

European Organisation for Astronomical Research in the Southern Hemisphere

Programme: GEN

Project/WP: Science Operation Software Department

ESOTK Pipeline User Manual

Document Number: ESO-XXXXXX

Document Version: 0.9.7

Document Type: Manual (MAN)

Released on: 2024-05-15

Document Classification: Public

Prepared by: ESOTK Pipeline Team

Validated by:

Approved by:

Name

Authors

Name	Affiliation
ESOTK Pipeline Team	ESO

Change Record from previous Version

Affected Section(s)	Changes/Reason/Remarks
None	The output product header of the recipe esotk_eop has been changed to be ESO conform, i.e. ESO.PRO.TYPE header keyword has been changed to the value `STATIC`.
None	A bug in the recipe esotk_spectrum1d_combine has been fixed for the use-case that the destination wavelength is provided by the user and not taken from the first input spectra. The ESO.PRO.CATG was not properly recognized by the recipe.
None	Update of the build system and compiler flags.

Contents

1 Introduction **7**

1.1 Scope . . . 7

1.2 Stylistic conventions . . . 7

1.3 Notational Conventions . . . 7

2 Related Documents **8**

2.1 Reference Documents . . . 8

3 Definitions, Acronyms and Abbreviations **9**

4 Overview **10**

5 Quick Start **11**

5.1 The ESOTK Pipeline Recipes . . . 11

5.2 Running the ESOTK Pipeline Recipes . . . 11

5.2.1 Getting Started with EsoRex . . . 12

5.3 Data Organization . . . 13

6 Known Issues **15**

7 Recipe Reference **16**

7.1 esotk_mask_create . . . 16

7.1.1 Short algorithm description . . . 16

7.1.2 Input Frames . . . 16

7.1.3 Product Frames . . . 16

7.1.4 Recipe Parameters . . . 17

7.2 esotk_masterfringe_create . . . 17

7.2.1 Short algorithm description . . . 17

7.2.2 Input Frames . . . 18

7.2.3 Product Frames . . . 18

7.2.4 Recipe Parameters . . . 19

7.3 esotk_masterfringe_correct . . . 19

7.3.1 Short algorithm description . . . 20
7.3.2 Input Frames . . . 20
7.3.3 Product Frames . . . 20
7.3.4 Recipe Parameters . . . 21
7.4 esotk_spectrum1d_combine . . . 21
7.4.1 Short algorithm description . . . 21
7.4.2 Input Frames . . . 22
7.4.3 Product Frames . . . 22
7.4.4 Recipe Parameters . . . 23
7.4.5 QC Parameter . . . 26
7.5 esotk_barycorr . . . 27
7.5.1 Short algorithm description . . . 27
7.5.2 Input Frames . . . 28
7.5.3 Product Frames . . . 28
7.5.4 Recipe Parameters . . . 29
7.6 esotk_eop . . . 29
7.6.1 Short algorithm description . . . 29
7.6.2 Input Frames . . . 30
7.6.3 Product Frames . . . 30
7.6.4 Recipe Parameters . . . 30

A Statistical Estimators **31**
A.1 Arithmetic mean - Arithmetic weighted mean . . . 31
A.2 Median . . . 31
A.3 Kappa-sigma clipping . . . 33

B Installation **34**

C Troubleshooting **35**

1 Introduction

1.1 Scope

This document is a guide to the instrument independent recipes as provided by the esotk (ESO Tool Kit) pipeline. It should guide the users through the installation of the software, and bring them quickly to the point where they can use the software tools offered by esotk

The examples on running individual pipeline recipes in this manual use the esorex command and manually created list of input files. Several interfaces to automatically organise the data, create the list of input files and execute the pipeline recipes in the proper sequence are available, see the ESO pipeline page for details.

Please note that the use of Gasgano as a GUI for processing data is deprecated. Its use is no longer recommended and the related section in this manual, as well as support for Gasgano as a data processing GUI application in general will be dropped entirely in a future release.

1.2 Stylistic conventions

Throughout this document the following stylistic conventions are used:

bold	in text sections for commands and other user input which has to be typed as shown
italics	in the text and example sections for parts of the user input which have to be replaced with real contents
`teletype`	in the text for FITS keywords, program names, file paths, and terminal output, and as the general style for examples, commands, code, etc

In example sections expected user input is indicated by a leading shell prompt.

In the text **bold** and *italics* may also be used to highlight words.

1.3 Notational Conventions

Hierarchical FITS keyword names, appearing in the document, are given using the dot–notation to improve readability. This means, that the prefix “HIERARCH ESO” is left out, and the spaces separating the keyword name constituents in the actual FITS header are replaced by a single dot.

2 Related Documents

2.1 Reference Documents

[RD01]	GEN-SPE-ESO-33000-5335 / ESO-044286	8.0	Phase 3 User Documentation - ESO Science Data Products Standard

3 Definitions, Acronyms and Abbreviations

CPL	Common Pipeline Library
CCD	Charge Coupled Device
DFS	Data Flow System
DRS	Data Reduction System
ESO	European Southern Observatory
EsoRex	ESO Recipe Execution Tool
ESOTK	ESO Tool Kit
FITS	Flexible Image Transport System
FOV	Field Of View
GUI	Graphical User Interface
OB	Observation Block
pixel	picture element (of a raster image)
PSF	Point Spread Function
QC	Quality Control
SDP	Science Data Product
SOF	Set Of Frames
TBD	To be defined
TBC	To be confirmed
VLT	Very Large Telescope
WCS	World Coordinate System

4 Overview

The ESO Tool Kit is a collection of in general instrument independent recipes that can be used to apply specific algorithms like fringe correction or stacking of spectra.

5 Quick Start

5.1 The ESOTK Pipeline Recipes

The ESOTK at the moment provides the following recipes:

- Recipes to remove fringes from an image
  - esotk_mask_create: Determines the object mask
  - esotk_masterfringe_create: Determines the masterfringe frame
  - esotk_masterfringe_correct: Scales the masterfringe frame and removes the fringes from the image.
- Recipe to coadd spectra in IDP and non-IDP format
  - esotk_spectrum1d_combine: Combines a list of input 1D spectra into a single spectrum.
- Recipes to derive and apply the barycentric correction of an observation.
  - esotk_barycorr: Derives and applies the barrycentric correction of an observation
  - esotk_eop: Retrieves the IERS Earth Orientation Parameter used by the esotk_barycorr recipe.

The recipes for fringe correction can be run within a reflex workflow, whereas the other recipes are standalone recipes without a workflow.

5.2 Running the ESOTK Pipeline Recipes

Before being able to call pipeline recipes to process a set of data, the data must be correctly classified, and associated with the appropriate calibrations. The *Data Classification* consists of tasks such as: "What kind of data am I?", *e.g.*, BIAS, "to which group do I belong?", *e.g.*, to a particular Observation Block or observing mode. *Data Association* is the process of selecting appropriate calibration data for the reduction of a set of frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. Since all the required information is stored in the FITS headers, data association is based on a set of header keywords (called "association keywords") and the process is specific to each type of calibration. The process of data classification and association is known as *data organisation*.

The ESOTK pipeline consists of a set of data processing modules that can be called from different host applications, namely:

- *Reflex* is a graphical tool that helps the user to execute data reduction workflows that contain several recipes. *Reflex* takes care of grouping the different data sets, associating the calibration frames and managing the interdependencies between recipes in the calibration cascade. **Reflex is the recommended software tool for reducing your data.**
- *EsoRex* is a command line tool used to run each pipeline recipe. *EsoRex* commands can be easily scripted.

5.2.1 Getting Started with EsoRex

EsoRex is a command-line tool which can be used to execute the recipes of all standard VLT/VLTI instrument pipelines. With *EsoRex* in your path, the general structure of an *EsoRex* command line is

```
1> esorex [esorex options] [recipe [recipe options] [sof [sof]...]]
```

where options appearing before the recipe name are options for *EsoRex* itself, and options given after the recipe name are options which affect the recipe.

All available *EsoRex* options can be listed with the command

```
1> esorex --help
```

and the full list of available parameters of a specific recipe can be obtained with the command

```
1> esorex --help <recipe name>
```

The output of this command shows as parameter values the current setting, i.e. all modifications from a configuration file or the command line are already applied.

The listing of all recipes known to *EsoRex* can be obtained with the command

```
1> esorex --recipes
```

The last arguments of an *EsoRex* command are the so-called *set-of-frames*. A *set-of-frames* is a simple text file which contains a list of input data files for the recipe. Each input file is followed by an unique identifier (frame classification or frame tag), indicating the contents of this file. The input files have to be given as an absolute path, however *EsoRex* allows the use of environment variables so that a common directory prefix can be abreviated. Individual lines may be commented out by putting the hash character (#) in the first column. An example of a *set-of-frames* is shown in the following:

```
1> cat bias.sof
/data/iiinstrument/raw/IIINSTRUMENT.2019-03-29T09:48:53.153.fits BIAS
$RAW_DATA/IIINSTRUMENT.2019-03-29T09:50:36.645.fits BIAS
$RAW_DATA/IIINSTRUMENT.2019-03-29T09:52:16.513.fits BIAS
$RAW_DATA/IIINSTRUMENT.2019-03-29T09:53:47.996.fits BIAS
#$RAW_DATA/IIINSTRUMENT.2019-03-29T09:55:04.515.fits BIAS
```

These *set-of-frames* files will have to be created by the user using a text editor, for instance. Which classification has to be used with which recipe is described in section 5.3

Finally, if more than one *set-of-frames* is given on the command-line *EsoRex* concatenates them into a single *set-of-frames*.

5.3 Data Organization

ESOTK includes instrument independent recipes. Therefore the classification tags of the frames are kept general. The tags accepted by the recipes as input and output tags are:

- Recipe esotk_mask_create

```
Input files:

  DO category:                Explanation:           Required:
  --------------------------  -----------            ---------
  RAW                         Data                   YES
  RAW_BPM                     Bad pixel mask         NO
  RAW_CONFMAP                 Confidence mask        NO

Output files:

  DO category:                Explanation:
  --------------------------  -----------
  OBJ_MASK                    Object mask
```

- Recipe esotk_masterfringe_create

```
Input files:

  DO category:                Explanation:           Required:
  --------------------------  -----------            ---------
  RAW                         Data                   YES
  RAW_BPM                     Bad pixel masks        NO
  RAW_ERROR                   Error                  NO
  RAW_VARIANCE                Variance image         NO
  OBJ_MASK                    Object masks           NO
  FRINGE_MASK                 Fringe mask            NO

Output files:

  DO category:                Explanation:
  --------------------------  -----------
  MASTER_FRINGE               Master-fringe
  MASTER_FRINGE_ERROR         Error of master-fringe
  MASTER_FRINGE_CONTRIBUTION  Contribution map of master-fringe
  MASTER_FRINGE_BPM           BPM of master-fringe
```

- esotk_masterfringe_correct

Input files:

DO category:	Explanation:	Required:
RAW	Data	YES
RAW_BPM	Bad pixel masks	NO
RAW_ERROR	Error	NO
RAW_VARIANCE	Variance	NO
OBJ_MASK	Object masks	NO
FRINGE_MASK	Fringe mask	NO
MASTER_FRINGE	Master-fringe	YES
MASTER_FRINGE_BPM	BPM of master-fringe	NO
MASTER_FRINGE_ERROR	Error of master-fringe	NO

Output files:

DO category:	Explanation:
FRINGE_CORRECTED	Fringe corrected images
FRINGE_CORRECTED_BPM	Bad Pixel Mask of the fringe corrected images
FRINGE_CORRECTED_ERROR	Error of the fringe corrected images

- esotk_spectrum1d_combine

Input files:

DO category:	Explanation:	Required:
SPECTRUM_1D	Spectrum in IDP format	YES
SPECTRUM_COLLAPSE_WLENGTHS	Destination wavelengths	NO

Output files:

DO category:	Explanation:
ESOTK_SPECTRUM_COMBINED	The combined spectrum
ESOTK_SPECTRUM_IDP_FORMAT	The combined spectrum in IDP format

6 Known Issues

7 Recipe Reference

The ESOTK pipeline process data that are usually products of other data reduction pipelines. These are identified by the value of a DO category the user must provide or adapt before using an ESOTK recipe. In this section we summarize all relevant information to use the various recipes.

7.1 esotk_mask_create

This recipe detects the objects in the input images (`RAW`) and creates an objects mask (`OBJ_MASK`). The latter is then used in the esotk_masterfringe_create recipe to exclude the objects when deriving the masterfringe frame.

7.1.1 Short algorithm description

The recipe first estimates the local sky background over the field and tracks any variations at adequate resolution to eventually remove them (if bkg.estimate=TRUE). Then it detects objects/blends of objects and keep a list of pixels belonging to each blend. Finally, it flags all pixels belonging to the detected objects. Please note that pixels are weighted in the source detection according to their value in the confidence map.

7.1.2 Input Frames

Frame Tag	Type	Count	Description
RAW	raw	1..n	Data image
RAW_BPM	bpm	1..n	Bad pixel mask
RAW_CONFMAP	confmap	1..n	Confidence mask

The recipe assumes that all data input frames are either single extension frames with the data in the primary section only, or multi extension frames. If they are multi extension frames the primary extension is supposed to be empty and is not read. Moreover, if optional bad pixel, and/or confidence masks are provided, the number (and dimension) of additionally provided confidential maps (and bad pixel masks) must match the number of data (`RAW`) frames.

7.1.3 Product Frames

Default File Name	Frame Tag	Description
esotk_object_mask_0000.fits	OBJ_MASK	object mask

The product frames are single extension or multi extension frames depending on the input frames. They contain a mask that flags all pixels belonging to an object with unity (1) whereas all other (background) pixels are not

flagged, i.e. they are set to 0[1].

7.1.4 Recipe Parameters

Parameter	Type	Values	Description
obj.min-pixels	int	**4**	Minimum pixel area for each detected object.
obj.threshold	double	**3.**	Detection threshold in sigma above sky.
obj.deblending	boolean	**TRUE**	Deblend overlapping objects if set to true
obj.core-radius	double	**5.0**	Value of Rcore in pixels.
bkg.estimates	boolean	**TRUE**	Estimate background from input, if false it is assumed the input is already background corrected with median 0.
bkg.mesh-size	int	**64**	Background smoothing box size.
bkg.smooth-gauss-fwhm	double	**2.0**	The FWHM of the Gaussian kernel used in convolution for object detection.
det.effective-gain	double	**3.0**	Detector gain value. Used to convert intensity to electrons.
det.saturation	double	**infinity**	Detector saturation value.
pfm	string	closing, dilation **closing**	Post filtering mode. This mathematical operation is applied to the object mask after object detection.
pfx	int	**3**	X size of the post filtering kernel. Only odd values are allowed
pfy	int	**3**	Y size of the post filtering kernel. Only odd values are allowed

7.2 esotk_masterfringe_create

This recipe derives a master fringe frame from the input images. Properly scaled, it can be used to efficiently remove the fringes from single frames and improve their photometric accuracy.

7.2.1 Short algorithm description

The algorithm consists of two main steps: i) Using a Gaussian mixture model the recipe estimates the background (a scalar) and the fringe scaling-factor (amplitude - a scalar) for each image tagged as `RAW_BPM`. The

[1] The flagging follows the CPL convention

background is then subtracted and the amplitude used to normalize the background-subtracted image to the same scale. ii) The resulting images are then collapsed into a master-fringe image. There are different collapsing methods available (for details of the more sofisticated algorithms see appendix A) and they are controlled by the `-collapse.` recipe parameters. In other words: the amplitude of the fringes is computed for each input image (RAW) and used to re-scale each of them before stacking.

Different masks controlling the algorithm in different stages can be passed to the recipe:

A) `RAW_BPM`: The bad pixel mask of each single input image (one for each `RAW` image). Each step in the algorithm takes this mask into account.

B) `OBJ_MASK`: Masks flagging the objects in the input image (one for each `RAW` image). Each step in the algorithm takes this mask into account.

C) `FRINGE_MASK`: A single mask. This mask is only taken into account in step i), i.e. when computing the image background and fringe amplitude for each RAW image. Step ii), i.e. the collapsing algorithm ignores this mask. Please note that if the fringe mask has mask-data only in the primary extension or only in the first extension but the raw frames are multi-extension files, the mask-data is applied to all extensions. This allows the user to e.g. exclude the borders of all extensions in step i) without the need to create a multi-extension mask-file. Part of the code is paralelised with OpenMP[2].

7.2.2 Input Frames

Frame Tag	Type	Count	Description
RAW	raw	1..n	Data image
RAW_BPM	bpm	1..n	Bad pixel map
RAW_ERROR	error	1..n	Error image
RAW_VARIANCE	variance	1..n	Variance image
OBJ_MASK	object	1..n	Object mask
FRINGE_MASK	fringe	1	Fringe mask

The recipe assumes that all data input frames are either single extension frames with the data in the primary section only, or multi extension frames. If they are multi extension frames the primary extension is supposed to be empty and is not read. Moreover, if optional bad pixel, error, variance, and/or object masks are provided, the number (and dimension) of each optional input provided must match the number of data (`RAW`) frames.

7.2.3 Product Frames

Default File Name	Frame Tag	Description
esotk_masterfringe.fits	MASTER_FRINGE	master fringe frame
esotk_masterfringe_error.fits	MASTER_FRINGE_ERROR	master fringe error frame

continued on next page

[2] See https://www.openmp.org

continued from previous page

Default File Name	Frame Tag	Description
esotk_masterfringe_contribution.fits	MASTER_FRINGE_CONTRIBUTION	master fringe contribution mask
esotk_masterfringe_bpm.fits	MASTER_FRINGE_BPM	master fringe bad pixel mask

The product frames are single extension or multi extension frames depending on the input frames. The recipe derives the master fringe frame (`MASTER_FRINGE`), the corresponding error frame (`MASTER_FRINGE_ERROR`), the contribution mask (`MASTER_FRINGE_CONTRIBUTION`) and the bad pixel mask (`MASTER_FRINGE_BPM`). The contribution mask contains the number of pixels that contributed to the creation of the master fringe frame.

7.2.4 Recipe Parameters

Parameter	Type	Values	Description
collapse.method	string	MEAN, WEIGHTED_MEAN, MEDIAN, SIGCLIP, MINMAX **MEDIAN**	Method used for collapsing/combining the data to derive the master fringe frame
collapse.sigclip.kappa-low	double	**3.0**	Low kappa factor for kappa-sigma clipping algorithm.
collapse.sigclip.kappa-high	double	**3.0**	High kappa factor for kappa-sigma clipping algorithm.
collapse.sigclip.niter	int	**5**	Maximum number of clipping iterations for kappa-sigma clipping.
collapse.minmax.nlow	int	**1**	Low number of pixels to reject for the minmax clipping algorithm.
collapse.minmax.nhigh	int	**1**	High number of pixels to reject for the minmax clipping algorithm.

7.3 esotk_masterfringe_correct

This recipe properly scales the master fringe frame and removes the fringes from the input images.

7.3.1 Short algorithm description

The algorithm consists of two main steps: i) The recipe estimates the background and the fringe scaling-factor (amplitude) for the master-fringe image (MASTER_FRINGE) of each image tagged as RAW. The required offset and scaling factor is determined using a least-square fit of the master-fringe image to the RAW image. ii) The properly scaled (if rescale=TRUE) master-fringe frame is then subtracted from each individual RAW frame.

Different masks controlling the algorithm in different stages can be passed to the recipe:

A) RAW_BPM: The bad pixel mask of each single input image (one for each RAW image). Each step in the algorithm takes this mask into account.

B) OBJ_MASK: Masks flagging the objects in the input image (one for each RAW image). These masks are only taken into account in step i), i.e. when computing the scaling factors between MASTER_FRINGE and RAW, but ignored for step ii).

C) FRINGE_MASK: A single mask. This mask is only taken into account in step i), i.e. when computing the scaling factors between MASTER_FRINGE and RAW, but ignored for step ii). Please note that if the fringe mask has mask-data only in the primary extension or only in the first extension but the raw frames are multi-extension files, the mask-data is applied to all extensions. This allows the user to e.g. exclude the borders of all extensions in step i) without the need of creating a multi-extension mask-file. Part of the code is paralelised with OpenMP[3].

7.3.2 Input Frames

Frame Tag	Type	Count	Description
RAW	raw	1..n	Data image
RAW_BPM	bpm	1..n	Bad pixel map
RAW_ERROR	error	1..n	Error image
RAW_VARIANCE	variance	1..n	Variance image
OBJ_MASK	object	1..n	Object mask
FRINGE_MASK	fringemask	1	Fringe mask
MASTER_FRINGE	masterfringe	1	Master fringe
MASTER_FRINGE_BPM	bpm	1	Master fringe bad pixel mask
MASTER_FRINGE_ERROR	error	1	Master fringe error

7.3.3 Product Frames

Default File Name	Frame Tag	Description
esotk_fringecorrected_n.fits	FRINGE_CORRECTED	fringe corrected frame
esotk_fringecorrected_n_err.fits	FRINGE_CORRECTED_ERROR	fringe corrected frame error

continued on next page

[3] See https://www.openmp.org

continued from previous page		
Default File Name	**Frame Tag**	**Description**
esotk_fringecorrected_n_bpm.fits	FRINGE_CORRECTED_BPM	fringe corrected frame bpm

The product frames are single extension or multi extension frames depending on the input frames. Additional to the fringe corrected frames (`FRINGE_CORRECTED`) the recipe also determines the corresponding error frame (`FRINGE_CORRECTED_ERROR`), and the bad pixel mask (`FRINGE_CORRECTED_BPM`).

7.3.4 Recipe Parameters

Parameter	Type	Values	Description
rescale	boolean	**TRUE**	measure fringe amplitude and rescale fringe correction before subtracting?.

7.4 esotk_spectrum1d_combine

This recipe combines a list of input spectra into a single product spectrum and is currently used by the QC group to create OB-stacked data products[4]. In order to create products that follow the ESO 1D spectroscopic standard[5] (IDP) the input must be fully compliant with this standard. If this is not the case, there is also the possibility to relax the IDP input checks with a recipe parameter and stack non IDP compliant spectra. Obviously, in this case the output is also not fully IDP compliant.

7.4.1 Short algorithm description

The recipe first resamples (by interpolation) all input spectra (`SPECTRUM_1D`) to a common output grid and then combines/stacks them. The user can choose between different interpolation and combination methods using by dedicated recipe parameters (see table 7.4.4). For the interpolation of the spectrum we rely on the GSL implementation. Moreover, the user can also provide the desired output grid (`SPECTRUM_COLLAPSE_WLENGTHS`), or, if no destination wavelengths are specified, the wavelengths of the first spectrum are used instead.

If the ESO 1D spectroscopic standard should be followed (by setting the recipe parameter `--noIDP` to `FALSE`), the FLUX (or FLUX_REDUCED) column must contain the spectrum flux, the ERR (or ERR_REDUCED) column, the spectrum error and the WAVE column the wavelengths. The QUAL column (optional, controlled by the recipe parameter `--use-quality-column`) indicates a bad pixel if the value is different from 0. See the ESO 1D spectroscopic standard for more information.

If the ESO 1D spectroscopic standard is not mandatory (by setting the recipe parameter `--noIDP` to `TRUE`), the extension and the columns containing the data can be set by appropriate recipe parameters (see section 7.4.4 for the parameter). Moreover, not only the usage of the quality flag is optional in this mode (controlled

[4] See e.g. OB-stacked data products for GIRAFFE
[5] See ESO Science Data Products Standard

by `--use-quality-column`) but also the presence of a column containing the error (controlled by the presence or absence of the column name specified by `--noIDP.colname-err` in the table).

In order to be able to combine a very low number of spectra (e.g. N = 2) which are affected by cosmic ray events, a preprocessing step was implemented. The preprocessing algorithm first collapses the stack of input spectra by using the median to generate a master spectrum. Then it subtracts the master spectrum from each individual spectrum and derives the bad pixels on the residual spectra by thresholding, rejecting all pixels exceeding the thresholds. This works very well for e.g. GIRAFFE spectra when one sets the lower threshold to such a high value (e.g. 1e6) that it has no effect and then filter only for cosmics (positive outliers). Usually when combining more than 2 or 3 spectra the preprocessing step is unnecessary because the sigma clipping algorithm (if used) suppresses the cosmics very efficiently.

Additionally to the previously mentioned preprocessing, the user can normalize the input spectra to the flux level of the first input file by setting the recipe parameter `--rescale-spectra` to `TRUE`. The scaling is done by dividing by the median of the flux level. Errors are propagated as appropriate. Please note that the errors reported in the spectrum are used to calculate the error of the scaling factor. As in some cases these errors are extremely high also the error associated with the scaling factor is large, which in turn leads to a decreased SNR of the stacked spectrum. This is not an error in the recipe algorithm but a signature of a problem present in the input spectra.

7.4.2 Input Frames

Frame Tag	Type	Count	Description
SPECTRUM_1D	spectrum	1..n	1D spectrum
SPECTRUM_COLLAPSE_WLENGTHS	wavelength	1	Destination wavelengths

The input file format has to follow the ESO 1D spectroscopic standard.

7.4.3 Product Frames

Default File Name	Frame Tag	Description
combined_spectrum.fits	ESOTK_SPECTRUM_COMBINED	combined spectrum
combined_spectrum_idp_format.fits	ESOTK_SPECTRUM_IDP_FORMAT	combined spectrum in IDP format

The main product of the recipe is `ESOTK_SPECTRUM_IDP_FORMAT`, i.e. the combined/stacked spectrum that follows the ESO 1D spectroscopic standard, whereas the second product (`ESOTK_SPECTRUM_COMBINED`) contains mostly the same information but is not following the standard.
Please note that the `ESOTK_SPECTRUM_COMBINED` product is the only product, if the ESO 1D spectroscopic standard is not mandatory (by setting the recipe parameter `--noIDP` to `TRUE`).

7.4.4 Recipe Parameters

Parameter	Type	Values	Description
use-quality-column	boolean	**FALSE**	If TRUE and quality (QUAL column) exists and is not 0 the pixel is rejected. If FALSE (or if the column does not exist) all the pixels are assumed as valid.
rescale-spectra	boolean	**FALSE**	If TRUE all the spectra are multiplicatively rescaled to the (median) level of the first spectra in the SOF.
output-columns	Comma separated string	**ALL**	Comma separated string specifying the output columns. The "WAVE" column can be omitted as it is a mandatory column. The other columns can be selected (e.g. "FLUX,ERR,SNR"). If ALL is set, all columns computed by the recipe will be saved.

continued on next page

continued from previous page

Parameter	Type	Values	Description
copykeys	Comma separated string		Comma separated string specifying a list of keywords. If a keyword in this list is found in any of the input spectra, the keyword will be propagated to the output spectrum with the prefix HIERARCH ESO STACK and the number of the input file as suffix (e.g. copykeys='ESO.SEQ.CUMOFFSETA,-ESO.TEL.AIRM.START,-ESO.TEL.AIRM.END,-OBSERVER' –> HIERARCH ESO STACK SEQ CUMOFFSETA1, HIERARCH ESO STACK SEQ CUMOFFSETA2, ... HIERARCH ESO STACK TEL AIRM START1, HIERARCH ESO STACK TEL AIRM START2, ... HIERARCH ESO STACK TEL AIRM END1, HIERARCH ESO STACK TEL AIRM END2, ... HIERARCH ESO STACK OBSERVER1, HIERARCH ESO STACK OBSERVER2, ...) If the keyword to copy is a HIERARCH ESO keyword, the HIERARCH ESO is stripped before appending.
interpolation.method	string	LINEAR, CSPLINE, AKIMA **AKIMA**	Method used for Spectrum1D interpolation.
collapse.reject-bpm	boolean	**TRUE**	If TRUE rejects an interpolated wavelength whose neighbor (in the original spectrum) is a bad pixel. For the rejection to occur only one bad neighbor is sufficient.
collapse.method	string	MEAN, WEIGHTED_MEAN, MEDIAN, SIGCLIP, MINMAX **MEDIAN**	Method used for collapsing the data. For details of the more sofisticated algorithms see appendix A
collapse.sigclip.kappa-low	double	**3.0**	Low kappa factor for kappa-sigma clipping algorithm.

continued on next page

continued from previous page

Parameter	Type	Values	Description
collapse.sigclip.kappa-high	double	**3.0**	High kappa factor for kappa-sigma clipping algorithm.
collapse.sigclip.niter	int	**5**	Maximum number of clipping iterations for kappa-sigma clipping.
collapse.minmax.nlow	int	**1.0**	Low number of pixels to reject for the minmax clipping algorithm.
collapse.minmax.nhigh	int	**1.0**	High number of pixels to reject for the minmax clipping algorithm.
bpm.enable	boolean	**FALSE**	If TRUE the recipe tries to detect and mark bad pixels, e.g. cosmics on the single aligned input spectra by following the algorithm described in section 7.4.1. If FALSE the bpm parameters are ignored.
bpm.kappa-low	double	**3.0**	Low RMS scaling factor for image thresholding.
bpm.kappa-high	double	**3.0**	High RMS scaling factor for image thresholding.
bpm.method	string	absolute, relative, error **relative**	Thresholdig method to use for bpm detection.
noIDP	boolean	**FALSE**	If TRUE, the recipe will not check if the input is IDP compliant and will not produce an IDP compliant output. Moreover, the user can specify the extension and the column names - see parameters below.
noIDP.extension	int	**1**	Fits extension containing the spectrum table.
noIDP.colname-wave	string	**WAVE**	Name of the column containing the wavelength for a non IDP input.
noIDP.colname-flux	string	**FLUX**	Name of the column containing the flux for a non IDP input.
noIDP.colname-err	string	**ERR**	Name of the column containing the flux error for a non IDP input.
noIDP.colname-qual	string	**QUAL**	Name of the column containing the quality/bpm for a non IDP input.

7.4.5 QC Parameter

The following QC parameters are written by the recipe:

- QC.AIRM.AVG: This keyword is not recalculated frame by frame by esotk_spectrum1d_combine but expects that the instrument pipeline itself writes the value in the header of each IDP. The esotk_spectrum1d_combine recipe will then average these keywords and write the average to the pipeline product. If the keywords are not in the input frames the default value in the product will be -1
- QC.IA.FWHM.AVG: This keyword is not recalculated frame by frame by esotk_spectrum1d_combine but expects that the instrument pipeline itself writes the value in the header of each IDP. The esotk_spectrum1d_combine recipe will then average these keywords and write the average to the pipeline product. If the keywords are not in the input frames the default value in the product will be -1
- QC.IWV.AVG: This keyword is not recalculated frame by frame by esotk_spectrum1d_combine but expects that the instrument pipeline itself writes the value in the header of each IDP. The esotk_spectrum1d_combine recipe will then average these keywords and write the average to the pipeline product. If the keywords are not in the input frames the default value in the product will be -1
- QC.CONTRIB.AVG: This keyword contains the mean contribution of the stacked spectra as derived from the CONTRIB column.
- QC.CONTRIB.REL: This keyword contains the relative mean contribution of the stacked spectra as derived from the CONTRIB column, i.e. QC.CONTRIB.REL = QC.CONTRIB.AVG / Number-of-Frames
- QC.SNR.AVG: Mean Signal-to-Noise ratio excluding pixels with SNR ≤ 0
- QC.SNR.RATIO: QC.SNR.AVG / (mean SNR of input frames). The mean SNR of the input frames is calculated by averaging the SNR keyword of the input headers
- QC.SNR.IMPROV: QC.SNR.RATIO / sqrt(number of input frames).
- QC.SQRTNCOM: sqrt(number of input frames)
- QC.MAXDEV: The relative maximum scaling deviation. To calculate this keyword the scaling factors (controlled by the recipe parameter `--rescale-spectra`) are used. The QC parameter is defined as the maximum of the scaling values (i.e. the median of the flux spectrum i) as follows

$$\frac{|scale[i] - \overline{scale}|}{|\overline{scale}|}$$

Please note that whenever there are the standard IDP columns in the table, the normal columns are used to derive the QC parameter, if not, the `_REDUCED` columns are used. Moreover, bad pixels are always excluded in all calculations.

7.5 esotk_barycorr

The recipe derives the barycentric correction of an observation by using the ERFA (Essential Routines for Fundamental Astronomy) library and saves the correction in the product header using the keyword `ESO.DRS.BARYCORR`. ERFA is a C library containing key algorithms for astronomy, and is based on the SOFA library published by the International Astronomical Union (IAU).
On request the recipe also applies the barycentric correction to data stored in table format. Moreover, esotk_barycorr alters the input headers (primary, extensions) as little as possible to preserve the IDP standard if the input already follows this standard and updates a few wavelength related IDP keywords.

7.5.1 Short algorithm description

The recipe uses the ERFA function eraApco13() to calculate the barycentric correction of an observation. For details on the implemented algorithm please visit the above mentioned web pages and links therein.

A comparison between the here implemented algorithm and the one implemented in the ESPRESSO pipeline shows a very good agreement. For this about 7000 ESPRESSO IDPs from 2021 were re-analyzed with the ERFA based implementation and the differences read as follows:

- Mean difference : 0.036 m/s
- Median difference: 0.055 m/s
- Standard deviation: 0.281 m/s
- MAD (median absolute deviation): 0.217 m/s
- Maximum deviation: 1.573 m/s

The esotk_barycorr recipe allows the user to read the input values (e.g. ra, dec, ...) requested by the ERFA routine from the header of the frames by specifying the header keywords containing these values. For this the various recipe parameters (e.g. `barycorr.hname-ra`, `barycorr.hname-dec`, ...) can be used.
Please note that the pressure, the temperature, the humidity, and the wavelength (recipe) parameters were always kept to their default value of 0. No tests with other values were performed.
The derived barycentric correction is saved in the product header using the keyword `ESO.DRS.BARYCORR`, in km/s units, e.g.
`HIERARCH ESO DRS BARYCORR = 18.3512096944018 / Barycentric correction [km/s]`
This keyword is always saved independent if the user also requests to apply the correction or not.

In order to apply the correction to the data, three prerequisites must be fulfilled:

1. The user requests to apply the correction by setting the recipe parameter `barycorr.apply` to TRUE
2. The data are not yet corrected, i.e. the `SPECSYS` header keyword is set to `TOPOCENT` or does not exist at all
3. The input data are stored in table format.

If all prerequisites are fulfilled the correction (*barycorr*) is applied to the wavelength columns (λ) as follows

$$\lambda_{corr} = \lambda_{orig} \times (1 + barycorr/c)$$

with c being the speed of light. The SPECSYS primary header keyword is changed into BARYCENT and a few IDP related header keywords (e.g. WAVELMIN, WAVELMAX, ...) are updated accordingly.

7.5.2 Input Frames

Frame Tag	Type	Count	Description
RAW	raw	1..n	Frame containing the data to correct
EOP_PARAM	calib	1	Earth Orientation Parameter file

The recipe derives the barycentric correction by using keywords from the primary header of the RAW frame. Therefore, the input RAW file can be of practically any type (1D-, 2D-, 3D-images, table, ...) as long as the requested keywords are present. When it comes to apply the barycentric correction, only the table format is supported. Moreover, the header of the output product is as close as possible to the header of the input product.

Additionally to the RAW frame, the EOP_PARAM frame has to be passed to the recipe. The latter contains the Earth Orientation Parameter as a function of time (MJD-OBS). As the time resolution of the file is only one day, the parameters are interpolated by the recipe to have the most accurate values at the time of observation. A EOP file is included as static calibration in the pipeline distribution, but one can also use the recipe esotk_eop (see section 7.6) to download the most up to date file from IERS.

Please note that the SOF can contain many RAW frames that can also be completely independent (even from different instruments). The recipe loops over all files and tries to process them. If for some reason the barycentric correction can not be derived for one file (e.g. a requested header parameter for this file is not present or has a different name) this file is simply skipped and the next one is processed. Obviously there will be a corresponding warning in the log file.

7.5.3 Product Frames

Default File Name	Frame Tag	Description
esotk_barycorr_0000.fits	ESOTK_BARYCORR	Barycentric corrected file and/or a primary header with the keyword DRS.BARYCORR containing the correction itself

Depending on the input data and recipe parameters, the product frames contain

- The barycentric corrected data and the correction itself in the header keyword DRS.BARYCORR
- A primary header only with the correction itself written to the keyword DRS.BARYCORR

7.5.4 Recipe Parameters

The following recipe parameters can be used to control the recipe:

Parameter	Type	Description
barycorr.hname-ra	string	Name of the header keyword containing the target right ascension (J2000) [deg]
barycorr.hname-dec	string	Name of the header keyword containing the target declination (J2000) [deg]
barycorr.hname-mjdobs	string	Name of the header keyword containing the mjd-obs, i.e. start of observation [d]
barycorr.hname-exptime	string	Name of the header keyword containing the exposure time [s]
barycorr.hname-longitude	string	Name of the header keyword containing the longitude of the observatory, i.e. the telescope geodetic longitude (+ = East) [deg]
barycorr.hname-latitude	string	Name of the header keyword containing the latitude of the observatory, i.e the telescope geodetic latitude (+ = North) [deg]
barycorr.hname-elevation	string	Name of the header keyword containing the elevation of the observatory, i.e. the telescope elevation above sea level [m]
barycorr.pressure	double	Pressure during observation, i.e. pressure at the observer [hPa == mbar]
barycorr.temperature	double	Temperature during observation, i.e. ambient temperature at the observer [deg C]
barycorr.humidity	double	Humidity during observation, i.e. relative humidity at the observer [range 0 - 1]
barycorr.wavelength	double	Wavelength during observation, i.e. observing wavelength [micrometer]
barycorr.apply	boolean	Apply barycentric correction to the wavelength column.
barycorr.extension	int	Fits extension containing the spectrum table.
barycorr.colname-wave	string	Name of the table column containing the wavelength to be corrected.

7.6 esotk_eop

This recipe downloads the latest version of the Earth Orientation Parameter and DUT[6] (Difference to Universal Time: UT1-UTC) from IERS. The IERS Earth Orientation Parameters (EOP) describe the irregularities of the earth's rotation. It stores the parameter in a fits-file that is used as input for the computation of the barycentric correction (see section 7.5).

7.6.1 Short algorithm description

Download of the latest version of the Earth Orientation Parameter and DUT from IERS.

[6]The time correction equal to the difference between Universal Time (UT1), which is defined by Earth's rotation, and Coordinated Universal Time (UTC), which is defined by a network of precision atomic clocks.

7.6.2 Input Frames

No input frame needed - use empty SOF or no SOF.

7.6.3 Product Frames

Default File Name	Frame Tag	Description
esotk_eop_param.fits	EOP_PARAM	File containing the Earth Orientation Parameter

The product file contains the EOP table that is used as input for the computation of the barycentric correction (see section 7.5).

7.6.4 Recipe Parameters

The following recipe parameters can be used to control the recipe:

Parameter	Type	Description
eop_host	string	Host to retrieve the EOP from. [https://datacenter.iers.org]
eop_urlpath	string	URL path of the EOP file to retrieve. [/products/eop/rapid/standard/finals2000A.data]
eop_usertag	string	TAG provided by the user. [None]
eop_instrument	string	Product header keyword value for INSTRUME. [ESOTK]

A Statistical Estimators

A.1 Arithmetic mean - Arithmetic weighted mean

The arithmetic mean of a sample $x_1, x_2, \ldots, x_N$ is defined as

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \,. \tag{1}$$

For a random variable X that is normally distributed ($N(\mu, \sigma)$), the sample mean is the most efficient estimator of the population mean μ. By the central limit theorem this statement holds asymptotically for many noise distributions encountered in practice.

If the noise distribution of X is unknown, its variance can be estimated from the sample as

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N} (x_i - \overline{x})^2 \,. \tag{2}$$

If X has variance σ then the average (1) has variance

$$\sigma^2_{\rm av} = \frac{\sigma^2}{N}. \tag{3}$$

This relation can be used in place of Eq. (2) if an *a priori* estimate of the noise in the images is available and is the same for all images (homescedastic case).

When each image has its own noise image $\sigma_i^2(x, y)$ (heteroscedastic case [7]), then the weighted mean

$$\overline{x} = \left(\sum \frac{x_i}{\sigma_i^2}\right)\left(\sum \frac{1}{\sigma_i^2}\right)^{-1} \tag{4}$$

can be used, giving higher weight to values with lower noise. The variance of the average value at a given pixel is in this case given by

$$\frac{1}{\sigma^2_{\rm av}} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \,. \tag{5}$$

While the average is computationally efficient, it is also sensitive to the presence of outliers, i.e. sample values which arise from a process which is not included in the $N(\mu, \sigma)$ model of the background noise. Cosmic ray hits and the like that affect single exposures in the set therefore show up clearly in the stacked image. The average fails in the presence of a single outlier in the sample.

A.2 Median

For the computation of the sample median, the sample $\{x_i,\ i = 1 \ldots N\}$ is first sorted such that $x_1 < x_2 < \cdots < x_N$. The median is then the 50th percentile of the ordered set, i.e.

$$x_{\rm med} = x_{\frac{N+1}{2}} \quad \text{if } N \text{ is odd} \tag{6}$$

[7] In statistics, a vector of random variables is heteroscedastic if the variability of the random disturbance is different across elements of the vector.

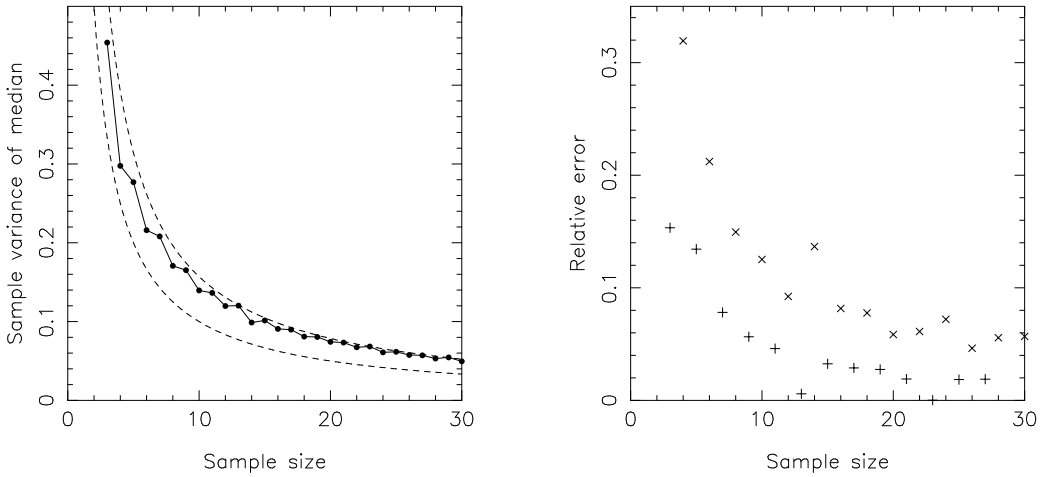


Figure A.1: Sample variance of the median as a function of sample size (left panel). The lower dashed curve is the expected variance of the arithmetic mean, the upper dashed curve the asymptotic variance of the median. The right panel shows the relative error compared to the asymptotic variance. Crosses are for even sample sizes, pluses for odd sample sizes.

and

$$x_{\mathrm{med}} = \frac{1}{2}(x_{\frac{N}{2}} + x_{\frac{N}{2}+1}) \quad \text{if } N \text{ is even.} \tag{7}$$

This definition of the median for even-sized samples ensures that the median is an unbiased estimator of the population mean μ for a Gaussian error distribution or indeed any distribution that is symmetric about its mean.

The sample median has a variance larger than the sample average, i.e. it is less efficient as an estimator of the population mean. It is however robust against outliers in the sample since only the central one or two sample values are actually used for the computation. The median only fails if half or more of the sample values are outliers, i.e. if a given pixel is affected by a cosmic ray hit in half or more of the images in the stacking list.

While the sample distribution of the median (which is not Gaussian) can be written down easily, the computation of its moments and hence its variance is difficult. A scheme to compute it analytically has been described in [2] and exact values for a few sample sizes and a number of parent distributions are given in [3]. However, these are hardly useful for practical purposes. The asymptotic value for the ratio of the variances of mean and median (the *asymptotic relative efficiency* of the median, [1]) in the Gaussian case is

$$\sigma^2_{\mathrm{med}} = \frac{\pi}{2}\sigma^2_{\mathrm{av}} = \frac{\pi\sigma^2}{2N}. \tag{8}$$

In Fig. A.1, we determine the variance of the median from simulations by drawing for any given sample size 10 000 samples from a standard normal distribution. The asymptotic value is approached from below, which makes it a conservative choice for an estimate of the median's variance. The relative error, defined as

$$\frac{\pi/2N - \hat{\sigma}^2_{\mathrm{med}}}{\pi/2N}, \tag{9}$$

is plotted in the right hand panel of Fig. A.1. For even sample sizes, the relative error is significantly larger than for odd sizes because the variance is actually lower. For odd-sized samples, the error is below 10% for $N \geq 7$

($N \geq 5$ according to [3]), whereas for even sized samples this threshold is not crossed until $N \gtrsim 12$. However, as mentioned above, the asymptotic value in Eq. (8) is conservative for any N and it is therefore used.

When each image has its own noise image $\sigma_i^2(x, y)$ (heteroscedastic case), then a weighted median can be defined as the point where cumulative weights is equal to $1/2$. Specifically, the weighted median of an ordered sample of N values x_i with weights w_i is

$$x_{\mathrm{wmed}} = \begin{cases} \dfrac{x_j + x_{j+1}}{2}, & \text{if} \quad \sum_{i=1}^{j} w_i = 0.5 \\ x_j, & \text{if} \quad \sum_{i=1}^{j} w_i > 0.5 \quad \text{and} \quad \sum_{i=1}^{j-1} w_i < 0.5 \end{cases} \tag{10}$$

A.3 Kappa-sigma clipping

In the min-max rejection algorithm, a fixed number of sample values are rejected, regardless whether they can be identified as outliers or not. If one wants to retain all "good" sample values and only reject true outliers, one has to employ an adaptive method which compares each sample value to the distribution of the entire sample.

In the $\kappa\sigma$ clipping algorithm, all values that deviate from the mean by more than κ standard deviations are rejected as outliers. Typically, $\kappa = 3$. The mean is usually estimated from the data, as is the standard deviation if no independent error estimate is available. One can introduce an iteration which stops once no more values are rejected.

Standard $\kappa\sigma$ clipping is not very efficient in removing low-significance outliers because the estimates of the mean and the standard deviation are themselves affected by the outliers. Using more robust estimators of location and scale, like the median and the inter-quartile range (IQR) or the median absolute deviation (MAD), improves the situation and makes $\kappa\sigma$ clipping a robust and easy-to-use adaptive outlier-rejection algorithm. This implementation uses the median and the MAD as robust estimators for the location and scale. For a Gaussian distribution, the standard deviation σ is given in terms of the MAD through

$$\sigma = \mathrm{MAD} \times 1.4826. \tag{11}$$

$\kappa\sigma$ clipping has one parameter, the clipping threshold κ. The value of κ is not critical if outliers are expected to lie far from the sample distribution, as is the case for cosmic ray hits. Weaker effects, such as satellite trails, may require fine-tuning of κ.

The functions that make use of $\kappa\sigma$ clipping have a second parameter, e.g. `niter` which specifies the maximum number of iterations to run.

The method might fail if more than half of the input images are affected by outliers at a given pixel because then the estimate of the MAD might be affected and cause outlier rejection to fail. The method also requires a sufficient number of input images to be able to obtain reasonably accurate estimates of the mean and standard deviation.

The variance of the estimator now varies from pixel to pixel depending on the number of values that are rejected as outliers. Since outliers are not drawn from the same distribution as the "good" data values, the cleaned sample is equivalent to a smaller sample drawn from the noise distribution. In the general heteroscedastic case, the variance is therefore estimated as

$$\sigma_{\kappa\sigma}^2 = \frac{1}{N_{\mathrm{good}}^2} \sum_{\mathrm{good}} \sigma_i^2 \,. \tag{12}$$

B Installation

ESO pipelines can be installed via several methods, depending on your OS, most of which facilitate easy installation, upgrade and removal. Please see the "ESO Data Reduction Pipelines and Workflow Systems" page.

C Troubleshooting

[1] Anirban DasGupta. *Asymptotic Theory of Statistics and Probability*. Springer, New York, 2008. 32

[2] J. S. Maritz and R. G. Jarrett. A note on estimating the variance of the sample median. *Journ. Am. Stat. Assoc.*, 73:194–196, mar 1978. 32

[3] Paul R. Rider. Variance of the median of small samples from several special populations. *Journ. Am. Stat. Assoc.*, 55:148–150, mar 1960. 32, 33