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

SPHERE IRDIS EDPS-GUI tutorial

1 rev. 20260727

Issue 1

Date July 27 2026

Prepared:	M. Marsset	July 27 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	01/07/2026	all	First public release for 0.59.1

This page was intentionally left blank

Contents

1 Introduction

1.1 Scope

This document describes how to reduce SPHERE IRDIS data with the `edps-gui` (Graphic User Interface), the dashboard of the ESO Data Processing System (`EDPS`), which is the recommended interface to reduce data from ESO telescopes. Details on the SPHERE IRDIS 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 SPHERE IRDIS pipeline itself, consult the pipeline manual available at: https://www.eso.org/sci/software/pipe_aem_table.html.

Note: this tutorial refers to:

- SPHERE IRDIS instrument pipeline named `spher`, version 0.59.1.
- SPHERE IRDIS workflow: `sphere.sphere_irdis_wkf`
- `EDPS` version 1.7.1.
- `edps-gui` version 0.91.

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 consists 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. Then, the workflow can be 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 a 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 process 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 (defining 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 paths 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 SPHERE IRDIS demo data. We assume that the EDPS, edps-gui, the SPHERE IRDIS 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).

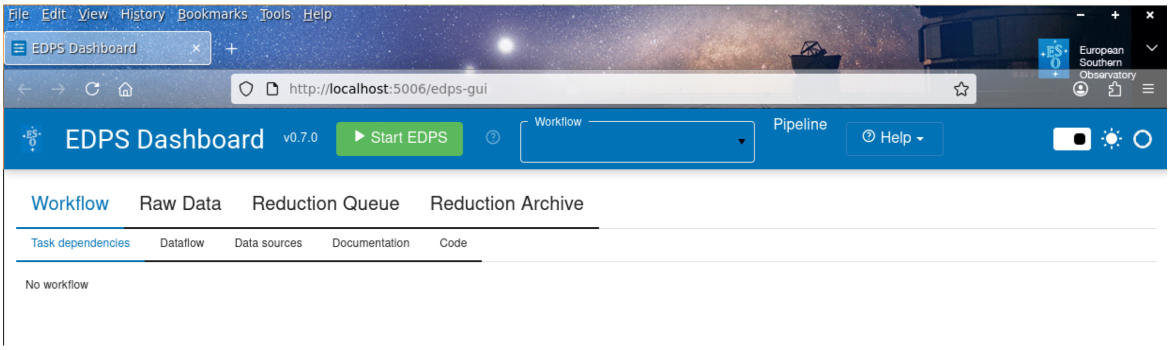


Figure 1: The empty edps-gui Dashboard; the underlying EDPS engine has not yet been started and no workflow has been loaded.

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

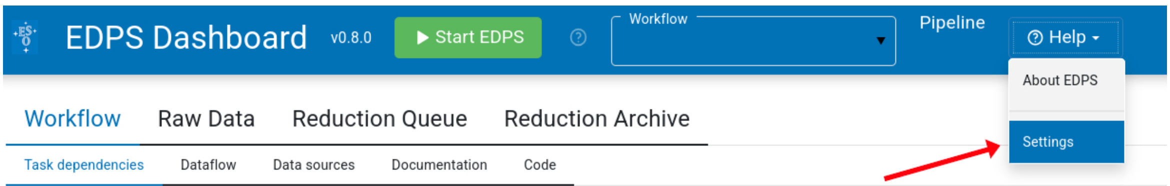


Figure 2: The “Help” –> “Settings” menu.

4. On the browser window with the dashboard, press the button ‘Start EDPS‘.
5. Choose the sphere.sphere_irdis_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.

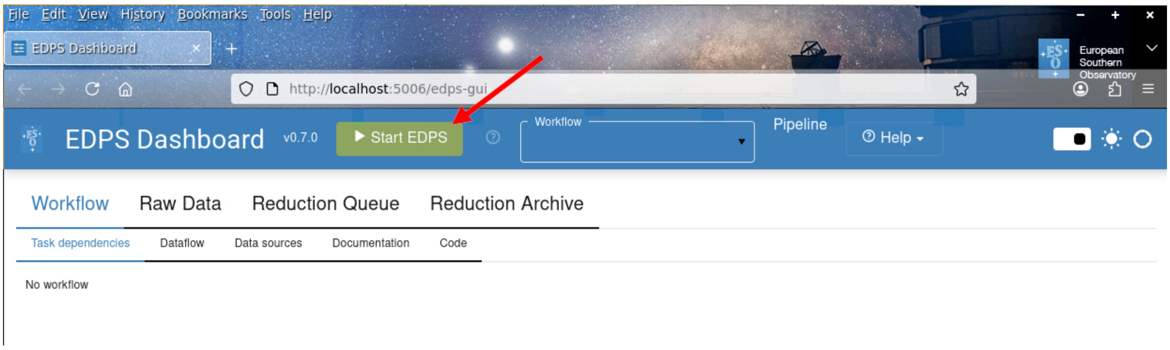


Figure 3: The "Start EDPS" button.

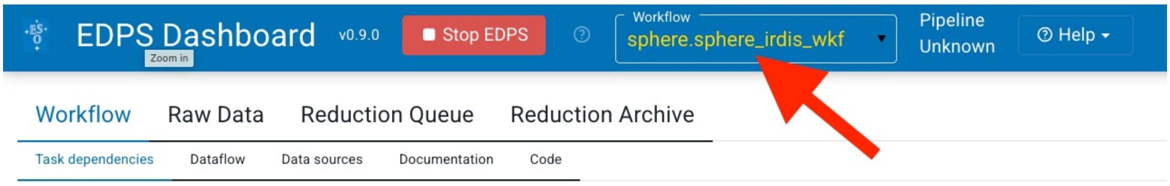


Figure 4: The `edps-gui` with the SPHERE IRDIS workflow loaded.

2.2 Selecting the input data

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

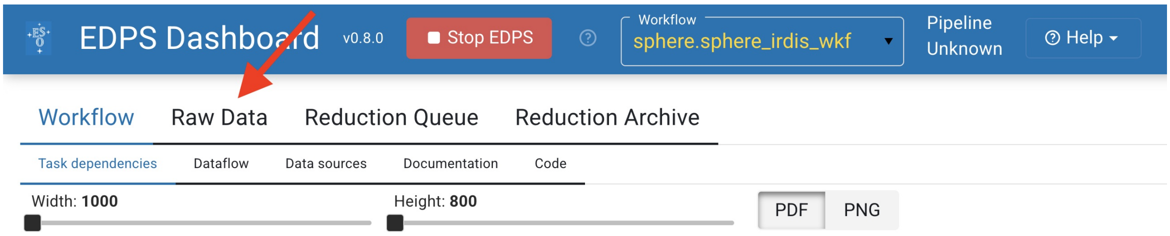


Figure 5: How to select RAW data Tab.

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 **??**.

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.

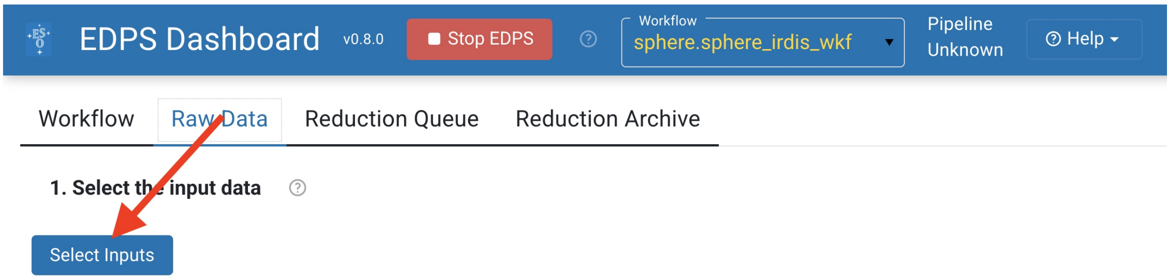


Figure 6: How to select input data.

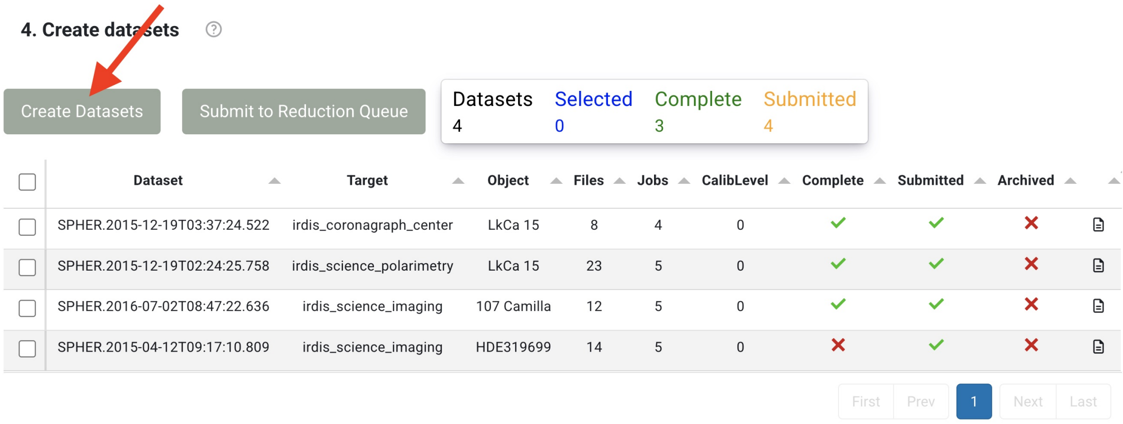


Figure 7: How to inspect the input data directory to create datasets.

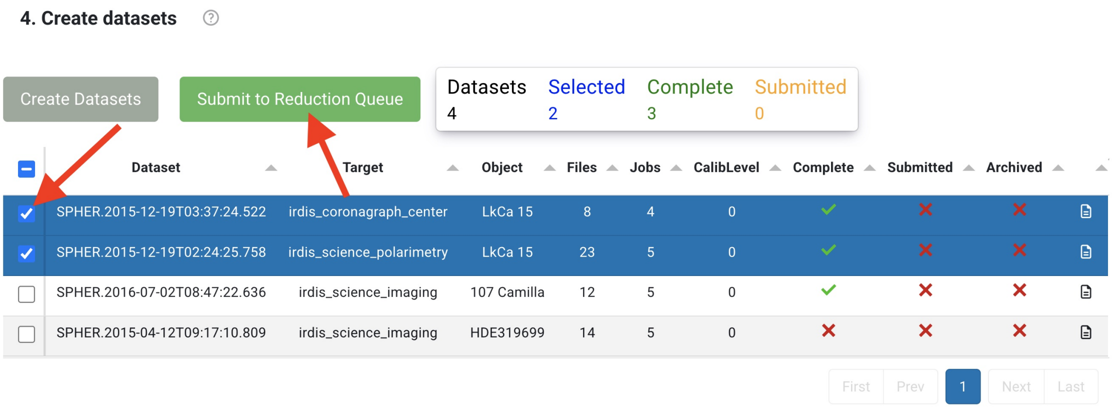


Figure 8: How to send the selected datasets to the Reduction Queue for processing.

2.3 Start the reduction

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

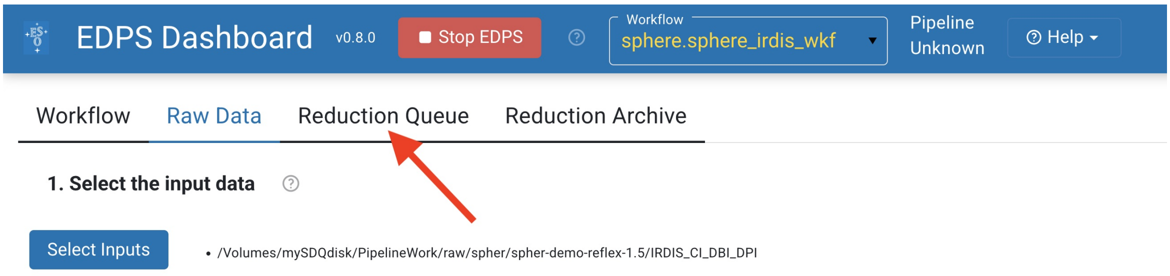
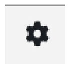


Figure 9: How to select Reduction Queue tab.

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 4.2 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.

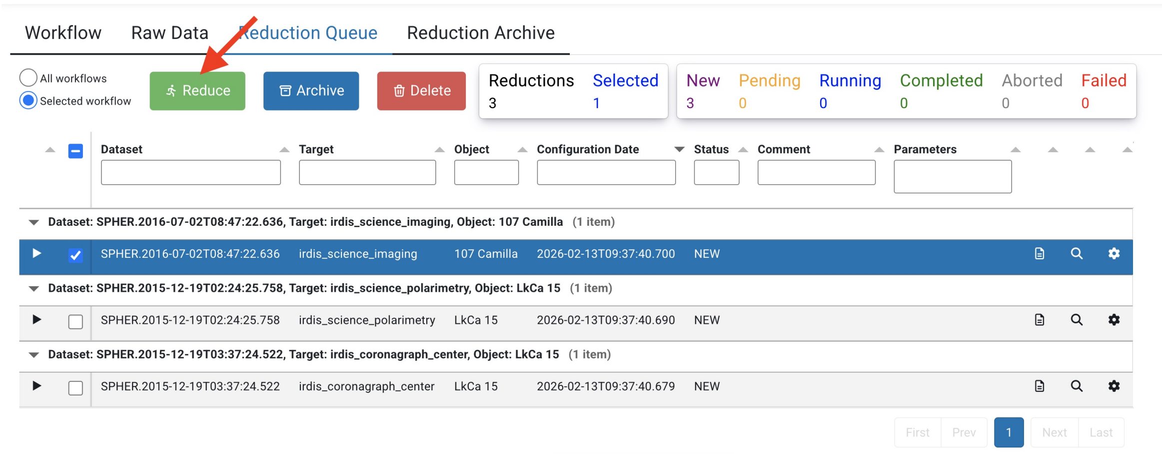


Figure 10: Reduce.

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.

2.3.1 Quality plots

It is possible to inspect the information on each job, such as quality plots showing the products (for the most important jobs and products), the list of inputs and output files, the recipe parameters and logs. All the information on the job processing can be inspected from the 'Reduction Queue' window. While quality plots are produced only for completed jobs, the other relevant information is available also for failed jobs.

The information associated for the main product can be inspected by pressing the magnifying glass symbol at the right side of each dataset. To inspect the information associated to other individual jobs (e.g., calibrations), proceed as follows:

- 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.

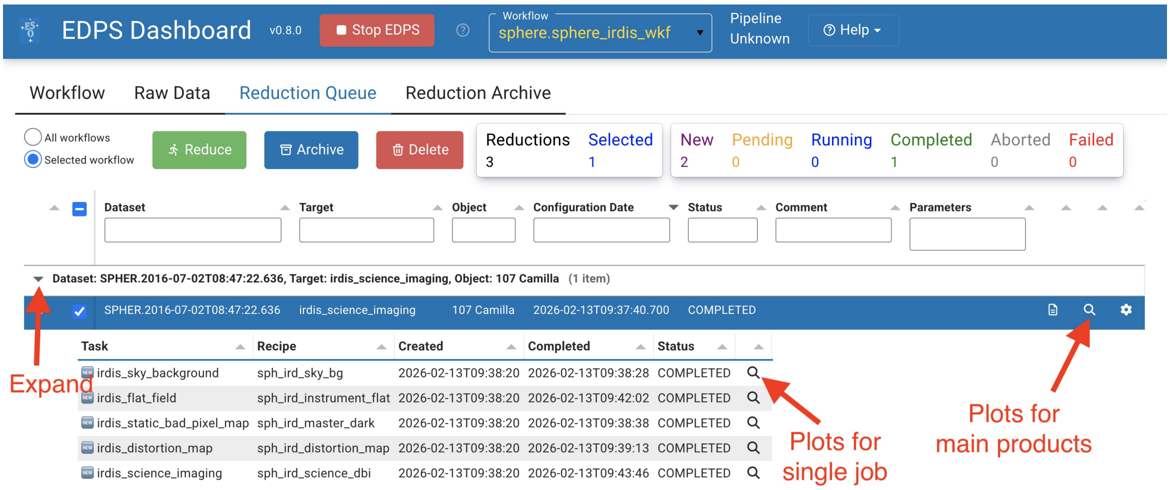


Figure 11: How to look for job information from the Reduction Queue tab.

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. Additionally, even if all products for all tasks are saved in the EDPS_data directory, the most important products can be ‘exported’ to a desired location.

To do so, proceed as follows:

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

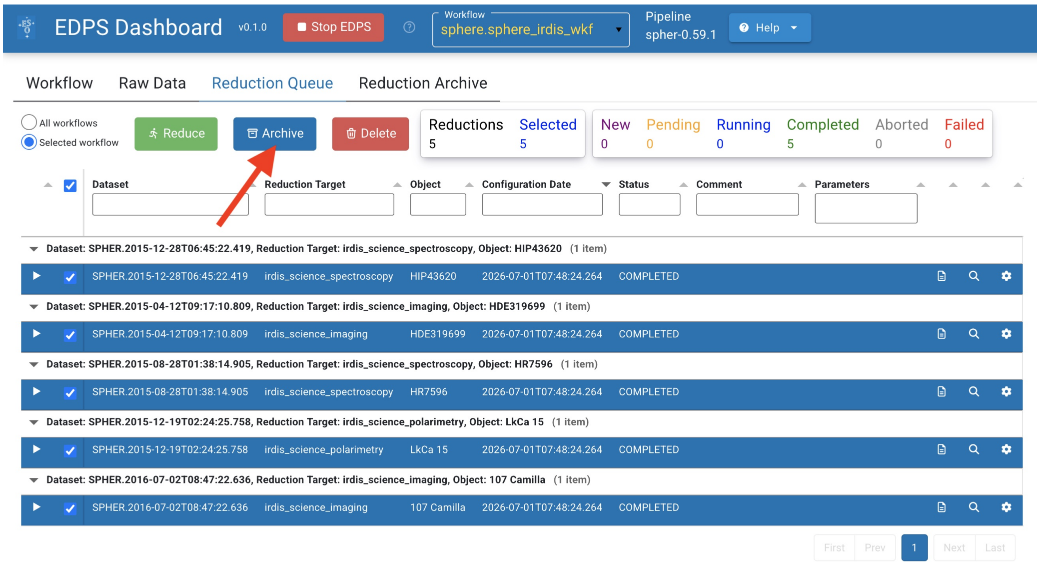


Figure 12: How to archive a completed reduction from the Reduction Queue tab.

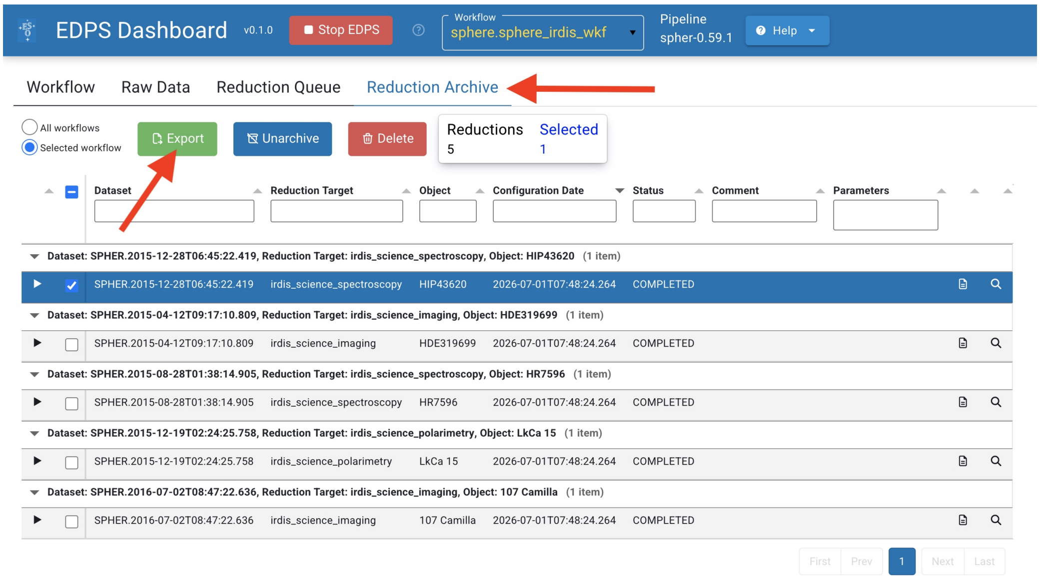


Figure 13: The reduction archive tab. This table contains all the different configurations of datasets that are declared "finished" and removed from the Reduction Queue. From this page, the user can export the most important files into a desired local directory.

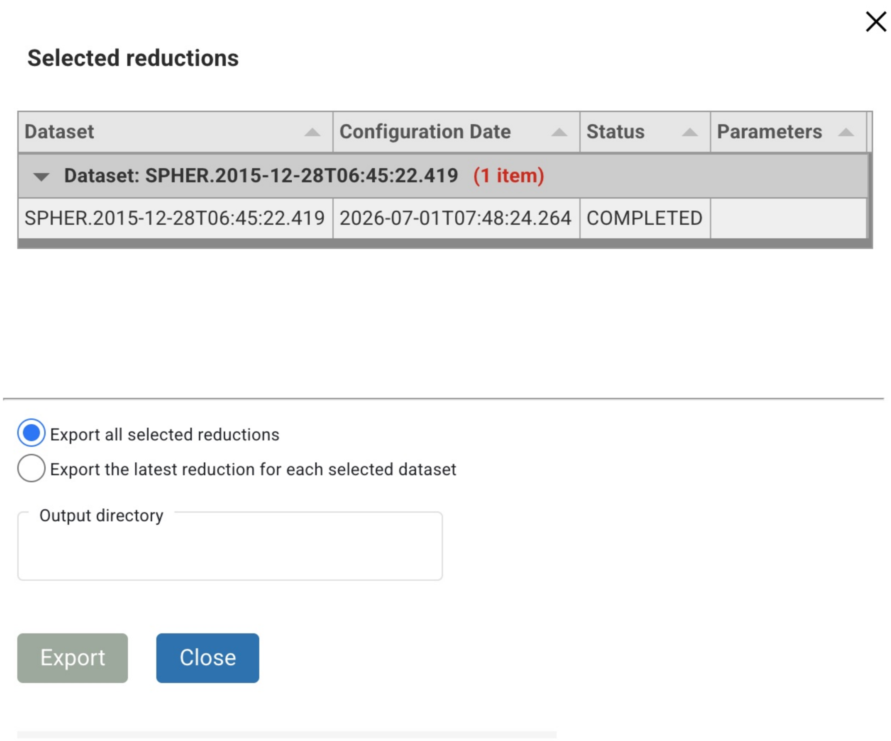


Figure 14: The EXPORT dialogue window, where the user can decide which reduced configuration to save and where.

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 are written to the specified output directory. Their filenames follow the convention:

IRDIS_<MODE>_<ARCFILE>.fits,

where <MODE> identifies the observing mode

(e.g., IMAGE, POLARIMETRY, SPECTROSCOPY_LEFT, SPECTROSCOPY_RIGHT, or STAR_CENTER) and <ARCFILE> is the archive filename stored in the FITS header keyword ARCFILE.

For imaging and spectroscopy modes, the output is a calibrated science image together with associated quality maps: bad-pixel map, number of input frames contributing to each pixel (ncomb) and RMS map.

For polarimetric observations, the product contains eight FITS extensions. The first four correspond to the reduced science intensity image and include the science image itself, a bad-pixel map, a ncomb map, and an RMS map. The remaining four extensions contain the reduced polarization image together with its associated bad-pixel map, ncomb map, and RMS map.

Finally, a binary table listing the measured stellar centre coordinates is provided for star-centre observations.

3 The SPHERE IRDIS data reduction flow.

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

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 EDPS workflow is designed to execute the tasks that deliver the final reduced data 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 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 standard 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 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 Section 4 for more information.

The reduction steps are:

3.1 Subworkflow: Dark Background

Recipes: `sph_ird_master_dark`, `sph_ird_ins_bg`, `sph_ird_sky_bg`

This subworkflow generates the dark-current, instrumental-background, and sky-background calibration products. It consists of the following tasks:

- **irdis_dark_imaging**
- **irdis_static_bad_pixel_map**
- **irdis_instrument_background**
- **irdis_sky_background**

First, **irdis_dark_imaging** uses recipe `sph_ird_master_dark` to create the master dark frame by combining raw dark exposures using the selected collapse algorithm (typically Clean Mean).

The task **irdis_static_bad_pixel_map** runs the same recipe, `sph_ird_master_dark`,

and exports the `IRD_STATIC_BADPIXELMAP` product.

Next, **irdis_instrument_background** uses `sph_ird_ins_bg` to generate the instrumental-background calibration frame by combining raw background exposures using the selected collapse algorithm. Unlike the master dark recipe, no bad-pixel map is created.

SPHERE IRDIS workflow

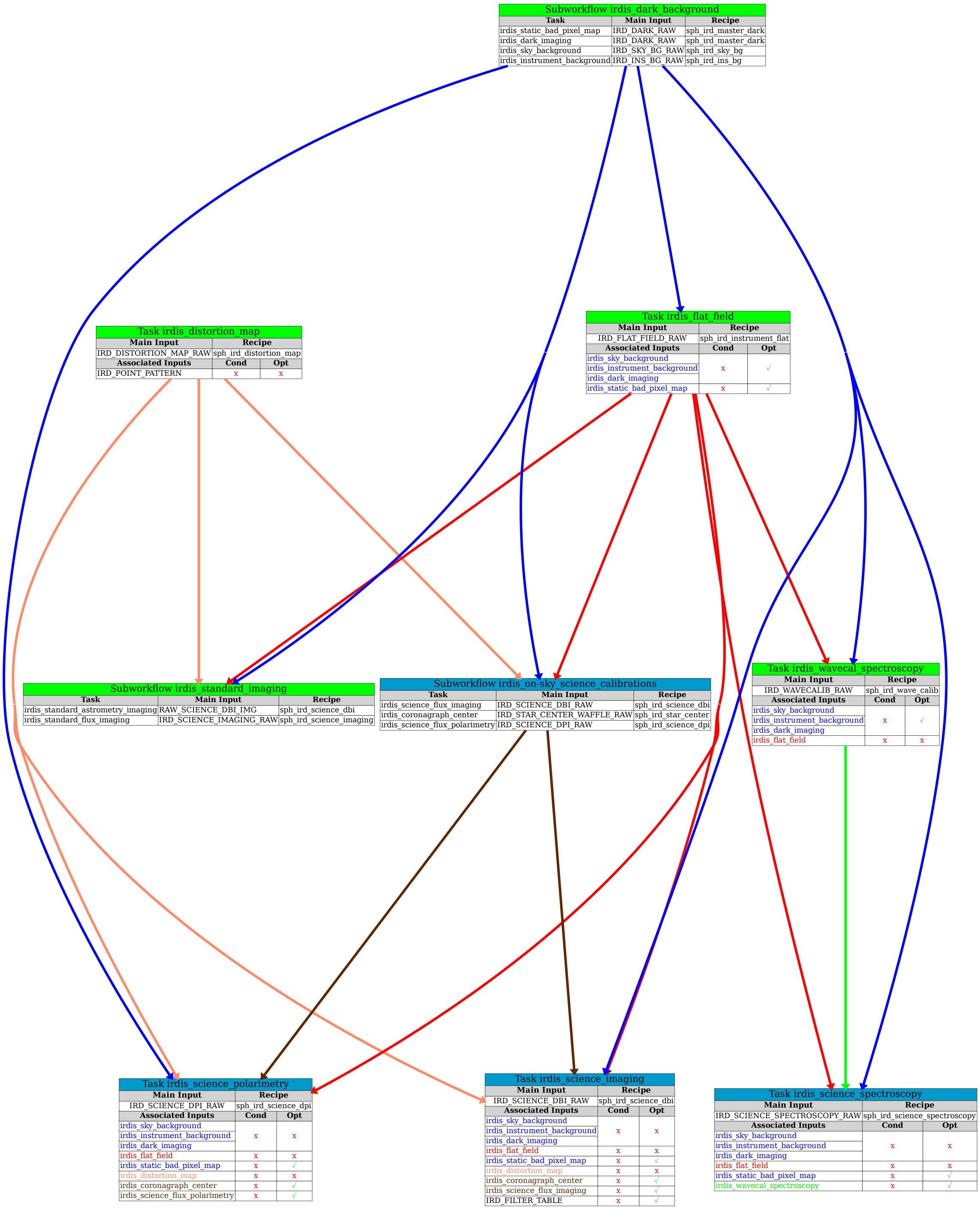


Figure 15: The data reduction cascade of the SPHERE IRDIS workflow.

Finally, **irdis_sky_background** uses `sph_ird_sky_bg` to produce the sky-background calibration frame by combining raw sky-background exposures. As for the instrumental background recipe, no bad-pixel map is generated.

The resulting `MASTER_DARK`, `INS_BG`, and `SKY_BG` products are intended for subtraction from raw science or calibration frames. They are neither flat-fielded nor corrected using other dark or background calibrations.

The output products are multi-extension FITS files containing the calibration image, a bad-pixel map, a weight map recording the number of input frames contributing to each pixel, and an RMS map.

Customization Recipe parameters:

- `ird.sky_bg.coll_alg` / `ird.master_dark.coll_alg` / `ird.ins_bg.coll_alg`: Collapse algorithm used to combine the input frames (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

3.2 Master Flat

Recipe: `sph_ird_instrument_flat`

The task **irdis_flat_field** uses the pipeline recipe to generate the IRDIS detector flat field from narrow- or broadband lamp exposures acquired at different illumination levels.

Background inputs are optional. Raw dark frames with matching detector integration times (DITs) may be provided. If no suitable raw darks are available, EDPS may use the following calibration products in decreasing order of priority `SKY_BG` → `INS_BG` → `MASTER_DARK`.

Customization Recipe parameters:

- `ird.instrument_flat.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).
- `ird.instrument_flat.robust_fits`: Enables robust linear fitting. The default value is `FALSE`. Setting this parameter to `TRUE` can improve rejection of outliers such as cosmic rays.

3.3 Distortion Map

Recipe: `sph_ird_distortion_map`

The task **irdis_distortion_map** uses the recipe to derive the IRDIS geometric distortion map from calibration observations that have been reduced as standard field-stabilized science images without dithering. The reduction may optionally include dark subtraction and flat-field correction.

After combining the input frames, point sources are detected using a user-defined detection threshold. If no reference point pattern is provided, the recipe generates one automatically. Otherwise, the detected source positions are compared with the reference pattern to measure the optical distortion.

The distortion map is written as a multi-extension FITS file containing 16 extensions:

- Extensions 1–4 contain the distortion map, bad-pixel map, RMS map, and weight map for the left field of view in the detector x direction.
- Extensions 5–8 contain the corresponding products for the left field of view in the detector y direction.
- Extensions 9–12 contain the distortion map, bad-pixel map, RMS map, and weight map for the right field of view in the detector x direction.
- Extensions 13–16 contain the corresponding products for the right field of view in the detector y direction.

Optional quality-control products include images of the detected point patterns, distortion-corrected detector and field-of-view images, and residual distortion maps. For a successful solution, the residual distortion is typically below 0.1 pixels.

The estimated optical-axis position and the polynomial coefficients describing the distortion solution are recorded in the FITS header for quality monitoring and verification.

Customization Recipe parameters:

- `ird.distortion_map.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).
- `ird.distortion_map.threshold`: Detection threshold for point sources (default: `3.0`).

3.4 Subworkflow: Standard Imaging

This subworkflow processes photometric and astrometric standard-star observations using the following tasks:

- **irdis_standard_flux_imaging**
- **irdis_standard_astrometry_imaging**

These tasks execute the pipeline recipes `sph_ird_science_imaging` (classical imaging, CI)

and `sph_ird_science_dbi` (dual-band imaging, DBI), respectively, described hereafter.

3.4.1 Standard Flux Imaging

Recipe: `sph_ird_science_imaging`

The task **irdis_standard_flux_imaging** uses the recipe to reduce photometric standard-star observations for zero-point calibration.

Raw frames are dark-subtracted, optionally flat-fielded, and a combined bad-pixel map is generated. The left and right detector regions are extracted using the IRDIS instrument model. Individual exposures are then combined using one of the available collapse algorithms.

The output product is a multi-extension FITS file containing eight extensions: the image, bad-pixel map, weight map, and RMS map for both the left and right fields of view.

Customization Recipe parameters:

- `ird.science_imaging.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

3.4.2 Standard Astrometry Imaging

Recipe: `sph_ird_science_dbi`

The task **irdis_standard_astrometry_imaging** uses the recipe to reduce astrometric standard observations.

The processing is identical to that of the photometric standard recipe. Raw frames are dark-subtracted, optionally flat-fielded, combined with an updated bad-pixel map, and split into left and right detector regions using the IRDIS instrument model. The individual exposures are combined using the selected collapse algorithm.

The final product is a multi-extension FITS file containing the image, bad-pixel map, weight map, and RMS map for both detector fields of view.

Customization Recipe parameters:

- `ird.science_dbi.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

3.5 Subworkflow: On-Sky Science Calibrations

Recipes: `sph_ird_science_dpi`, `sph_ird_science_dbi`, `sph_ird_star_center`

This subworkflow processes on-sky calibration observations acquired in IRDIS imaging (classical, dual band) and polarimetry modes.

3.5.1 Imaging Flux Standard

The task **irdis_science_flux_imaging** executes the recipe `sph_ird_science_dbi` to reduce flux-calibration frames acquired for coronagraphic observations, in which the target star is intentionally offset outside the coronagraphic mask.

These products are intended for quality control only and are not used during the science reduction, since no absolute flux calibration is applied to the final science products.

3.5.2 Polarimetry Flux Standard

The task **irdis_science_flux_polarimetry** executes the recipe `sph_ird_science_dpi` to reduce flux-calibration observations acquired for coronagraphic polarimetric observations.

As for imaging mode, these products are not propagated into the science reduction and serve only as quality-control products.

3.5.3 Coronagraph Centre

Recipe: `sph_ird_star_center`

The task **irdis_coronagraph_center** determines the stellar position behind the coronagraph by analysing waffle calibration images.

The illuminated left and right detector regions are processed independently using a source-detection algorithm with a user-defined detection threshold. For each exposure, the output table records:

- the exposure start time,
- the measured stellar centre coordinates,
- the IRDIS DMS position converted from microns to detector pixels.

The output consists of a binary FITS table containing the measured centre positions for all processed frames.

Customization Recipe parameters:

- `ird.star_center.coll_alg:` Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).
- `ird.star_center.sigma:` Sigma threshold used for source detection (default: `10.0`).

3.6 Subworkflow: Wavelength Calibration

Recipe: `sph_ird_wave_calib`

In long-slit spectroscopy (IRDIS-LSS) mode, the task **irdis_wavecal_spectroscopy** derives the wavelength calibration using the recipe `sph_ird_wave_calib`.

The input calibration frames are combined, dark-subtracted, flat-fielded, and cleaned of bad pixels. The resulting image is sliced along the dispersion direction, and calibration emission lines are searched for within a window of ±`ird.wave_calib.line_tolerance` around their expected detector positions.

The measured line positions are matched to their reference wavelengths and fitted with a polynomial. This polynomial defines the wavelength solution and provides the interpolation between calibration lines. The final solution is written to the Pixel Description Table (PDT).

The output product is a FITS file containing six image extensions corresponding to the PDT columns:

- wavelength,

- spectrum identifier,
- slit identifier,
- wavelength width (or uncertainty),
- second derivative,
- illumination fraction.

Additional extensions contain the combined calibration image, a bad-pixel map, and an RMS map.

The wavelength-calibration task produces `IRD_WAVECALIB` products. These products are associated with spectroscopy science observations for calibration-selection purposes only and are filtered out from the recipe input of `sph_ird_science_spectroscopy`.

Customization Recipe parameters:

- `ird.wave_calib.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

3.7 Science Imaging

Recipe: `sph_ird_science_dbi`

The task **irdis_science_imaging** reduces IRDIS imaging science observations, including classical, dual-band, coronagraphic, and SAM modes, using the recipe `sph_ird_science_dbi`. The recipe supports observations obtained with or without dithering and optionally performs Angular Differential Imaging (ADI) and Spectral Differential Imaging (SDI) processing.

Raw frames are dark-subtracted, optionally flat-fielded, and combined with the static bad-pixel information derived from the calibration products. The left and right detector regions are extracted using the IRDIS instrument model before the individual exposures are combined using the selected collapse algorithm.

A dark or background calibration frame is required. If several suitable products are available, they are selected according to the following priority: `SKY_BG` → `INS_BG` → `MASTER_DARK`.

The output is a multi-extension FITS file containing eight extensions, namely, for each detector image (right and left): the science image, the bad-pixel map, the weight map (ncomb), and the RMS map.

When the required calibration tables are available, the recipe can also compute quality-control parameters such as the Strehl ratio.

Customization Recipe parameters:

- `ird.science_dbi.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

- `ird.science_dbi.use_adi`: Enables Angular Differential Imaging processing (`0` = disabled; default, `1` = enabled).
- `ird.science_dbi.use_sdi`: Enables Spectral Differential Imaging processing (`0` = disabled; default, `1` = enabled).

3.8 Science Polarimetry

Recipe: `sph_ird_science_dpi`

The task **irdis_science_polarimetry** reduces science observations acquired in IRDIS Dual-Polarization Imaging (DPI) mode using the pipeline recipe `sph_ird_science_dpi`.

The reduction procedure is essentially identical to that of the DBI science recipe. Raw frames are dark-subtracted, optionally flat-fielded, corrected for bad pixels, and combined using the selected collapse algorithm. The principal difference lies in the format of the final products, which are provided as polarization and intensity images rather than left and right detector images.

The standard output is a multi-extension FITS file containing eight extensions:

- Intensity image:
  - science image,
  - bad-pixel map,
  - weight map (ncomb),
  - RMS map.
- Polarization image:
  - science image,
  - bad-pixel map,
  - weight map (ncomb),
  - RMS map;

When ADI processing is enabled, only the polarization products are written, resulting in a FITS file with four extensions.

Customization Recipe parameters:

- `ird.science_dpi.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).
- `ird.science_dpi.use_adi`: Enables Angular Differential Imaging processing (`0` = disabled; default, `1` = enabled).
- `ird.science_dpi.use_sdi`: Enables Spectral Differential Imaging processing (`0` = disabled; default, `1` = enabled).

3.9 Science Spectroscopy

Recipe: `sph_ird_science_spectroscopy`

The task **irdis_science_spectroscopy** reduces science observations obtained in IRDIS Long-Slit Spectroscopy (LSS) mode using the pipeline recipe `sph_ird_science_spectroscopy`.

The reduction consists of dark subtraction, optional flat-field correction, bad pixel handling, and combination of the individual exposures using the selected collapse algorithm. EDPS subtracts an associated background calibration (sky background preferred, otherwise instrumental background, otherwise master dark).

The output product is a multi-extension FITS file containing the reduced two-dimensional spectrum for one detector field of view (left or right, depending on the selected slit). The FITS file contains four extensions: the science image, the bad-pixel map, the weight map (ncomb), and the RMS map.

For observations in which both left and right spectra are recorded, separate products are generated for each detector field of view.

Customization Recipe parameters:

- `ird.science_spectroscopy.coll_alg`: Collapse algorithm (`0` = Mean, `1` = Median, `2` = Clean Mean; default).

4 Overview of all the data reduction configuration options

4.1 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.

4.2 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 of 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.

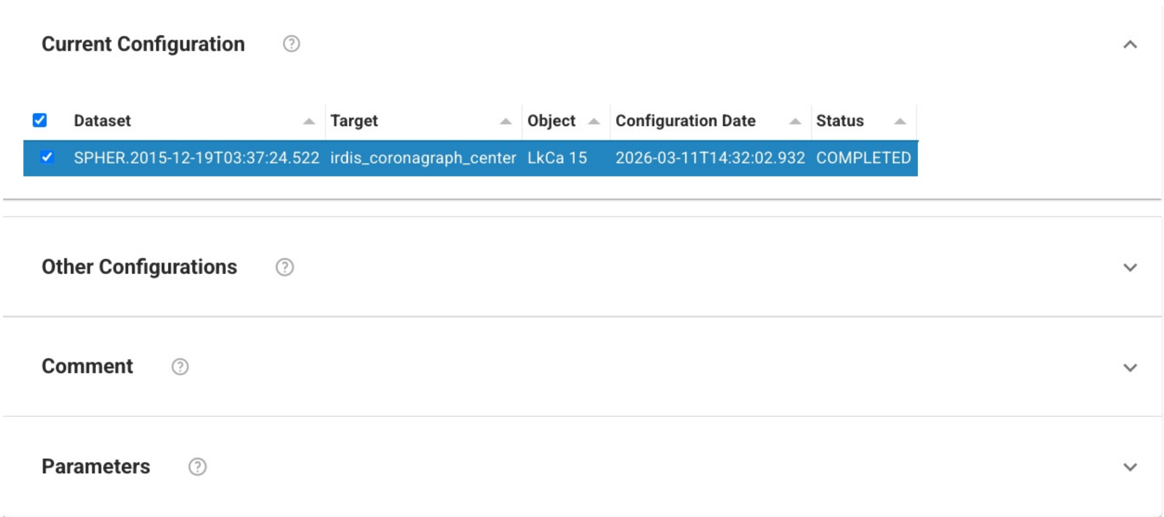


Figure 16: The Reduction Configuration editor. It contains 4 sections, that indicate the current configuration, list of other configurations to set, comments to insert, and the parameters to modify.

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.

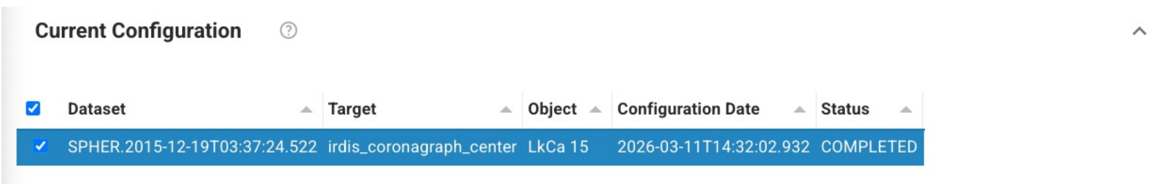


Figure 17: The first part of the Reduction Configuration Editor, that indicates the selected configuration.

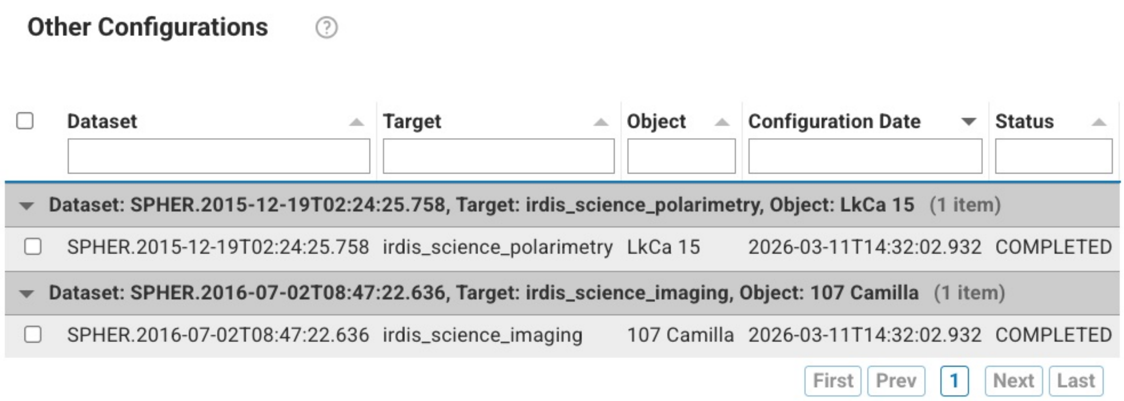


Figure 18: The second part of the Reduction Configuration Editor, that indicates other configurations for which we'd like to apply the changes.

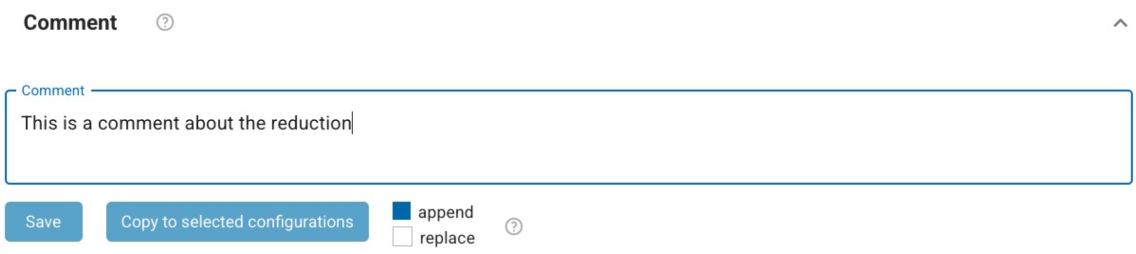


Figure 19: The third part of the Reduction Configuration Editor, that allows to specify a comment to the selected configurations.

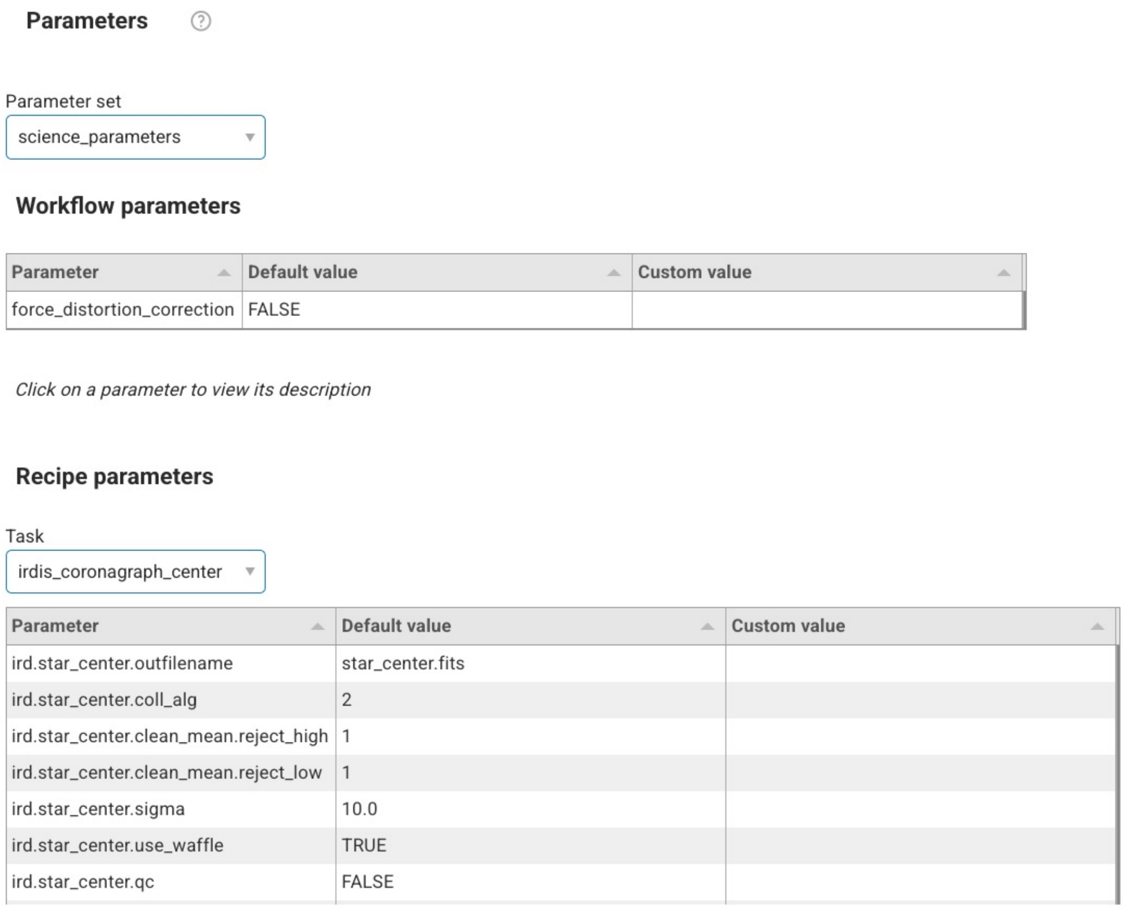


Figure 20: The fourth part of the Reduction Configuration Editor, that allows to specify the parameters sets and the recipe parameter per task. These settings can be applied to the "Selected Configuration" (Fig. 17) or to the "Other Configurations (Fig. 18).

4.3 Troubleshooting

This section provides guidance on configuring the reduction, as well as diagnosing and resolving common issues encountered during the SPHERE IRDIS data-reduction cascade.

4.3.1 Dark regions in the background

Figure 21 shows an example of dark regions appearing in the products of the imaging workflow. These features are caused by astrophysical sources present in the sky-background frames used for background subtraction.

If sky-background frames are not provided to the pipeline, master darks are used instead for the background correction, which removes this pattern.

4.3.2 Fine tuning the instrument flat

The recipe `sph_ird_instrument_flat` is executed using modified bad-pixel rejection thresholds. Instead of the default values of 0.1 and 10 for `ird.instrument_flat.badpix_lowtolerance` and `ird.instrument_flat.badpix_uptolerance`, respectively, values of 0.75 and 1.25 are adopted.

For K-band data, the High Contrast Data Center (HC-DC) recommends increasing

`ird.instrument_flat.badpix_uptolerance` to 1.5.

4.3.3 Distortion maps are not used

The pipeline recipe `sph_ird_distortion_map` determines the geometric distortion of IRDIS data using internal calibration observations obtained with a pinhole mask. The recipe can also measure the relative offset between the LEFT and RIGHT detector images when the workflow parameter `force_distortion_correction` is enabled.

In practice, distortion solutions derived from internal calibration data have been found to be insufficiently robust for routine processing. Calibration sequences acquired close in time may produce noticeably different distortion solutions, including variations in the derived geometric mapping. When applied to science or astrometric observations, these solutions generally do not improve the calibration beyond what is achieved using a simple anamorphism correction alone, and in some cases may even degrade the data.

For this reason, processing of internal distortion calibrations is disabled. A more robust approach is to correct only the dominant distortion component, which is a stable anamorphism of approximately 0.6%. This can be applied in a simple and reproducible manner by rescaling the detector coordinates, multiplying the X and Y axes by factors of 1.0059 and 1.0011, respectively.

4.3.4 Waffle spots cannot be detected

In the task **irdis_coronagraph_center**, the recipe `sph_ird_star_center` may require parameter adjustments depending on the observing setup.

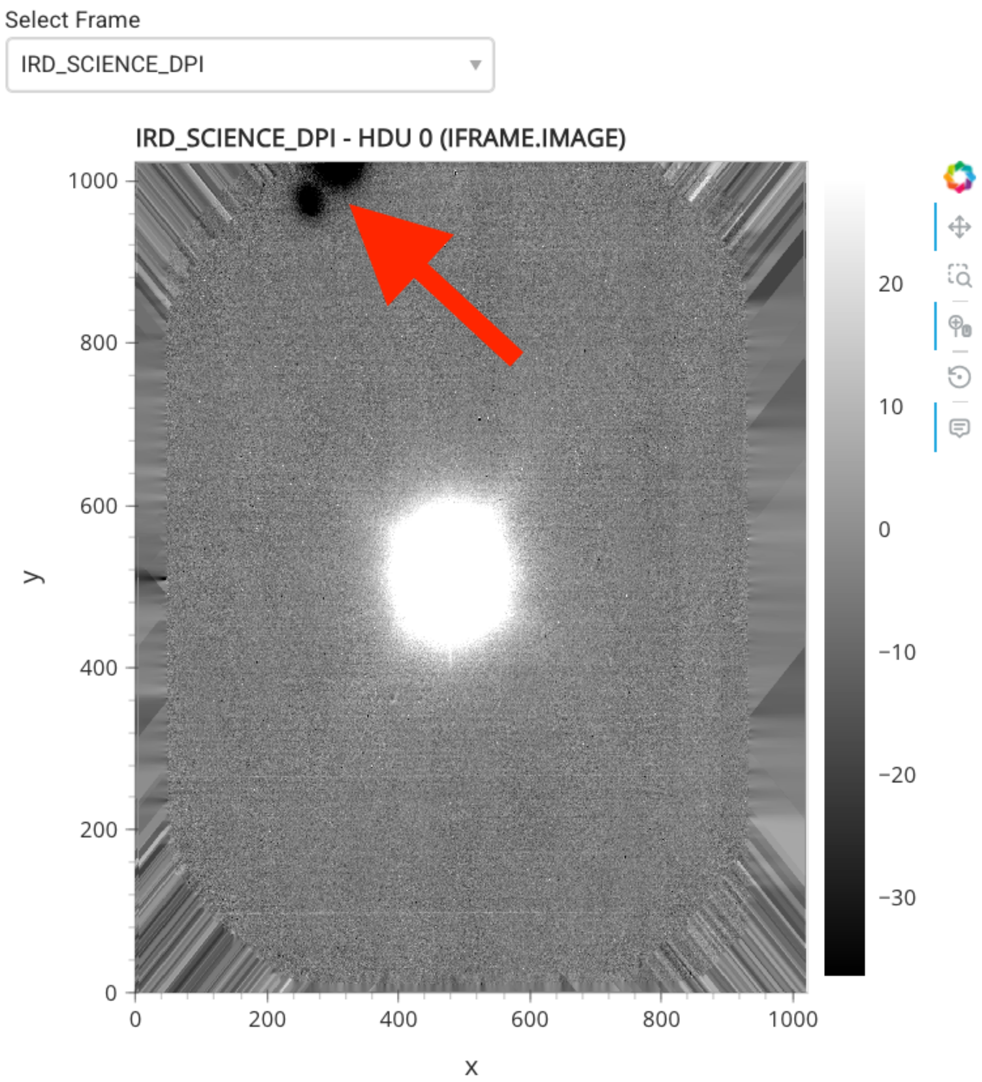


Figure 21: Dark regions in the final image product for `SPHER.2015-12-19T02:24:25.758_tpl` (DPI).

For K-band data, detection of the waffle spots can be challenging owing to the high thermal background. Increasing `ird.star_center.sigma` to 100 may improve the detection. Alternatively, the background can be removed manually before running the recipe.

The default parameters are optimized for detecting waffle spots in coronagraphic observations. For data acquired without a coronagraph, the following parameters should be modified:

- `ird.star_center.use_waffle = FALSE`;
- `ird.star_center.sigma = 500` (the stellar PSF is typically very bright in this configuration).

4.3.5 Angular Differential Imaging and Spectral Differential Imaging

The recipe `sph_ird_science_dbi` runs with both Angular Differential Imaging (ADI) and Spectral Differential Imaging (SDI) disabled by default.

If SDI processing is enabled, the Data Centre (DC) recommends setting

`ird.science_dbi.minr = 20` and `ird.science_dbi.maxr = 70`,

instead of the default values of 4 and 40.

These values are calibrated for a wavelength of $\lambda = 1.6\,\mu$m and should be scaled linearly with wavelength for other observing bands.

4.3.6 Getting additional data products

The recipe `sph_ird_science_dbi` is used to process both Classical Imaging (CI) and Dual-Band Imaging (DBI) data. The workflow is configured to save stacked products for each detector arm

(`ird.science_dbi.save_addprod = TRUE`), while intermediate products are not retained by default.

To generate and preserve all intermediate products that may be useful for further analysis, set the recipe parameter

`ird.science_dbi.save_interprod = TRUE`.

5 List of workflow tasks

This table lists all the tasks and associated recipes in the SPHERE IRDIS workflow. Only some tasks are involved in the scientific reduction. These are marked as “yes” (executed by default) or “optional” (executed only when required by the workflow). Tasks marked as “no” are not part of the default science reduction; they are intended primarily for instrument monitoring and can be executed only when explicitly selected as the reduction target. When a task is selected as the target, all prerequisite tasks required to generate its input calibrations are executed automatically.

TASK	RECIPE	Used in science reduction	Notes
irdis_coronagraph_center	sph_ird_star_center	yes (coro.)	Determines the stellar position behind the coronagraph.
irdis_dark_imaging	sph_ird_master_dark	yes	Creates the master dark.
irdis_distortion_map	sph_ird_distortion_map	no	Derives the geometric distortion solution.
irdis_flat_field	sph_ird_instrument_flat	yes	Generates the detector master flat.
irdis_instrument_background	sph_ird_ins_bg	yes	Produces the instrumental background calibration.
irdis_science_flux_imaging	sph_ird_science_dbi	optional	Reduces off-axis flux observations used for coronagraphic imaging.
irdis_science_flux_polarimetry	sph_ird_science_dpi	optional	Reduces off-axis flux observations for polarimetric mode.
irdis_science_imaging	sph_ird_science_dbi	yes (imaging)	Science reduction for classical, dual-band, coronagraphic, and SAM imaging observations.
irdis_science_polarimetry	sph_ird_science_dpi	yes (DPI)	Science reduction for dual-polarization imaging observations.
irdis_science_spectroscopy	sph_ird_science_spectroscopy	yes (LSS)	Reduces long-slit spectroscopy observations.
irdis_sky_background	sph_ird_sky_bg	optional	Creates the sky-background calibration.
irdis_standard_astrometry_imaging	sph_ird_science_dbi	optional	Processes astrometric standard observations.
irdis_standard_flux_imaging	sph_ird_science_imaging	optional	Processes photometric standard observations used for zero-point calibration.
irdis_static_bad_pixel_map	sph_ird_master_dark	yes	Produces the static bad-pixel map.
irdis_wavecal_spectroscopy	sph_ird_wave_calib	optional	Computes the wavelength calibration for LSS observations.

Table 1: Tasks and associated recipes of the SPHERE IRDIS workflow.

Notes

- For science tasks, background calibrations are associated in the following priority order: `SKY_BG` → `INS_BG` → `MASTER_DARK`.
- For polarimetric science reductions, coronagraph-centre calibrations are associated only if they were obtained before the corresponding science observation.
- For science and IDP reductions, `force_distortion_correction` is set to `FALSE` by default (see Section 4.3.3). The distortion calibration may still be associated by EDPS, but it is not used by the scientific recipe. For QC0 processing, this parameter is set to `TRUE`.

6 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).