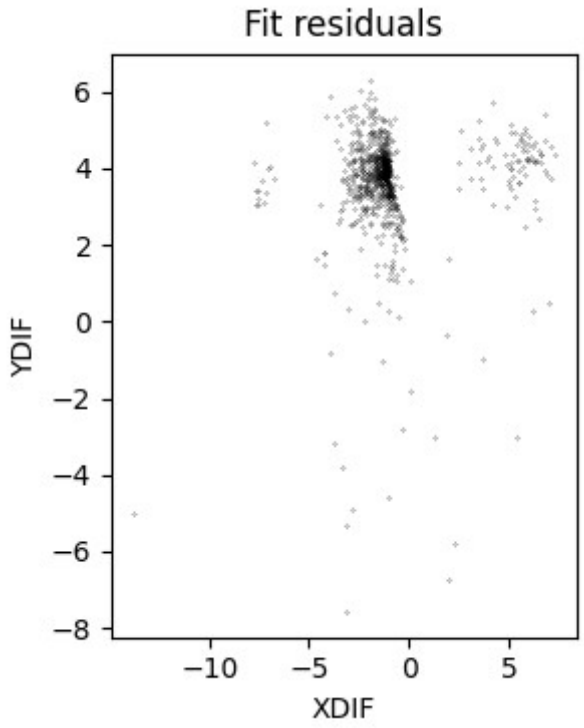

Figure 21: Fit residuals from the `predict` task

For a good fit result, most points in the plot should form a cluster that may be offset from $(0, 0)$ by a few pixels. The presence of a few outliers, like in the example, is usually not an issue. In a bad fit, no or only a very weak clustering could be seen.

Recipe parameters that are useful for improving the fit are:

- `trans_x` and `trans_y`: they shift the centre of the model in X and Y, respectively. Values of 5 to 20 pixels should be sufficient in general.
- `mbox_x` and `mbox_y`: they 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, then larger values of 60 to 80 pixels could be tried. Subsequent adjustment of `trans_x` and `trans_y` should then be performed and `mbox_x` and `mbox_y` should be set back to 40 pixels, followed by a final adjustment of `trans_x` and `trans_y` to arrive at the best initial guess wavelength solution.

- `ccd_rot_angle_off`: it is usually not necessary to add an offset in the CCD rotation angle. But if so, then a small rotation of 0.01 to 0.1 may make improvements.

6.5 Optimizing the wavelength calibration

The wavelength calibration solutions produced by the task `arc` 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. There are a number of parameters for the pipeline recipe `uves_cal_wavecal` that might be tried to further improve the solution. An improvement should result in a smaller RMS for the wavelength residuals, as reported, e.g., in the header keyword `QC LINE RESIDRMS WLU`, without a significant decrease of the number of identified/used arc lines (header keywords `QC LINTOT`, `QC LINSEL`, and `QC LINSOL`). The keywords can be inspected in the `LINE_TABLE_<CHIP>` pipeline products.

Recipe parameters that are useful for improving the fit:

- `range`: Twice the value of this parameter is the window width used for arc line detection (in pixels). Adjusting this parameter influences the number of arc lines that are detected. Varying this parameter in the range of 4 to 12 pixels is sometimes useful for optimizing 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 as well as in the catalogue) is smaller than $\Delta\lambda/\texttt{alpha}$, 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 of $\sim 0.03 \ldots 0.3$ 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 some times 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 of 0.1 ... 1.0 is sometimes useful for optimizing the number of lines used in the wavelength calibration solution.

- `kappa`: The level of sigma-clipping to be performed in the final iteration of the fit of the wavelength calibration model. Suggested values for this parameter are in the range of 3.0 ... 5.0.

6.6 Applying velocity correction

The velocity correction of the extracted spectra can be controlled in the task `object` with:

- `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 is taken from the `HIERARCH ESO DRS BARYCORR` keyword, and the heliocentric value is taken from the `HIERARCH ESO DRS HELICORR` keyword. Note that the application of the correction implies an adjustment of the wavelength values. Therefore, if a combination the spectra of the same target is wanted, the spectra must be interpolated to a common wavelength grid.

6.7 Improving spectral extraction

There are ten recipe parameters for the `object` task which could be used to improve the spectral extraction with the `uves_obs_scired` recipe.

- `reduce.extract.method`: The method to be used for extracting the science spectrum. It may be set to `optimal`, `average`, or `linear`. In most cases, the default optimal extraction is the desired method. If a 2D extraction is wanted, then the parameter needs to be set to `2d`.
- `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 $\sim$ 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. It may be set to `auto`, `virtual`, `gauss`, or `moffat`. 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: `optimal` or `median`. If set to `optimal`, the sky level is included in the optimal extraction model. If 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 $\mathrm{S/N} < 200$, and to 10 for $\mathrm{S/N} > 200$. Sometimes a larger value for the oversampling factor may give a higher S/N in the 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.

- `reduce.objslit`: Object window size (in pixels). Its value 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` except that it applies to REDU data only.

6.8 UVES (slit) tips and trouble shooting

- **To obtain best accuracy for flux calibration, 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 or 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 in 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 data set 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 want 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 item).

- **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: EG21, EG274, Feige110, GD71, GD153, LTT3218, and LTT7987. 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). For a solution to this problem, the points

for the corresponding entry in the static calibration file `RESP_FIT_POINTS_CATALOG.fits` need to be edited.

- **I have data sets of extended objects. How can I reduce them?** UVES observations of extended objects can be reduced with the workflow. It can, however, not be guaranteed that the chosen reduction strategy is appropriate in this case.
- **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 product, the `extract_method` parameter of the `uves_obs_scired` recipe needs to be set to `2d`. See Sect. 4.7. If both one- and two-dimensional spectra are wanted, then the workflow needs to be run twice with two different values for `reduce.extract.method`.
- **I have data sets of objects observed with the image slicer. How can I reduce them?** Reduction of image slicer observations is fully supported within the pipeline and by the workflow. If the object is extended and the full spatial resolution shall be preserved, then the recipe parameter `reduce.extract.method` needs to be set to `2d`.
- **I have data sets of objects observed with the the iodine cell. How can I reduce them?** Reduction of iodine cell data is fully supported. The workflow default is to use the iodine flats for flat fielding the science, if they are present. If this is not wanted, then the raw iodine flats need to be removed from the input data set or the workflow parameter `use_iodine_flat` needs to be set to `NO`.
- **Artifacts present in merged spectra of RED 760 nm (or beyond) settings. Can this be removed?** For data observed in the red settings (760 nm or beyond), we recommend to use `reduce.ffmethod="pixel"` together with `reduce.merge.delt1=14` and `reduce.merge.delt2=4`. Thereby avoiding artifacts 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. The described parameter setting are encoded in the workflow per default.
- **The wavelength calibration fails. How I can recover this problem?** A possible reason may be a large shift in the spectral format (mainly along cross order direction). In this case the physical model plot that is available for the `predict` task usually shows a widely scattered distribution of residuals instead of a clear correlation. 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 optimize the results of the model prediction as described in Sect. 6.4. This should allow to solve the problem.

6.9 UVES-Fibre trouble shooting

There are a few cases where the reduction of UVES-Fibre data may fail. Two of those include:

- **Spectrum extraction failing with error 'Keyword ESO TEL GEOLAT does not exist'.** From time to time the reduction of science frames may fail indicating that the header keyword `ESO TEL GEOLAT` does not exist. This keyword is required to measure heliocentric velocity correction. This issue can only

be fixed by ensuring that the input data have proper FITS headers, in particular containing the keywords RA, DEC, ESO TEL GEOLAT, ESO TEL GEOLAT, UTC, and MJD-OBS.

- **Spectrum extraction failing with error 'Input data do not match: Spectrum contains 12 orders, but line table absolute order numbering is 70- 60' while reduction 860 setting data.** This problem is due to the fact that the wavelength calibration has switched off the generation of a solution for some fibres because it has not found the minimum number of searched lines (minimum set to 1000). In this case we recommend to set the parameter minlines of flames_cal_wavecal to 600 (or less) and repeat the arc_mos and object_mos steps.

7 List of workflow tasks

This is the list of all the tasks and associated recipes in the UVES workflow. Only some of them are needed for scientific reduction, they are indicated by the flag "yes" (triggered by default) or "optional" (triggered only if requested by a workflow parameter). Other tasks are not used for scientific reduction (they are indicated by the flag "no"), they are mainly used for instrument monitoring and they can be executed only by specifying them as target. Note that, when a task is specified as target, all the tasks that generate the calibrations needed for it are automatically executed.

TASK	**RECIPE**	**Used in science reduction**	**Notes**
arc	uves_cal_wavecal	yes	Computes the wavelength calibration (UVES Slit)
arc_mos	flames_cal_wavecal	yes	Computes the wavelength calibration for UVES-Fibre
bias	uves_cal_mbias	yes	Creates a master bias
ccdtest	detmon_opt_lg	no	Processes technical calibrations for detector monitoring
cd_align	uves_cal_cd_align	no	Technical calibrations to check alignment of cross dispersers
combine_spectra	esotk_spectrum1d_combine	yes	Combines spectra from different exposures on same target
combined_flat	uves_cal_mflat_combine	optional	Combines regular and Deuterium master flat fields
dark	uves_cal_mdark	no	Computes the dark current
dflat	uves_cal_mflat	optional	Creates a master flat from Deuterium flats
efficiency	uves_cal_response	no	Processes standard stars for efficiency monitoring
flat	uves_cal_mflat	yes	Creates a master flat (UVES Slit)
flat_mos	flames_cal_mkmaster	yes	Creates a set of 3 master flats for UVES-Fibre
iflat	uves_cal_mflat	yes	Creates a master flat for the Iodine cell
object	uves_obs_scired	yes	Spectrum extraction (UVES slit)
object_mos	flames_obs_scired	yes	Spectrum extraction (UVES-Fibre)
orderpos	uves_cal_orderpos	yes	Determines the order positions (UVES Slit)
orderpos_mos	flames_cal_orderpos	yes	Determines the order positions (UVES-Fibre)
pdark	uves_cal_mdark	no	Computes the dark current from open-shutter dark exposures
plot_focus	*none*	no	Technical task for monitoring
predict	uves_cal_predict	yes	Computes first guess solutions for order and wavelength tables
predict_mos	flames_cal_predict	yes	Computes first guess solutions (UVES-Fibre)
response	uves_cal_response	yes	Computes the response curve from a flux standard star
science_slit	uves_utl_idp	yes	Creates a science product compatible with ESO Phase 3
sff_ofpos_mos	flames_cal_prep_sff_ofpos	yes	Computes the final flats for UVES-Fibre
solar	uves_obs_scired	optional	Spectrum extraction for reflected solar observations
telluric	uves_cal_response	no	Processes telluric standard stars for monitoring
tflat	uves_cal_tflat	no	Processes special lamp flats for monitoring

Table 4: UVES pipeline tasks overview

8 Frequently Asked Questions

- **Q1) Where can I find the final reduced data?**

  Answer: all the products of all the datasets and the reductions are saved into the EDPS_data directory, specified when executing the `edps-gui` for the first time. One can decide to export only the final products for selected datasets and only for the desired reduction attempts into another location for further analysis. See Section 2.4 for further instructions.

- **Q2) How do I stop the application?**

  Answer: Proceed as follows:

  1. Press "Stop EDPS" in the Dashboard.
  2. Type Ctrl-C in the terminal where the application is running. If the application doesn't terminate, type Ctrl-C again.
  3. Alternatively, kill the 'panel serve' process on your system, for example:

     ```
     ps -e | grep panel # get the process ID of the gui (<pid>).
     kill -9 <pid>
     ```

- **Q3) I have closed the browser window where the application is running. How can I reopen the application?**

  Answer: Point your browser to: `http://localhost:5006/edps-gui`

- **Q4) Where can I find some data that I can use to test the application?**

  Answer: Install the 'datademo' package provided with the pipeline installation or download the "Demo Data" package from https://www.eso.org/sci/software/pipe_aem_table.html.

  Please note that the demo data can be large (tens of Gigabytes).

  A convenient script to download demo data for any pipeline is also available and can be used from the command line:

  ```
  curl -O https://eso.org/sci/software/apptainer/eso_download_demodata.sh
  bash ./eso_download_demodata.sh
  ```

- **Q5) How can I start the edps-gui if the following message appears?**

  ```
  Cannot start Bokeh server, port 5006 is already in use
  ```

  Answer: The panel server was not closed properly. Kill it by typing:

  ```
  ps -e | grep panel # get the process ID of the gui (<pid>).
  kill -9 <pid>
  ```

- **Q6) How do I get additional support on EDPS or data reduction in general?**

  Answer: For suggestions, questions, or feedback in general, please open a ticket with the EDPS Support team. This https://support.eso.org/new-ticket?ticket%5Bticket_field_13%5D%5Bdata%5D=227 should take you directly to a webpage for creating and `EDPS` feedback ticket, but in case you want to navigate there 'manually', go to https://support.eso.org, login, click on "Submit Helpdesk Ticket", and specify the Help topic: "Post Observations", "ESO Data Processing System [EDPS]".

- **Q7) I have a lot of disk space, but when I install `EDPS` with `pip` or an ESO pipeline with `Homebrew` I get the error message: `Cannot mkdir: No space left on device`. How do I fix it?**

  Answer: This depends on how much disk space is allocated to the `/home`, `/var`, and `/tmp` directories. The final solution would be to resize the space allocated to the in the organization of the filesystem. However, we list here few tricks that might do the job.

  – Clearing the pip .cache to make space for new packages. Type the command:

    ```
    pip cache purge
    ```

    before installing `EDPS`.

  – Redirect the cache, `Homebrew` temporary build directories into a partition with enough space. Set some of the following environmental variables in your `.bashrc` file:

    ```
    export HOMEBREW_CACHE=<path_to_new_cache_directory>
    export XDG_CACHE_HOME=<path_to_new_cache_directory>
    export HOMEBREW_TEMP=<path_to_new_temporary_directory>
    export TMPDIR=<path_to_new_temporary_directory>
    ```

    The first moves only the location of `Homebrew` cache, the second the cache of most applications (instead of the default `/home/username/.cache`), the third moves the directory where `Homebrew` builds, extracts, and saves temporary files (instead of the defaults `/tmp` and `/var/tmp`). The last changes the global system temporary directory and affects most of the linux commands.

  – As extreme measure, one can move the `/home/linuxbrew/.linuxbrew` directory somewhere else, and create a symbolic link in `/home/linuxbrew`. For example:

    ```
    cd /home/linuxbrew
    mv -f .linuxbrew <path_to_new_directory>
    ln -s <path_to_new_directory> .linuxbrew
    ```

    Important note: this operation might break some internal links. Recipes requiring external packages such as `telluriccorr` might not work (impacts on KMOS, XSHOOTER, FORS2, and MOLECFIT pipelines).