# *EUMETSAT Data Tailor - User Guide*

***This Document is Public***

| | | |
|---|---|---|
| Doc.No. | : | EUM/SEP/DOC/19/1126669 |
| Issue | : | v1D e-signed |
| Date | : | 16 December 2019 |
| WBS/DBS | : | |

EUMETSAT
Eumetsat-Allee 1, D-64295 Darmstadt, Germany
Tel: +49 6151 807-7
Fax: +49 6151 807 555
http://www.eumetsat.int

***This Document is Public***

## *Change Record*

| ***Version*** | ***Date*** | ***DCR* No.*** *if applicable* | ***Description of Changes*** |
|---|---|---|---|
| 1C | 16/12/2019 | | First public version. |
| 1D | 16/12/2019 | | Add explanation of plugins and backends. |
| | | | |

****DCR = Document Change Request***

***This Document is Public***

## *Table of Contents*

***This Document is Public***

## *Table of Figures*

## *Table of Tables*

***This Document is Public***

# 1 GETTING STARTED

## 1.1 Overview

The EUMETSAT Data Tailor provides format conversion and basic product customisation capabilities for a set of EUMETSAT products.

The capabilities of a Data Tailor installation are provided by plug-ins, which contain one or more backends. The backends contained by a plug-in implement the functionality necessary in order to complete the requested customisations.

### 1.1.1 Functionalities

The following functionalities can be applied on input products; not all the functionalities may be available for a given product:

- Format conversion
- Aggregation
- Layer filtering
- Extraction of a region of interest (ROI)
- Reprojection
- Resampling
- Generation of a Quicklook

The functionalities can be applied in sequence, by specifying the configuration of a customisation chain. Configurations may be saved for reuse.

The specific configuration of a functionality (e.g. the extent of a ROI) can also be saved for later use; a saved configuration can be referred by name in the chain configuration.

### 1.1.2 Supported products

The current version of the EUMETSAT Data Tailor supports the following products; beside each product, its "Product type" is listed, as this is how the EUMETSAT Data Tailor identifies the product, and the platform where the support is available.

***Table 1: Products supported by the Data Tailor.***

| Platform | Product | Product type | Platform |
|---|---|---|---|
| EPS | AMSU-A L1B | AMSAL1 | All |
| | ASCAT L1B SZF | ASCATL1SZF | All |
| | ASCAT L1B SZO | ASCATL1SZO | All |
| | ASCAT L1B SZR | ASCATL1SZR | All |
| | AVHRR L1B | AVHRRL1 | All |
| | ASCAT-L2 Soil Moisture 25 km | ASCATL2SMO | All |
| | ASCAT-L2 Soil Moisture 12.5 km | ASCATL2SMR | All |
| | GOME2 L1B | GOME2L1 | All |
| | GOME L2 (PMA) | GOME_PMA_AOP | All |
| | GOME L2 (PMA) | GOME_PMA_COP | All |

| | | | |
|---|---|---|---|
| | HIRS L1B | HIRSL1 | All |
| | IASI L1C | IASIL1 | All |
| | IASI Sulphur dioxide | IASINSD02 | All |
| | MHS L1B | MHSL1 | All |
| LSA SAF | MDFAPAR | MDAFAPAR | All |
| | MET | MET | All |
| | METREF | METREF | All |
| | MLST | MLST | All |
| MSG | HRSEVIRI L1.5 | HRSEVIRI | All Output in HRIT, ISCCP formats only supported on Linux |
| | HRSEVIRI HRIT L1.5 | HRSEVIRI_HRIT | All Output in ISCCP formats only supported on Linux |
| | HRSEVIRI HRIT L1.5 (High Resolution Band) | HRSEVIRI_HRIT_HRV | All Output in ISCCP formats only supported on Linux |
| | MSGAMVE | MSGAMVE | All Linux |
| | MSGCLAP | MSGCLAP | All Linux |
| | MSGFIRG | MSGFIRG | All |
| | MSGMPEG | MSGMPEG | All |
| | MSGOCAE | MSGOCAE | All |
| | Cloud Mask | MSGCLMK | All |
| MTG | MTG IRS L1 Principal Component | MTGIRSL1 | All Linux |
| OSI SAF | Global SST | GLBSST | All |
| | ASCAT-L2 Winds and Soil Moisture (25 km) | OASW025 | All Input in BUFR only supported on Linux |
| | ASCAT-L2 Coastal Surface Wind (12.5 km) | OASWC12 | All Output in BUFR only supported on Linux |
| | ASCAT-L2 Surface Wind (25 km) -Data record 1 | OR1ASW025 | All Output in BUFR only supported on Linux |
| | ASCAT-L2 Coastal Surface Wind (25 km) - Data record 1 | OR1ASWC12 | All Output in BUFR only available on Linux |
| | ASCAT-L2 12.5 km winds data record | OR1ERW025 | All Input in BUFR only supported on Linux |
| | SeaWinds L2 25 km winds data record | OR1SWW025 | All Input in BUFR only supported on Linux |

Notes:

- The HRSEVIRI_HRIT_HRV product type allows to customise the High Resolution Band (HRV) of the HRSEVIRI HRIT L1.5 product
- The GOME_PMA_AOP and GOME_PMA_COP product types allow to customise the Aerosol Optical Properties and the Cloud Optical Properties layers respectively.

## 1.2 Supported platforms and installation

The EUMETSAT Data Tailor can be installed on:

- CentOS Linux 7 64bit
- Ubuntu Linux 18.04 64bit
- Windows 10 Pro 64bit
- RedHat Enterprise Linux 7 64 bit
- CentOS Linux 6 64bit

The minimum hardware requirements are:

- 64bit platform
- 4 GB of disk space
- 4 GB of free memory

Installation procedures are described below.

## 1.3 Install the EUMETSAT Data Tailor

This document describes how to build and install the EUMETSAT Data Tailor core components and the customisation plugins from the source code, for the following Operating Systems:

- Ubuntu Linux 18.04 64 bit
- CentOS v.7 64 bit
- Windows 10 Professional 64bit

The Installation of the EUMETSAT Data Tailor needs at least (note however that to build the packages, at least 6 GB of RAM are needed):

- 64bit platform
- 3 GB of disk space
- 4 GB of free memory

Building and installation require:

- the Data Tailor source code.
- conda, installed as described here.

If you already have a Data Tailor installation, you will need to either back it up, e.g. by moving it to a new directory, or remove it before completing the installation. This can be done as follows:

```
conda env remove -n epct-desktop
```

## 1.4 Additional software pre-requisites for the core components (Windows 10 Pro)

- Visual Studio 2015 with C++ installed

Install conda-build:
```
conda install -y conda-build
```

Enter the epct source directory.

Build the epct package:
```
conda build --override-channels -c anaconda -c conda-forge -c local epct
```

Build the epct-restapi package:

```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct-restapi
```

Build the epct-webui package:

```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct-webui
```

Customisation plugins are optional packages which add customisation functionalities to the EUMETSAT Data Tailor.

At least one must be present to use the EUMETSAT Data Tailor for customisation.

### 1.4.1 epct_plugin_gis

Supported platforms: all.

To build the epct_plugin_gis, execute:

```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct_plugin_gis
```

#### 1.4.1.1 msg-gdal-driver

The msg-gdal-driver driver package which adds the support for MSG HRV products to the epct_plugin_gis.

Download or clone the Data Tailor source code, and enter the conda/msg-gdal-driver (condamsg-gdal-driver on Windows) directory.

Download the Public WaveLet Transform Decompression Library from https://gitlab.eumetsat.int/open-source/PublicDecompWT/-/archive/2.06/PublicDecompWT-2.06.zip and rename it to PublicDecompWT.zip

Put the PublicDecompWT.zip file from the Public WaveLet Transform Decompression Library in the src dir.

Go back to the epct source code root directory and execute the following to build the msg-gdal-driver package:

```
conda build --override-channels -c local -c anaconda -c conda-forge conda
```

### 1.4.2 epct-plugin-umarf

Supported platforms: Linux.

Build the epct_plugin_umarf package as follows:

```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct_plugin_umarf
```

Note: the package does not contain the executables msg15toISCCP and msg15toXrit required to use it. They must be present on the target machine in the user's path. If not, the user should contact ops@eumetsat.int.

Note: the plugin is optional. If the above executables are not present on the machine, installation of the plugin does not fail, but the plugin will not be used by EUMETSAT Data Tailor.

### 1.4.3 epct-plugin-fist-iasil1c

Supported platforms: Linux.

Build the epct_plugin_fist_iasil1c package as follows:
```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct_plugin_fist_iasil1c
```

Note: the package does not contain the IASIL1c_v9A executable from FIST required to use it. It must be present on the target machine in the user's path. If not, the user should contact ops@eumetsat.int.

Note: the plugin is optional. If the above executable is not present on the machine, installation of the plugin does not fail, but the plugin will not be used by EUMETSAT Data Tailor.

### 1.4.4 epct-plugin-netcdf-generator

Supported platforms: Linux.

Build the epct_plugin_netcdf_generator package as follows:
```
conda build --override-channels -c anaconda -c conda-forge -c local \
    epct_plugin_netcdf_generator
```

Note: the package does not contain the netcdfgenerator compiled Java and its config directory. It must be present on the target machine in the user's path. If not, the user should contact ops@eumetsat.int.

Note: the plugin is optional. If the above executable is not present on the machine, installation of the plugin does not fail, but the plugin will not be used by EUMETSAT Data Tailor.

To install the Data Tailor on:

- a Linux platform, execute the following:
  ```
  conda env create -f conda/epct-desktop-linux-environment.yaml
  ```
- on Windows 10:
  ```
  conda env create -f conda\epct-desktop-win10-environment.yaml
  ```

This create the epct-desktop "environment" with all the supported components installed. It needs to be activated in the terminal to use epct. This can be done each time a new terminal is opened, with:
```
conda activate epct-desktop
```

On Linux, it can be activated by default by adding it to .bashrc:
```
echo conda activate epct-desktop >> $HOME/.bashrc
```

Once the epct environment is active, one can use epct; e.g. to retrieve the version:
```
epct --version
```

or to verify which plugins are installed:
```
epct info
```

The GUI can be started by running:
```
epct_webui
```

Important notes:
- (Linux only) to have the epct_plugin_umarf customisation plugin working, the msg15toISCCP msg15toXrit executables must be in the user's path; they are not installed by conda
- (Linux only) to have the epct_plugin_fist_iasil1c customisation backend working, the IASIL1c_v9A executable must be in the user's path; it is not installed by conda
- (Linux only) to have the epct_plugin_netcdf_generator customisation backend working, the netcdfgenerator executable must be in the user's path or the path must be configured by the user; it is not installed by conda.

To install a customised version of the EUMETSAT Data Tailor, edit the conda/epct-desktop-linux-environment.yaml (Linux) or condaepct-desktop-win10-environment.yaml (Windows) with a text editor and remove the lines listing the unwanted packages. For a minimal desktop installation including the GUI, the following packages are needed: epct, epct_restapi, epct_webui and at least one plugin (typically epct_plugin_gis).

After saving the file, install as above:
- on a Linux platform:
  ```
  conda env create -f conda/epct-desktop-linux-environment.yaml
  ```
- on Windows 10:
  ```
  conda env create -f conda\epct-desktop-win10-environment.yaml
  ```

## 1.5 Starting to use the EUMETSAT Data Tailor

The EUMETSAT Data Tailor provides different User Interfaces, which suit the typical needs of different user categories:
- the **GUI** provides a Graphical User Interface. The GUI is targeted to novice and new users as well as generic users. They may want to use to GUI to:
  - perform basic customization of small set of products;
  - set-up the configuration to be used in later automated processing;
- the **CLI** provides a Command Line Interface that can be called directly at the command line and in shell scripts. It is targeted at users that want to develop scripts which include customisations. An example would be a system administrator of an EUMETCast Receiving Station;
- the **API** provides an Application Programming Interface that can be invoked from Python programs. Other programming languages are not supported at the moment.
- the **Web Service Interface** provides a REST web interface than can be invoked from other applications through the web; its use is therefore not described in detail in this user guide.

To start using the EUMETSAT Data Tailor through the GUI, see "Using the GUI".

To start using the EUMETSAT Data Tailor through the CLI, see "Using the EUMETSAT Data Tailor CLI".

To start using the EUMETSAT Data Tailor through the API, see "Using the EUMETSAT Data Tailor API".

## 1.6 How to get User Support

Please contact the EUMETSAT User Service Helpdesk using the E-mail: ops@eumetsat.int.

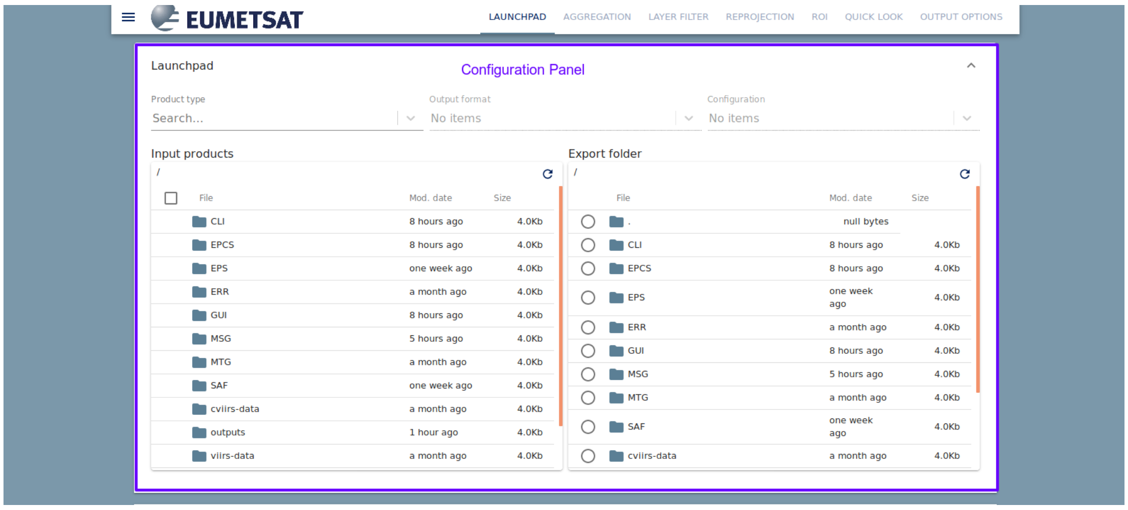
# 2 EUMETSAT DATA TAILOR - USER GUIDE

## 2.1 Using the GUI

### 2.1.1 The GUI: a description

The EUMETSAT Data Tailor Graphical User Interface is a Web Application that can be accessed using any modern browser.
It is logically divided into the following modules:

- a Tab Selection structure, where it is possible to select the main Tab to configure the processing and run (Launchpad) or other Customization Tabs where to set up and change configuration settings for each customization process;
- a Configuration Panel, where the functional chain is configured (Launchpad) and each customization process is set up (Customization tabs).
- the Action buttons, where it is possible to save the different configurations for functional chains, request the execution of customisation or open the Process Monitoring Panel. The action buttons "enter" in the page when the user hovers with the mouse on them



Product type: Search...
Output format: No items
Configuration: No items

Input products

/

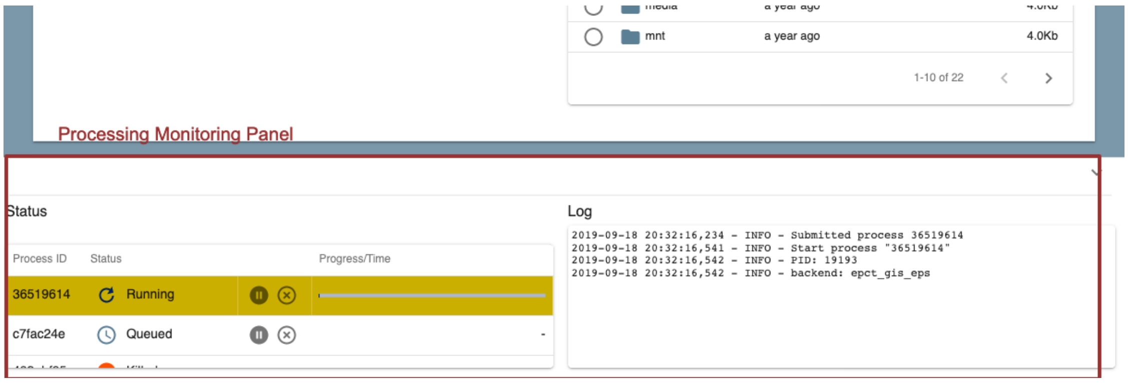
| File | Mod. date | Size |
|---|---|---|
| CLI | 8 hours ago | 4.0Kb |
| EPCS | 8 hours ago | 4.0Kb |
| EPS | one week ago | 4.0Kb |
| ERR | a month ago | 4.0Kb |
| GUI | 8 hours ago | 4.0Kb |
| MSG | 5 hours ago | 4.0Kb |
| MTG | a month ago | 4.0Kb |
| SAF | one week ago | 4.0Kb |
| cviirs-data | a month ago | 4.0Kb |
| outputs | 1 hour ago | 4.0Kb |
| viirs-data | a month ago | 4.0Kb |

Export folder

/

| File | Mod. date | Size |
|---|---|---|
| . | null bytes | |
| CLI | 8 hours ago | 4.0Kb |
| EPCS | 8 hours ago | 4.0Kb |
| EPS | one week ago | 4.0Kb |
| ERR | a month ago | 4.0Kb |
| GUI | 8 hours ago | 4.0Kb |
| MSG | 5 hours ago | 4.0Kb |
| MTG | a month ago | 4.0Kb |
| SAF | one week ago | 4.0Kb |
| cviirs-data | a month ago | 4.0Kb |

Another panel, the Process Monitoring Panel, opens when the execution of a new customisation is requested or when the User clicks on the corresponding button in the Action buttons. The panel allows to monitor the status of each process and to check the presence of warnings or errors through the Message Log.



| | | |
|---|---|---|
| mnt | a year ago | 4.0Kb |

Status

| Process ID | Status | Progress/Time |
|---|---|---|
| 36519614 | Running | |
| c7fac24e | Queued | - |

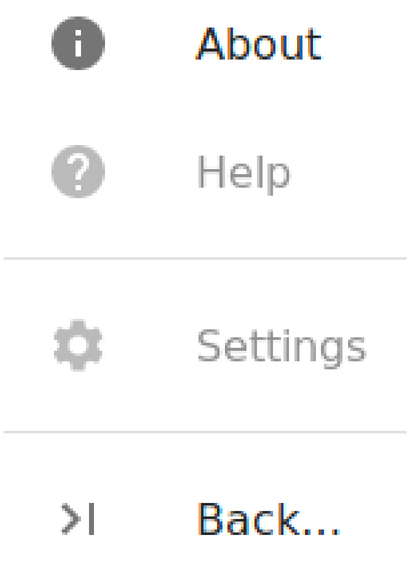
Log

```
2019-09-18 20:32:16,234 - INFO - Submitted process 36519614
2019-09-18 20:32:16,541 - INFO - Start process "36519614"
2019-09-18 20:32:16,542 - INFO - PID: 19193
2019-09-18 20:32:16,542 - INFO - backend: epct_gis_eps
```

A user is free to configure a customization process from scratch, or to select and eventually modify and save a configuration from a predefined list. For this reason, the Action button

, which allows to save it, is available in all the Tabs, while the content of the Configuration Panel for each Tab can change.
On the top left of the application a "Burger" menu button allows to access an informative section, where:

- "About" reports on the versions of the EUMETSAT Data Tailor components currently in use,
- "Help" will allow in the future to access to EUMETSAT Data Tailor Help.



By default, the first Tab to show at the application launch is the Launchpad, where the configuration process is set up.

Note: Test data is available in the test-data directory of the EUMETSAT Data Tailor source code.

### 2.1.2 Starting the GUI

To start the GUI, we need a terminal with the epct-desktop conda environment active; conda will be available as a command as it has been already installed (see "Supported platforms and installation"). To activate the environment, execute the following command:

```
conda activate epct-desktop
```

Then execute the following to launch the GUI and open it in the default browser:

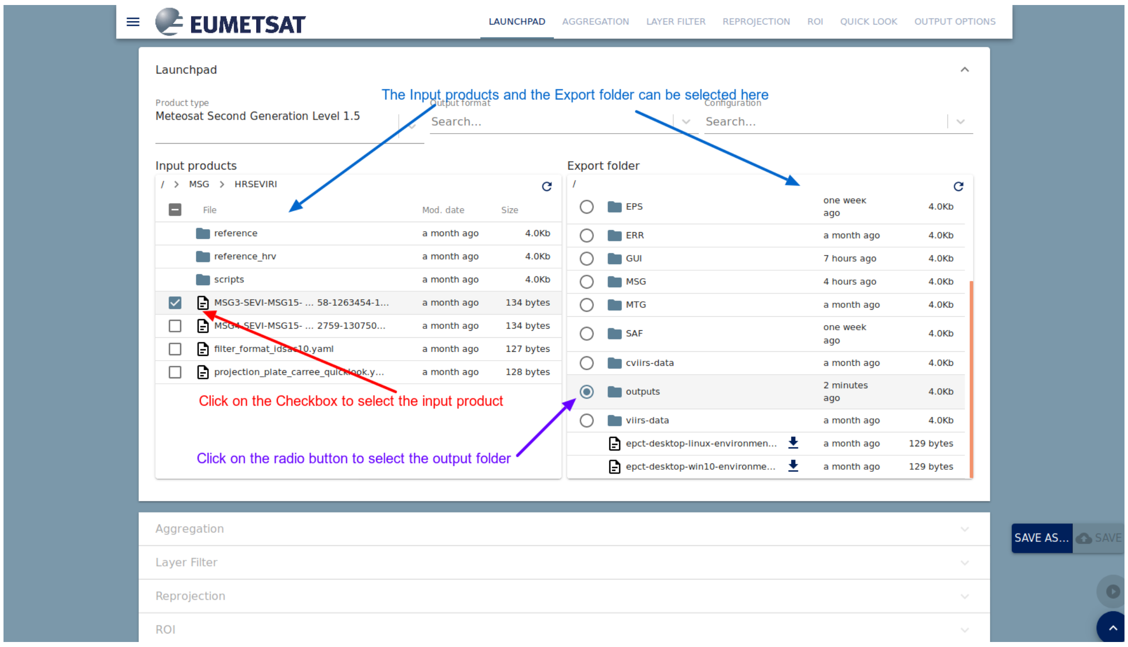
```
epct_webui
```

To change the default ports for the GUI and the REST API, see "Advanced: customising the EUMETSAT Data Tailor installation".

### 2.1.3 Running a simple customization

Let's start with a very simple example to let the user understand the concept of operation. In this example we will see how to extract the channel measurements from a given native format (in this example we will use the High Rate SEVIRI Level 1.5 Image Data, but the main idea is the same for all supported product types) and use this data to create a GeoTIFF.

- In the Launchpad tab, select Meteosat Second Generation Level 1.5 from the Product type drop-down.
- In the Input products panel, navigate to the directory that holds the input data. To do this, you can click on a folder icon to access it, and click on the small "breadcrumbs" (the text below the list of file and directory) to go back. Then select the input file by clicking on the checkbox on the left of its name.



Launchpad

Product type: Meteosat Second Generation Level 1.5
Output format: Search...
Configuration: Search...

The Input products and the Export folder can be selected here

Input products

/ > MSG > HRSEVIRI

| | File | Mod. date | Size |
|---|---|---|---|
| | reference | a month ago | 4.0Kb |
| | reference_hrv | a month ago | 4.0Kb |
| | scripts | a month ago | 4.0Kb |
| ✓ | MSG3-SEVI-MSG15- ... 58-1263454-1... | a month ago | 134 bytes |
| ☐ | MSG4-SEVI-MSG15- ... 2759-130750... | a month ago | 134 bytes |
| ☐ | filter_format_idsac10.yaml | a month ago | 127 bytes |
| ☐ | projection_plate_carree_quicklook.y... | a month ago | 128 bytes |

Click on the Checkbox to select the input product

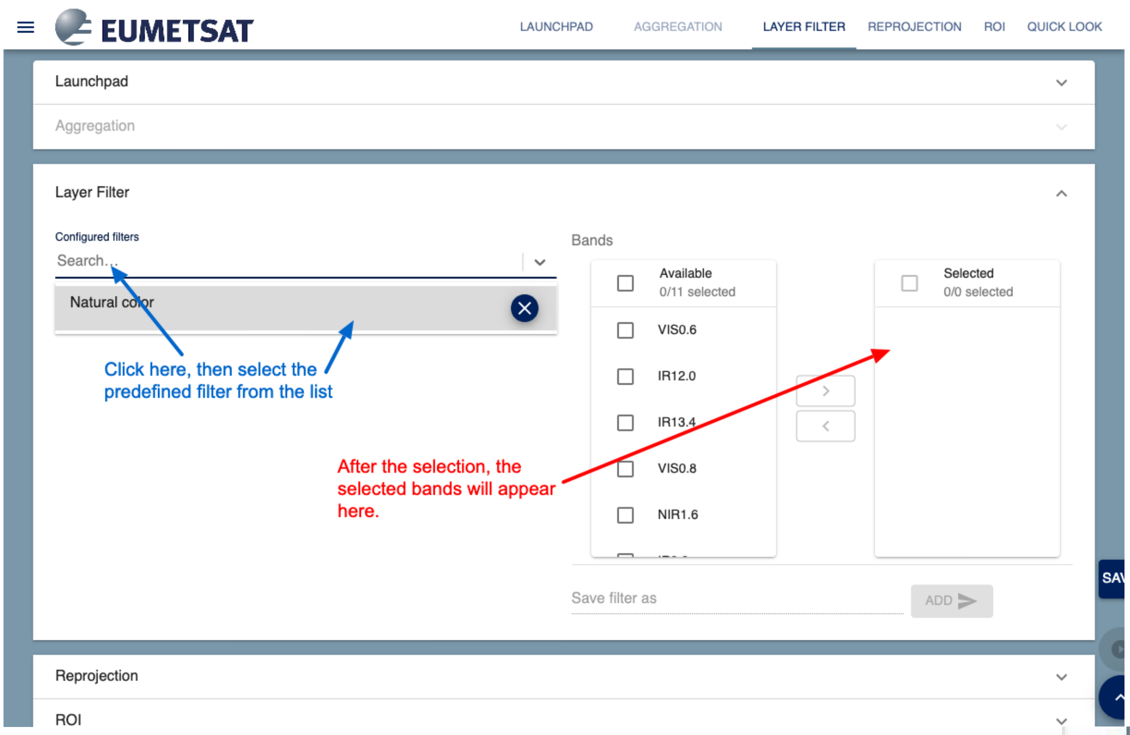
Click on the radio button to select the output folder

Export folder

/

| | | | |
|---|---|---|---|
| ○ | EPS | one week ago | 4.0Kb |
| ○ | ERR | a month ago | 4.0Kb |
| ○ | GUI | 7 hours ago | 4.0Kb |
| ○ | MSG | 4 hours ago | 4.0Kb |
| ○ | MTG | a month ago | 4.0Kb |
| ○ | SAF | one week ago | 4.0Kb |
| ○ | cviirs-data | a month ago | 4.0Kb |
| ● | outputs | 2 minutes ago | 4.0Kb |
| ○ | viirs-data | a month ago | 4.0Kb |
| | epct-desktop-linux-environmen... | a month ago | 129 bytes |
| | epct-desktop-win10-environme... | a month ago | 129 bytes |

- In the Export folder panel, navigate to the directory where the results are to be saved. Then select the desired folder by clicking on the radio button on the left of its name.
- Select the desired output format from the Format drop-down menu (e.g. GeoTiff).
- Leave the Configuration field empty, as we are going to create a new customisation chain
- Select a Layer filter to extract the channels required:
  - Select the Layer filter tab to display the preconfigured filters to extract the product bands for the chosen Product type.
  - Select the chosen filter (Natural color in this example) from the "Configured filters".
  - Note that selecting the filter makes the corresponding bands appear in the "Bands" "Selected" panel.



- Run the Customisation on the selected product by clicking on the Run button on the right side.

After the button is clicked, the Process Monitoring panel appears, reporting on the status of the Customisation process.

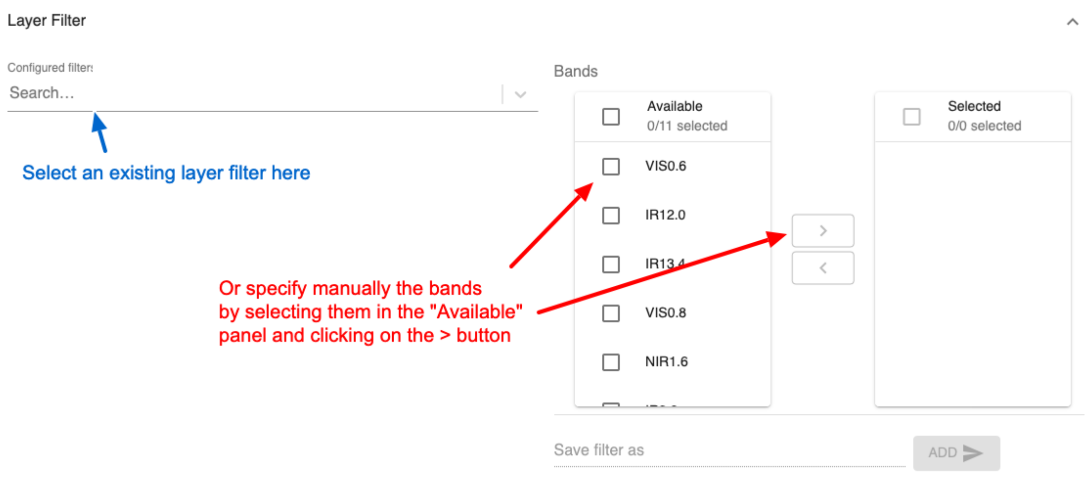
One the status of the Customisation process turns to “Completed”, refresh the list of files in the selected output folder by clicking on the Refresh button beside “Export folder”. The output GeoTIFF file is listed inside the selected folder.

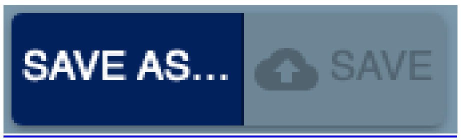
### 2.1.4 Configuring and saving customization configuration

Customisations created within the GUI are known collectively as “Configurations” and these can be saved and reused within the GUI. The following is an example of a procedure to configure and save a new customization configuration with Layer filtering and reformatting.:

- In the Launchpad tab, select Product type, Input products directory, Export folder and the selected output format as described in the previous example.
- To start creating the configuration of a customisation, ensure that the Configuration field is empty.
- Select a Layer filter to extract the channels required:
  - Select the Layer filter tab to display the preconfigured filters to extract the channels
  - for the chosen Product type.
  - Select the chosen filter (e.g.: “Natural color”) from the Configured filters.

***This Document is Public***

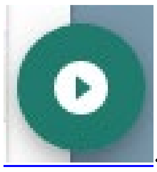

Layer Filter
Configured filter:
Search...
Select an existing layer filter here
Bands
Available
0/11 selected
Selected
0/0 selected
VIS0.6
IR12.0
IR13.4
VIS0.8
NIR1.6
Or specify manually the bands by selecting them in the "Available" panel and clicking on the > button
Save filter as
ADD

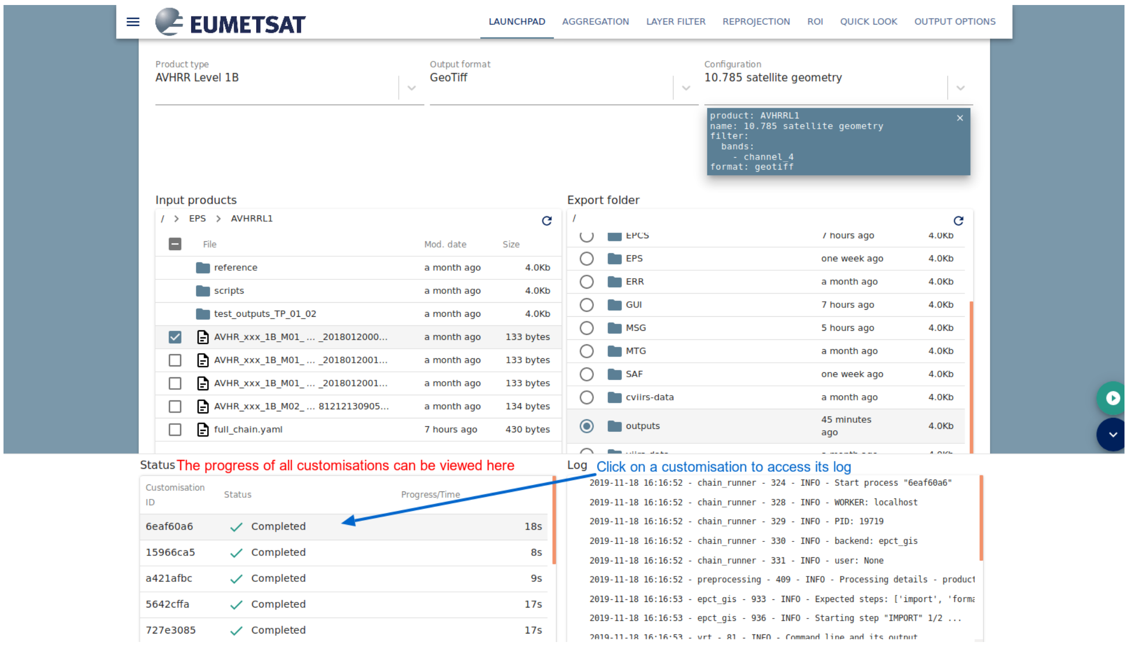
- Save the customization configuration:
  - click on the action button Save as on the right side of the page 

  - write in the pop-up window the name of the new configuration
  - click on the "Save" button to save it.

### 2.1.5 Execute an existing configuration and monitor its status

As mentioned before, users can create new customisation configurations, save them or modify an existing one. In this section we illustrate how to use an existing configuration to run repetitive customizations and how to monitor a process status in more detail (logs).

- In the Launchpad tab, select Product type, Input product and Export folder as described in previous examples.
- Select from the "Configurations" drop down menu an existing configuration. Note that the details of the configuration appear below the menu.
- To run the chosen configuration on the selected product, select the green "Run" button .



Product type: AVHRR Level 1B
Output format: GeoTiff
Configuration: 10.785 satellite geometry

```
product: AVHRRL1
name: 10.785 satellite geometry
filter:
  bands:
    - channel_4
format: geotiff
```

Input products
/ > EPS > AVHRRL1

| | File | Mod. date | Size |
|---|---|---|---|
| | reference | a month ago | 4.0Kb |
| | scripts | a month ago | 4.0Kb |
| | test_outputs_TP_01_02 | a month ago | 4.0Kb |
| ✓ | AVHR_xxx_1B_M01_ … _2018012000… | a month ago | 133 bytes |
| ☐ | AVHR_xxx_1B_M01_ … _2018012001… | a month ago | 133 bytes |
| ☐ | AVHR_xxx_1B_M01_ … _2018012001… | a month ago | 133 bytes |
| ☐ | AVHR_xxx_1B_M02_ … 81212130905… | a month ago | 134 bytes |
| ☐ | full_chain.yaml | 7 hours ago | 430 bytes |

Export folder
/

| | | | |
|---|---|---|---|
| ○ | EPCS | 7 hours ago | 4.0Kb |
| ○ | EPS | one week ago | 4.0Kb |
| ○ | ERR | a month ago | 4.0Kb |
| ○ | GUI | 7 hours ago | 4.0Kb |
| ○ | MSG | 5 hours ago | 4.0Kb |
| ○ | MTG | a month ago | 4.0Kb |
| ○ | SAF | one week ago | 4.0Kb |
| ○ | cviirs-data | a month ago | 4.0Kb |
| ● | outputs | 45 minutes ago | 4.0Kb |

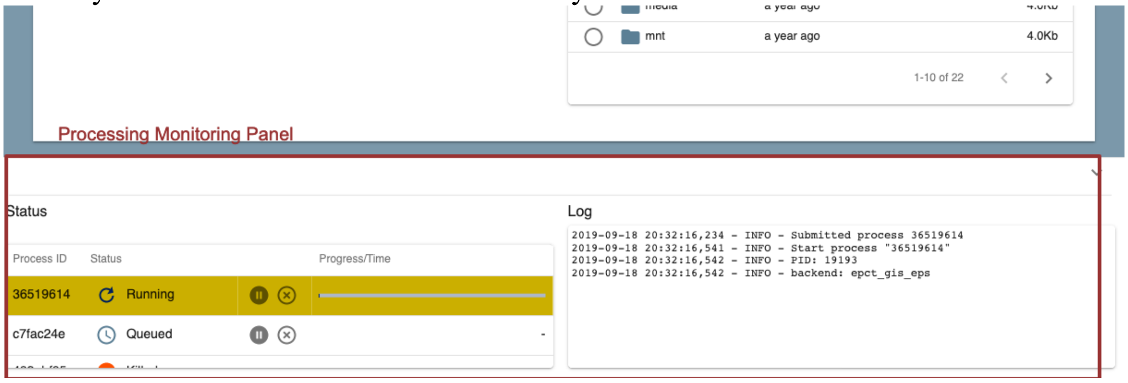
Status The progress of all customisations can be viewed here

| Customisation ID | Status | Progress/Time |
|---|---|---|
| 6eaf60a6 | ✓ Completed | 18s |
| 15966ca5 | ✓ Completed | 8s |
| a421afbc | ✓ Completed | 9s |
| 5642cffa | ✓ Completed | 17s |
| 727e3085 | ✓ Completed | 17s |

Log Click on a customisation to access its log

```
2019-11-18 16:16:52 - chain_runner - 324 - INFO - Start process "6eaf60a6"
2019-11-18 16:16:52 - chain_runner - 328 - INFO - WORKER: localhost
2019-11-18 16:16:52 - chain_runner - 329 - INFO - PID: 19719
2019-11-18 16:16:52 - chain_runner - 330 - INFO - backend: epct_gis
2019-11-18 16:16:52 - chain_runner - 331 - INFO - user: None
2019-11-18 16:16:52 - preprocessing - 409 - INFO - Processing details - product
2019-11-18 16:16:53 - epct_gis - 933 - INFO - Expected steps: ['import', 'forma
2019-11-18 16:16:53 - epct_gis - 936 - INFO - Starting step "IMPORT" 1/2 ...
2019-11-18 16:16:53 - vrt - 81 - INFO - Command line and its output
```

After clicking on the Run button, the Process monitoring panel appears. Its main function is to allow the User to monitor the status of one or more customisation processes. The Panel is available in every tab and it can be hidden or shown by the dedicated Action button in the bottom right.



Status

| Process ID | Status | Progress/Time |
|---|---|---|
| 36519614 | Running | |
| c7fac24e | Queued | - |

Log

```
2019-09-18 20:32:16,234 - INFO - Submitted process 36519614
2019-09-18 20:32:16,541 - INFO - Start process "36519614"
2019-09-18 20:32:16,542 - INFO - PID: 19193
2019-09-18 20:32:16,542 - INFO - backend: epct_gis_eps
```

The Panel is divided into two logical panels:

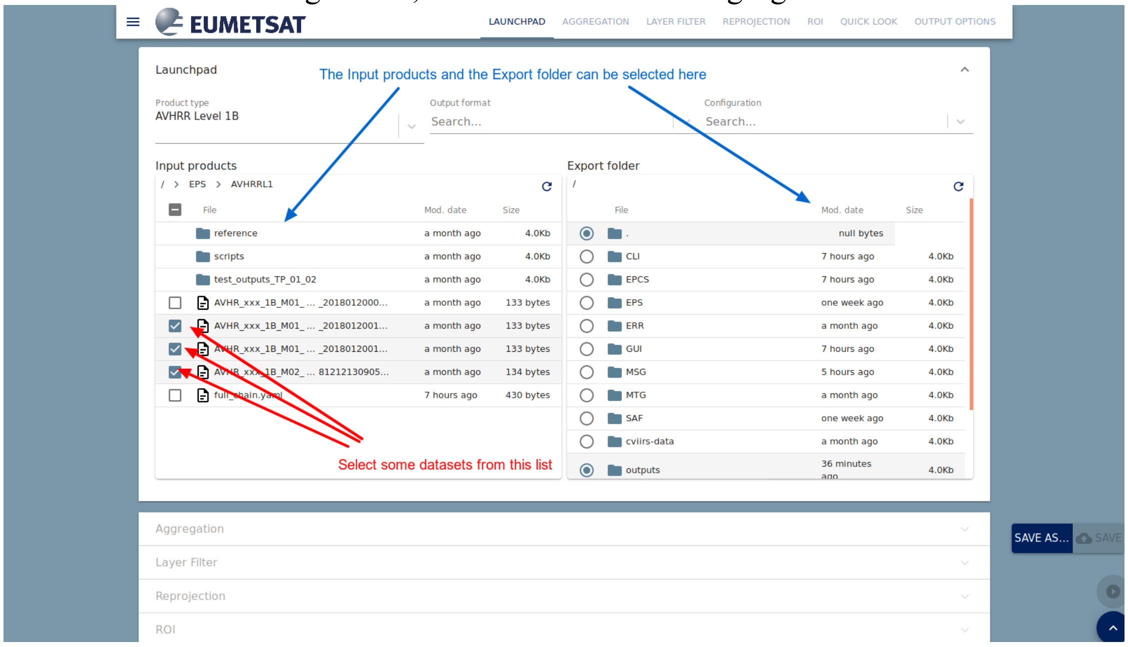
- Status Monitoring Panel: including the general progress bar (Status bar) and a progress bar for each process, along with the related progress. The status of a process can assume the following values:
  - Queued – the process has been submitted and it is waiting the completion of one or more running processes
  - Running – the process is in execution
  - Completed – the process was successfully completed
  - Failed - the process failed due to an error
  - Cancelled - the process has been stopped by the User, by clicking on the “Cancel” button on the process bar.
- Message log: displaying message log for a selected process.

### 2.1.6 Request the execution of multiple processes and manage the results

Finally, we describe how we can run the same configuration on multiple inputs, and how to manage the resulting processes. The GUI can request the execution of any number of processes; they will be queued by the EUMETSAT Data Tailor and run as soon as the processing resources become available.

***This Document is Public***

- Select the "Launchpad" tab, select Product type (AVHRR Level 1B in this example), Input products directory and Export folder as described in previous examples.
- Select GeoTiff as the Output format.
- Select the input products to be customised and leave the "Configuration" field empty to define a new configuration, as show in the following figure:



Launchpad

The Input products and the Export folder can be selected here

Product type: AVHRR Level 1B
Output format: Search...
Configuration: Search...

Input products

/ > EPS > AVHRRL1

| | File | Mod. date | Size |
|---|---|---|---|
| | reference | a month ago | 4.0Kb |
| | scripts | a month ago | 4.0Kb |
| | test_outputs_TP_01_02 | a month ago | 4.0Kb |
| ☐ | AVHR_xxx_1B_M01_ ... _2018012000... | a month ago | 133 bytes |
| ✓ | AVHR_xxx_1B_M01_ ... _2018012001... | a month ago | 133 bytes |
| ✓ | AVHR_xxx_1B_M01_ ... _2018012001... | a month ago | 133 bytes |
| ✓ | AVHR_xxx_1B_M02_ ... 81212130905... | a month ago | 134 bytes |
| ☐ | full_chain.yaml | 7 hours ago | 430 bytes |

Select some datasets from this list

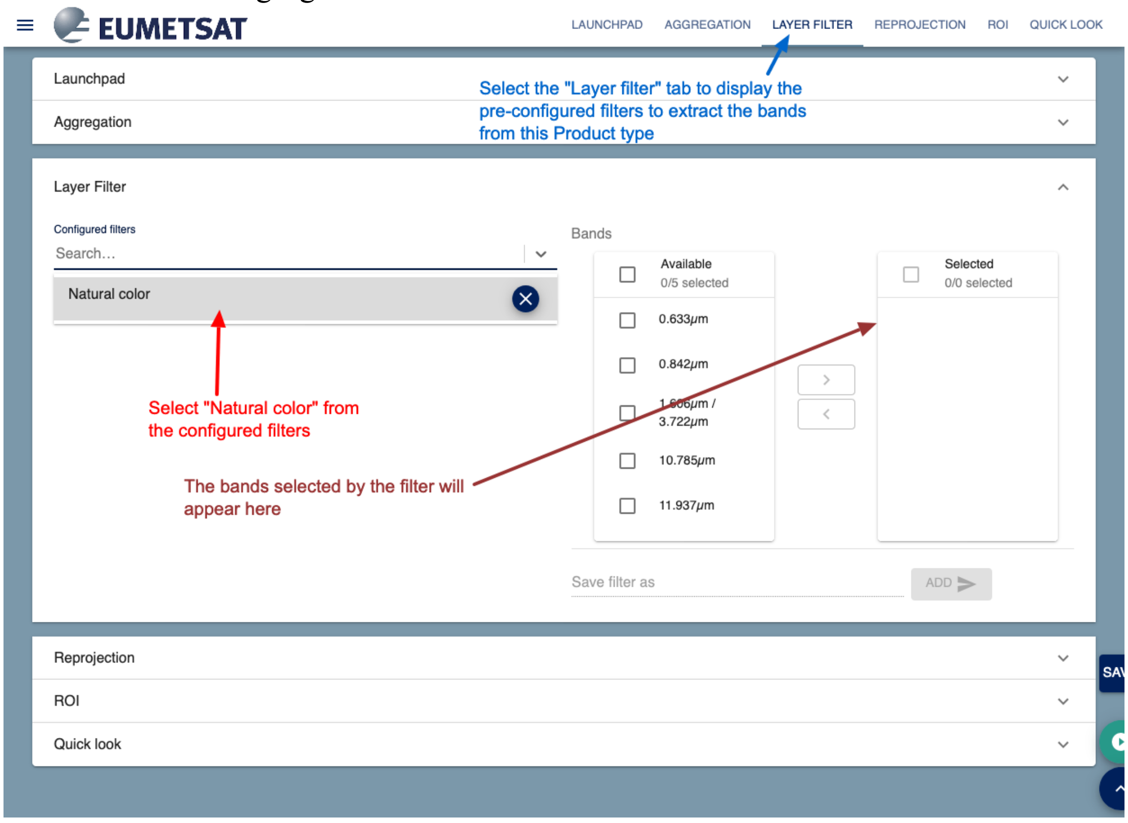
Export folder

/

| | File | Mod. date | Size |
|---|---|---|---|
| ● | . | | null bytes |
| ○ | CLI | 7 hours ago | 4.0Kb |
| ○ | EPCS | 7 hours ago | 4.0Kb |
| ○ | EPS | one week ago | 4.0Kb |
| ○ | ERR | a month ago | 4.0Kb |
| ○ | GUI | 7 hours ago | 4.0Kb |
| ○ | MSG | 5 hours ago | 4.0Kb |
| ○ | MTG | a month ago | 4.0Kb |
| ○ | SAF | one week ago | 4.0Kb |
| ○ | cviirs-data | a month ago | 4.0Kb |
| ● | outputs | 36 minutes ago | 4.0Kb |

Aggregation
Layer Filter
Reprojection
ROI

- Select a Layer filter to extract the channels required for a natural colour PNG (RGB) as shown in the following figure:



Launchpad
Aggregation

Select the "Layer filter" tab to display the pre-configured filters to extract the bands from this Product type

Layer Filter

Configured filters: Search...

Natural color

Select "Natural color" from the configured filters

The bands selected by the filter will appear here

Bands

| Available 0/5 selected | Selected 0/0 selected |
|---|---|
| ☐ 0.633μm | |
| ☐ 0.842μm | |
| ☐ 1.606μm / 3.722μm | |
| ☐ 10.785μm | |
| ☐ 11.937μm | |

Save filter as

Reprojection
ROI
Quick look

- Run the configuration on the selected products, clicking on the green "Run" button on the bottom right.

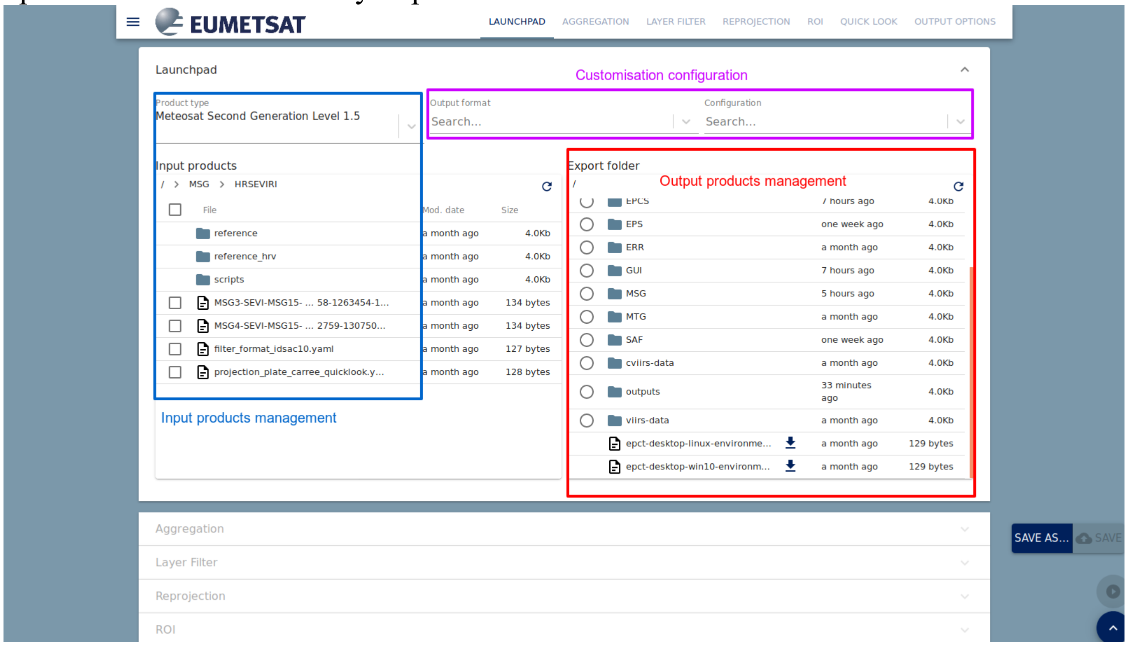
The Process monitoring panel displays a line for each selected products, revealing that one customisation process has been launched for each.
In the panel, the User can monitor the status of the launched processes; clicking on one of them, displays the log for the selected process in the Log panel.
A process can be managed through the buttons that are present on its status bar in the Process monitoring panel:

- Pause : available when the process is running, allows to pause it.
- Play : available when the process is paused, allows to resume its execution
- Cancel : allows to cancel the process.

## 2.1.7 Tab-by-tab description

### 2.1.7.1 Launchpad Tab

The Launchpad has the main function to allow the user to quickly configure and run a customization process. The User Interface has been designed in order to make configuration operations intuitive and easy to perform.



Product type: Meteosat Second Generation Level 1.5
Output format: Search...
Configuration: Search...

Input products
/ > MSG > HRSEVIRI

| File | Mod. date | Size |
|---|---|---|
| reference | a month ago | 4.0Kb |
| reference_hrv | a month ago | 4.0Kb |
| scripts | a month ago | 4.0Kb |
| MSG3-SEVI-MSG15- ... 58-1263454-1... | a month ago | 134 bytes |
| MSG4-SEVI-MSG15- ... 2759-130750... | a month ago | 134 bytes |
| filter_format_idsac10.yaml | a month ago | 127 bytes |
| projection_plate_carree_quicklook.y... | a month ago | 128 bytes |

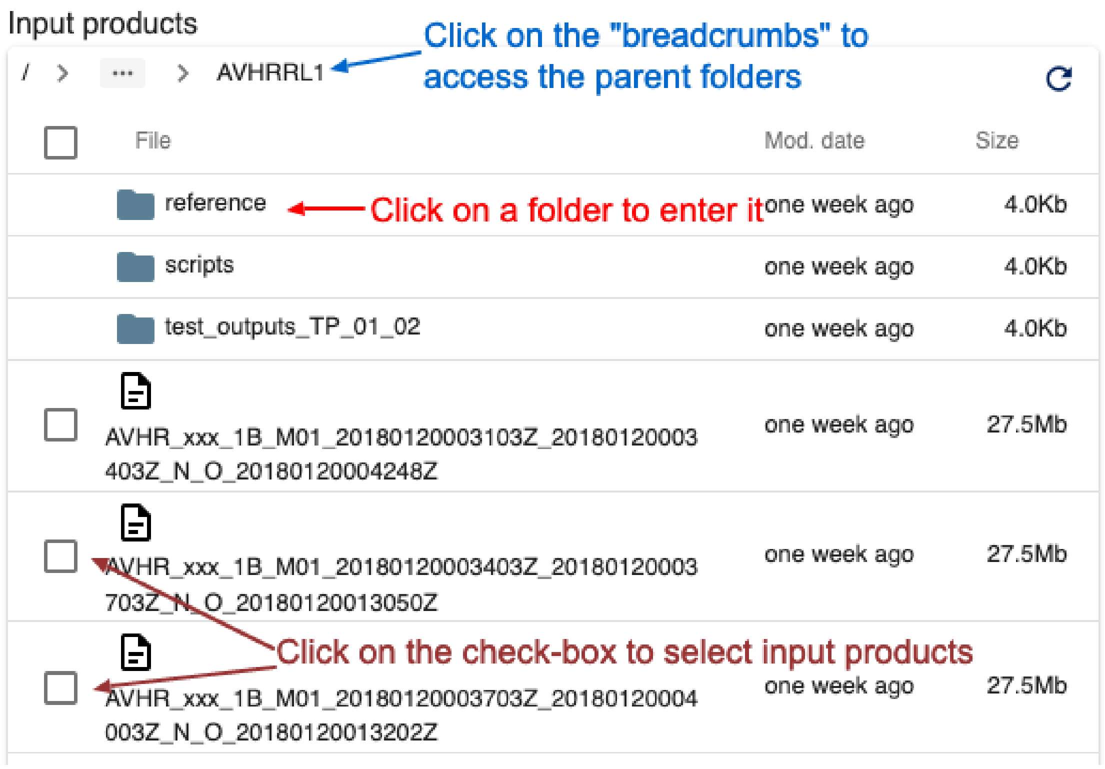
Export folder
/

| | | |
|---|---|---|
| EPCS | 7 hours ago | 4.0Kb |
| EPS | one week ago | 4.0Kb |
| ERR | a month ago | 4.0Kb |
| GUI | 7 hours ago | 4.0Kb |
| MSG | 5 hours ago | 4.0Kb |
| MTG | a month ago | 4.0Kb |
| SAF | one week ago | 4.0Kb |
| cviirs-data | a month ago | 4.0Kb |
| outputs | 33 minutes ago | 4.0Kb |
| viirs-data | a month ago | 4.0Kb |
| epct-desktop-linux-environme... | a month ago | 129 bytes |
| epct-desktop-win10-environm... | a month ago | 129 bytes |

The Launchpad Tab is logically divided into the following panels:

- Input Products: the panel is located on the left of the Configuration Panel and allows the user to select input data for the processing.
  – The Product type widget allows to select the product type from the drop-down menu or search for it by writing a search string in the widget. Note that the Product type is the first data that should be specified, and it is needed to configure any customisation.

  - The Input products allows to browse the folder tree on the local machine and to select the input products to customise. To browse the folder tree, click on the folder to access it; to go back to a parent folder, click on the "breadcrumbs" above the directories. To select input products for customisations, click on the checkbox on the left of their names. To refresh the contents of a directory, click on the "Reload" button on the right of the "Input products".
- Configuration of the customisation: the panel is located in the top-right of the Configuration Panel and allows the user to select the output format or to select, start modifying or delete a customisation configuration.
  - To select the output format, choose it from the Output format drop-down menu. Note that selecting the format modifies which Configurations are available, as only those compatible with the format will be listed. To clear the menu, enter it and delete the format name. Selecting the output format also affects which customisation functions will be available for the customisation.
  - To start configuring a new customisation, leave the Configurations field empty.
  - To select a pre-existing configuration click on the Configurations drop-down menu and select the desired option. Configurations can also be searched by writing the search string in Configurations. Every time a configuration is selected, a full list of the corresponding customization steps will appear below the name. Note that if the output format is not selected yet, choosing a customisation configuration will set it to the one specified by the configuration.
  - To delete a pre-existing configuration, click on the Configurations drop-down menu, then on the "delete" button on the right of its name. A confirmation pop-up will appear., which allows to confirm the deletion.
- Export folder ``: the panel is located on the right of the Configuration Panel and allows the user to select browse the folder tree and select the directory where the outputs of a customisation need to be stored. Browsing works in the same way as the ``Input products panel. To select a folder, click on the radio button on the left of its name; when inside the folder, the folder itself can be selected by click on the radio button close to the folder named as . The content of the panel can be refreshed by clicking on the "Refresh" button; this is especially useful when a customisation process is completed, to see the generated output products.


Input products

/ > ... > AVHRRL1

| | File | Mod. date | Size |
|---|---|---|---|
| | reference | one week ago | 4.0Kb |
| | scripts | one week ago | 4.0Kb |
| | test_outputs_TP_01_02 | one week ago | 4.0Kb |
| ☐ | AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z | one week ago | 27.5Mb |
| ☐ | AVHR_xxx_1B_M01_20180120003403Z_20180120003703Z_N_O_20180120013050Z | one week ago | 27.5Mb |
| ☐ | AVHR_xxx_1B_M01_20180120003703Z_20180120004003Z_N_O_20180120013202Z | one week ago | 27.5Mb |

Click on the "breadcrumbs" to access the parent folders
Click on a folder to enter it
Click on the check-box to select input products

### 2.1.8 Product Customization Tabs

The GUI includes a set of customization Tabs dedicated to the different types of customizations to apply. A customisation tab can be selected by clicking on it at the top of the page, or on the collapsed panel located below the Launchpad page.

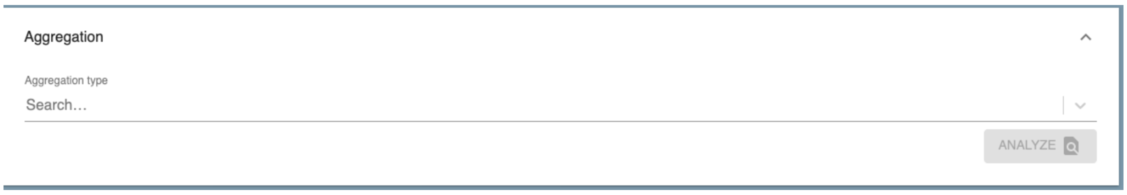
If a tab is configured, the corresponding Customisation function will be applied; so to remove a function from the configuration of a customisation, simply remove the values in the fields of the corresponding tab.

The following Sections report the description of each Tab with the related interactions with the user.

Note: The tabs cannot be accessed unless both the Product type and one in Output format or Configurations are selected.

#### 2.1.8.1 Aggregation

The Aggregation customization function is configurable through the Aggregation tab.

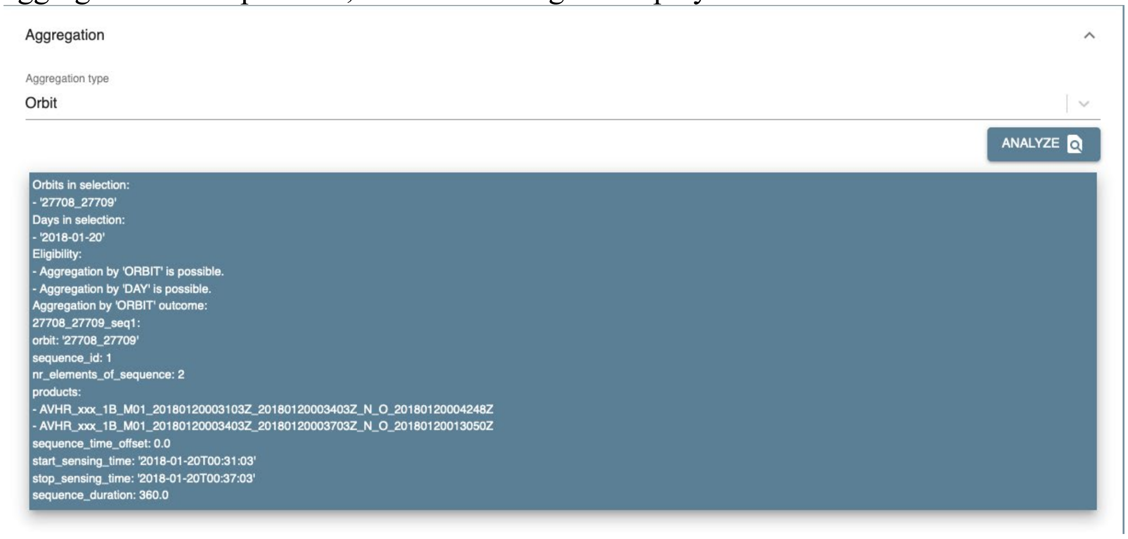


The Aggregation Tab allows to select the Aggregation Type (the availability of each depends on the selected product type), i.e.:

- Orbit: all the input data will be aggregated as 1 output product per orbit;

- Disc: (MSG product only) input data are aggregated as 1 output per disc
- Time: (disc products only): input products corresponding to different times are saved in the same output product (only if the output product is NetCDF-4, HDF5).

Once the Aggregation Type is selected, clicking on the Analyse button reports on the expected output product info: estimated size on the base of the input data, segments to process. If aggregation is not possible, an error message is displayed.



Aggregation type
Orbit

```
Orbits in selection:
- '27708_27709'
Days in selection:
- '2018-01-20'
Eligibility:
- Aggregation by 'ORBIT' is possible.
- Aggregation by 'DAY' is possible.
Aggregation by 'ORBIT' outcome:
27708_27709_seq1:
orbit: '27708_27709'
sequence_id: 1
nr_elements_of_sequence: 2
products:
- AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z
- AVHR_xxx_1B_M01_20180120003403Z_20180120003703Z_N_O_20180120013050Z
sequence_time_offset: 0.0
start_sensing_time: '2018-01-20T00:31:03'
stop_sensing_time: '2018-01-20T00:37:03'
sequence_duration: 360.0
```

#### 2.1.8.2 Layer Filtering

The Layer Filtering customization function can be customised though the Layer filter configuration tab.

The Layer filter Tab is made of two parts:

- Configured filters: this panel includes a list of configured layer filters available for user quick selection. The set of configured filters includes predefined filters and all filters previously created and saved by the user. Filters are displayed in the alphabetical order. To apply one of the configured filters, just click on it, and the selected output bands will be automatically displayed on the Bands - Selected panel. At the same time the remaining (not selected) bands will be displayed on the Bands - Available panel. Only configured filters applicable to input product type will be displayed. It is possible to add or remove a band from a configuration and save the new configuration as a new filter. The panel also allows to remove a pre-configured filter, by clicking on the drop-down menu, then on the Delete button on the right of the filter name.
- **Bands which allows to select manually which bands should appear in the output product.**
  - Available: this panel displays available bands from the input products not currently selected as output bands. To manually select one or more of these bands for the output products, click on them and then click on the (right-ward arrow) button. Drag-and-drop of single bands from Available to Selected is also supported.
  - Selected: the panel displays the selected output bands. To remove a band from the selection, just click on it and then click on the remove from selection (left-ward arrow) button, or drag and drop it to the Available panel. Drag-and-drop also allows to change the order in which the bands are selected.

Once the bands are selected, it is possible to save the filter configuration by:

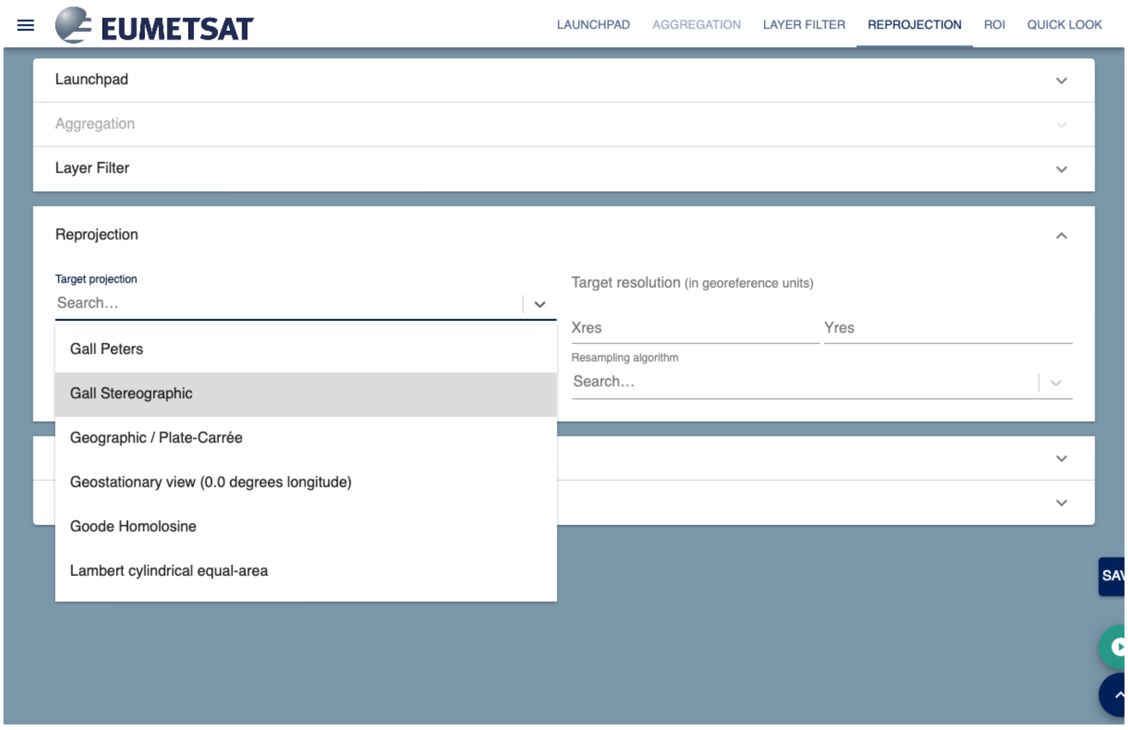
- Writing the filter name in the Save filter as… field located on the bottom right.
- Click on Add button to save it if the filter is a new one; the button turns to Update if the filter already exists.

#### 2.1.8.3 Reprojection and Resampling

The Reprojection and the Resampling customization functions is configured in the Reprojection configuration tab.

The Reprojection Tab is made of two parts:

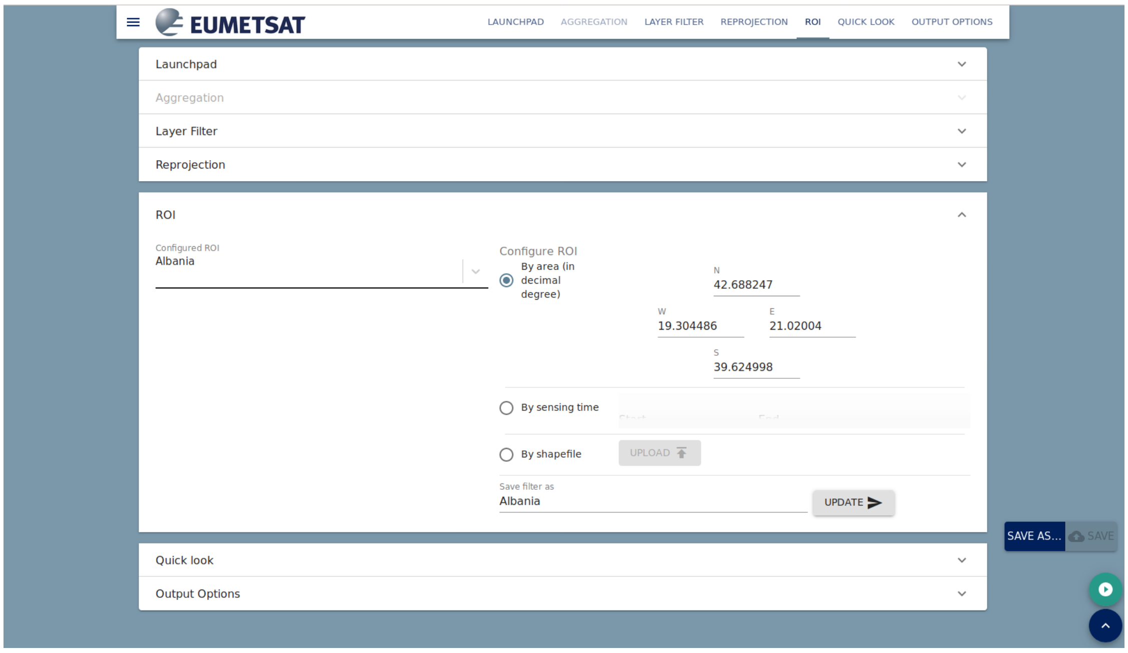
- Target projections, on the left, allows to select the target projection from a predefined list. To select a projection, just click on it; the related projection info will be displayed below.
- **the configuration of the resampling on the right, with the following widgets:**
  - Target resolution includes two fields for the specification of the target resolution: Xres (X resolution, or width resolution) and Yres (Y resolution, or height resolution). The user will specify two numerical values, in the units specified in the legend (meters or degrees); note that the units change according to the chosen projection.
  - Resampling algorithms: allows to select one of the available resampling algorithms. Just click on it to select it.



Note that reprojection must be configured to apply resampling.

#### 2.1.8.4 ROI

The ROI customization function can be configured in the tab ROI.



ROI

Configured ROI: Albania

Configure ROI
- By area (in decimal degree) [✓]
  - N: 42.688247
  - W: 19.304486
  - E: 21.02004
  - S: 39.624998
- By sensing time
- By shapefile

Save filter as: Albania

The ROI Tab is made of two parts:

- Configured ROI on the left, allows to select one of the configured ROIs. This list is independent on the input product type. The set of configured ROIs includes pre-defined ROIs and all the ROIs previously created and saved by the user. The ROI can be selected from the drop-down menu or can be searched by writing the search string in the field. To apply one of the configured ROIs, just click on it, and the ROI info will be automatically displayed in the Configure ROI part.
- **Configure ROI on the right allows to specify manually the ROI. A ROI can be specified:**
  - By area, i.e. as a geographical bounding box. The user will insert latitude values in the N (North) /S (South) fields and longitude values in the E (East)/W(West) fields. Latitude and longitude values must be expressed into decimal degrees.
  - By sensing time: ROI is specified by a time interval (which should be included in the start and stop sensing time of the data set). Start and Stop sensing times must be expressed in the ISO-8601 date time format (YYYYMMDDThhmmssZ).
  - By shapefile, i.e. through an ESRI Shapefile defining a polygon. The Shapefile must be a ZIP archive, containing at least the .shp .shx .dbf .prj files

  To select one of such methods, click on the radio button on the left of the method name.

A ROI defined by area can be saved by:

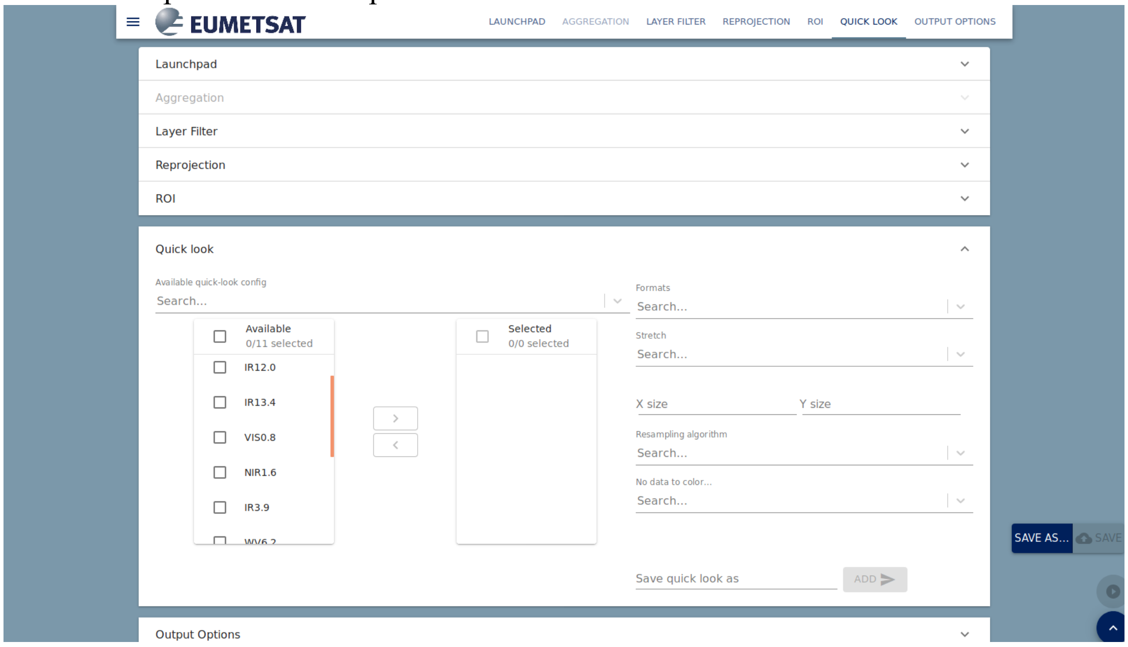
- Writing the ROI name in the Save filter as… field located on the bottom right under the Configure ROI;
- Click on “Add” (for a new ROI) or “Update” (for an existing ROI) button to save it (the button name changes automatically depending if the ROI name exists or not). In case of an update, a pop-up will appear to request confirmation before overwriting the existing ROI.

### 2.1.8.5 Quick-look

The Quicklook customization functions is configured in the Quicklook configuration tab.

The Quick-look tab includes the following configuration panels:

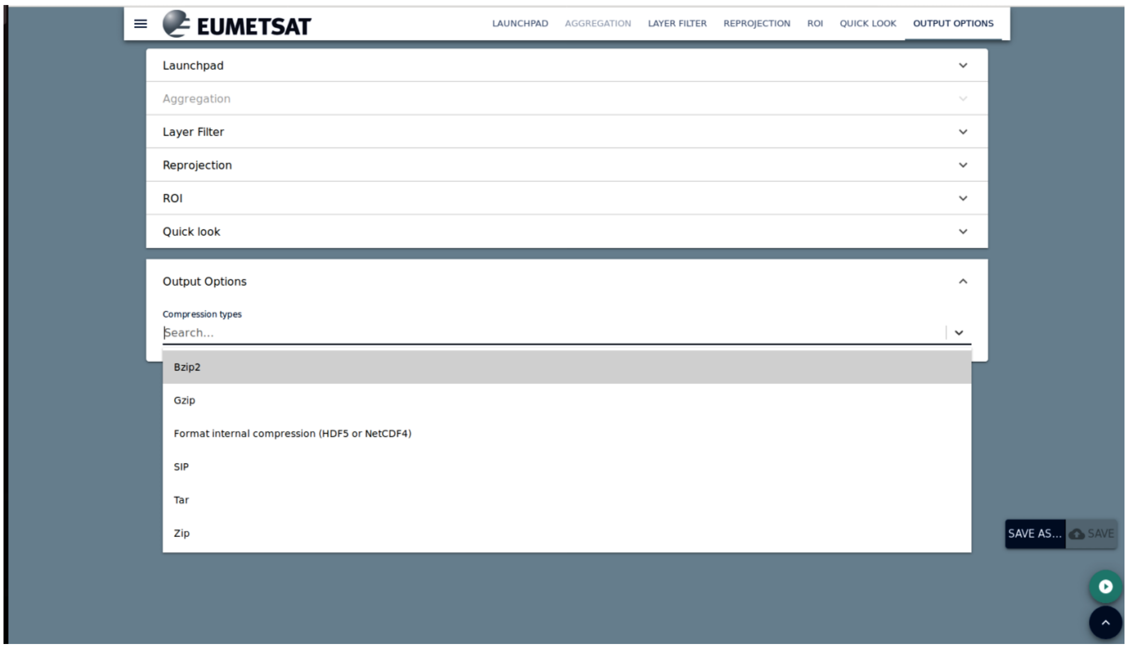
- Available quick-look config on the top-left: allows to select one of the QuickLook configurations available for the product type selected in the Launchpad. Selecting a Quicklook populates the other panels accordingly.
- Available and Selected bands panel: allows to manage the output bands for the quick-look product. The selection of bands works in the same way as in the Layer Filter tab. Note that the available bands depend on which bands (if any) have been selected in the Layer Filter tab.
- **the widgets allowing the configuration of the output file:**
  - the Formats drop-down menu allows users to select the file format;
  - the Stretch drop-down allows to select the stretching method to improve contrast and brightness;
  - the Xsize and Ysize fields allow to specify the size of the output in pixel. Note that:
    - if only one of the two is specified, the other will be automatically computed to keep the image size ratio constant.
    - if both are specified, the output format will be rescaled accordingly, distorting the image if needed.
  - the Resampling algorithms drop-down allows to select the desired resampling algorithm.
  - the No data to color allows to select which colour will be used to render "No Data" pixels in the output file.



### 2.1.8.6 Output options

Further options to specify the output can be provided in the `Output options` configuration tab.

The tab allows to select an optional compression format for the customisation results; products and quicklooks are compressed in a single archive file.



The following compression formats are available:

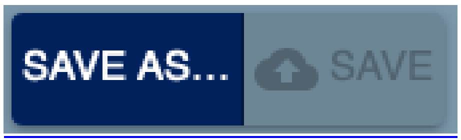
- `Bzip2`: The archive file compressed using the bzip2 algorithm; the output is given the .tar.bz2 extension
- `Gzip`: The archive file compressed using gzip algorithm; the output is given the .tar.gz extension
- `internal`: internal compression for HDF5 or NetCDF4 formats using shuffle filter and gzip compression (compression level 9). The output extension is preserved.
- `SIP`: The archive file is in the SIP format (Submission Information Package, ref. EUM/TSS/TEN/14/761451),
- compressed with zip; the output is given the .zip extension

- `Zip`: The archive file compressed using the zip algorithm; the output is given the .zip extension.

### 2.1.9 Action buttons

Action buttons are always available on the right border of the page. Buttons become more visible by moving the mouse pointer on them.

The following action buttons are available:

- Save/Save as button 

allow to save the current customisation configuration. Save will create or overwrite a new customisation configuration (if overwriting, a pop-up will ask for confirmation). Save as will save the configuration with a new name. In both cases, a pop-up will allow to specify the configuration name and a description. Note that the button are greyed out until the minimum required configuration (input product type and output format) is specified.

- Run Process button : allows to request the execution of a customisation process. The button is greyed out until the minimum required configuration (input product type, input product, export folder and output format) is specified. Clicking on the button also opens the Process Monitoring panel.
- Process Monitoring button : shows the Process Monitoring panel.

## 2.2 Using the EUMETSAT Data Tailor CLI

### 2.2.1 The CLI: a description

The CLI can be used to execute EUMETSAT Data Tailor functionalities from the command line or inside a shell script; it is installed by default by the EUMETSAT Data Tailor installation package. The CLI allows to:

- request the execution of a product customisation;
- manage saved customisation configurations;
- manage saved layer filters, ROI and quick-look configurations;
- request information on the possible outcome of an aggregation;
- manage a launched process
- explore the EUMETSAT Data Tailor configuration and retrieve the current capabilities in terms of:
  - supported input product types
  - available output formats for each product type
  - available customisation functions for a given product type and output format.
- explore the customisation logs.

The CLI commands have the following syntax:

```
epct <sub-command> [options] [parameters]
```

Depending on the sub-command, the options (e.g. the customisation configuration) and parameters (e.g. the paths to the input products) may be optional.
As an example, the request to execute a customisation described in the `test-customisation.yaml` file on the `AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z` product and to place the output files in the current directory is written as follows:

```
epct run-chain \
    -f test-customisation.yaml \
    AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z \
    --target-dir .
```

The GNU program argument syntax conventions is used to specify optional arguments (e.g.: short form: `-h` , long form: `--help`).

Both the entry point to the interface (`epct`) and the sub-commands accept the option `-h` (`--help`) to show the description of the command, and the available options.

The CLI accepts as inputs the paths to the input products or the URLs to the input products in OLDA. In the latter case, users must also provide their authentication token.

The available sub-commands are provided in the following table:

| **Category** | **Sub-command** | **Description** |
|---|---|---|
| Customisation | `run-chain` | Run a customisation with a given configuration on the specified input files. |
| | `customisations` | Show the list of the customisations and their status. |
| | `log` | Print log information for a given customisation process |
| | `manage-process` | Manage a running process to see its status or to send it a signal. |
| Customisation configuration management | `read` | Show available options for a given product type. |
| | `delete` | Delete user configurations for the following SECTIONs: “chain”, “filter”, “roi” and “quicklook”. |
| | `save` | Save user configurations for the following SECTIONs: “chain”, “filter”, “roi” and “quicklook”. |
| Data Tailor configuration | `config` | Show the general configurations. It also allows to reset the EUMETSAT Data Tailor configuration. |
| | `info` | Return general information about EUMETSAT Data Tailor (version, installed components, defaults). |
| Miscellaneous | `aggregation-analyse` | Show the result of the aggregation analysis for a given product and for the chosen input files. |

Note: Test data is available in the test-data directory of the EUMETSAT Data Tailor source code.

### 2.2.2 Accessing the CLI

To use the CLI in a terminal, the epct-desktop conda environment must be activated first; conda will be available as a command as it has been already installed (Supported platforms and installation).

To activate the environment, execute the following command:

```
conda activate epct-desktop
```

### 2.2.3 Running a simple customisation

In this section we will provide examples to show how run a customization configuration via CLI.

**Example 1.** Customise an AVHRRL1 product as described in an existing configuration for AVHRRL1 products named format-geotiff:

```
epct run-chain \
    -c format-geotiff \
    -o <target_dir> \
    AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z
```

**Example 2.** Customise an AVHRRL1 product as described in an existing configuration for AVHRRL1 products as specified in the YAML configuration file format-geotiff.yaml:

```
epct run-chain \
    -f format-geotiff.yaml \
    -o <target_dir> \
    AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z
```

**Example 3.** Specify inline the customisation configuration for an AVHRRL1 product and run it:

```
echo -e "product: AVHRRL1\nfilter: natural-color\nformat:
geotiff\nprojection: geographic" \
    | epct chain-run  -y \
    -o <my_output_dir> \
<input_dir>/AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_2018012000424
8Z
```

**Example 4.** Customise a MSG HRSEVIRI L1.5 product as described in an existing configuration specified in the YAML configuration file format-geotiff.yaml, which only reformats to GeoTIFF, and extracting the ROI specified in the Shapefile central_germany.zip:

```
epct run-chain \
    -f format-geotiff.yaml \
    -s central_germany.zip \
    -o <target_dir> \
    MSG3-SEVI-MSG15-0100-NA-20160701121240.370000000Z-20160701121258-1263454-
1.nat
```

### 2.2.4 Explore and manage the EUMETSAT Data Tailor configuration through the CLI

This section describes how to explore and manage the EUMETSAT Data Tailor configuration through the CLI.

If you are not familiar with such configuration, the Appendix "Understanding the EUMETSAT Data Tailor configuration" provides a useful overview.

#### 2.2.4.1 Reading the configuration via CLI

The key sub-command to retrieve available configuration is `read`. This command allows to query the general configuration and/or product configurations.

##### 2.2.4.1.1 Retrieving product types

To see all product types, execute the following command:

```
epct read products
```

The output will be something like:

```
AMSAL1:
  bands:
    channel_1:
      name: 28.8GH
      number: 1
    channel_10:
      name: 57.290344+-0.217GHz
      number: 10
    ...
  id: AMSAL1
  name: AMSU-A Level 1B
ASCATL1SZF:
  bands:
      ...
  id: ASCATL1SZF
  name: ASCAT-1B SZF
  ...
```

#### 2.2.4.1.2 Retrieving ROIs and other product-independent configuration

To see all the pre-configured ROIs execute the following:

```
epct read rois
```

The output will be something like this:

```
albania:
    NSWE:
    - 42.688247
    - 39.624998
    - 19.304486
    - 21.02004
    id: albania
    name: Albania
antarctic:
    NSWE:
    - -60
    - -90
    - -180
    - 180
    id: antarctic
    name: Antarctic
    ...
```

For each general configuration the command shows the pretty name for the user (GUI visualizes only these names) and the internal key-names (`id`) used inside the configuration yaml files.
The same syntax can be used to retrieve product independent configuration for "sections" other than `rois`. To see available sections, execute `epct read`.

#### 2.2.4.1.3 Retrieving product-dependant information

Using the option `-p` followed by the product type the sub-command shows all the available configurations for a given product type, followed by a configuration section. For example to see supported output formats for the product type `HRSEVIRI`, execute:

```
epct read -p  HRSEVIRI formats
```

To show the contents of the natural_color_disc predefined configuration for HRSEVIRI:

```
epct read -p HRSEVIRI chains/natural_color_disc
```

The output will be:

```
filter: hrseviri_natural_color
format: geotiff
id: natural_color_disc
name: Natural color disc
product: HRSEVIRI
```

To expand all configuration for a given session and get more details just use the option `-x`:

```
epct read -p HRSEVIRI -x chains/natural_color_disc
```

The output will be:

```
filter:
  bands:
  - id: channel_3
    name: NIR1.6
    number: 3
  - id: channel_2
    name: VIS0.8
    number: 2
  - id: channel_1
    name: VIS0.6
    number: 1
  id: hrseviri_natural_color
  name: Natural color
  product: HRSEVIRI
format:
  description:  GeoTIFF  is  a  public  domain  metadata  standard  which  allows
georeferencing
    information to be embedded within a TIFF file.
  ext: .tif
  id: geotiff
  name: GeoTiff
id: natural_color_disc
name: Natural color disc
product: HRSEVIRI
source_text: 'product: HRSEVIRI
  name: Natural color disc
  filter: hrseviri_natural_color
  format: geotiff'
```

### 2.2.4.2 Save a piece of configuration

Users can save a piece of configuration using the CLI `epct save`. This sub-command saves a new configuration for the following sections: `chain`, `filter`, `roi` and `quicklook`. Users can pass the configuration as standard input or as path to the YAML file using the `-f` option.
For example to save a new ROI `west_africa` passing the configuration as standard input, the command line will be something like that (on Ubuntu OS):

```
echo -e "name: Test Africa\nNSWE: [37, 2, -19, 21]" \
| epct save roi
```

The output will be:

```
('test_africa', {'name': 'Test Africa', 'NSWE': [37, 2, -19, 21]})
```

To save the new ROI using a YAML file instead passing the configuration as standard input, the user can use the option `-f`:

```
epct save roi -f test_africa.yaml
```

#### 2.2.4.3 Deleting a piece of configuration

To remove an existing piece of configuration, the CLI `epct delete` is used, e.g. to remove the ROI saved in the previous paragraph:

```
epct delete roi test_africa
```

### 2.2.5 Retrieving general information on the EUMETSAT Data Tailor installation

To retrieve the information on the current EUMETSAT Data Tailor installation, the info sub-command is used. Executing:

```
epct info
```

Returns an output like:

```
epct_version: 2.4.0
etc_dir: ...
workspace_dir: ...
customisations_log_dir: ...
installed_plugins:
  - epct-plugin-umarf, v0.1
  - epct-plugin-gis, v0.1
  - epct-plugin-fist-iasil1c, v0.1
registered_backends:
  - epct_gis
  - epct_gis_eps
  - epct_gis_eps_grib2
  - epct_gis_glbsst
  - epct_gis_glbsst_grib2
  - epct_gis_grib2
  - epct_gis_hrit
  - epct_gis_hrit_grib2
  - epct_plugin_fist_iasil1c
  - epct_plugin_umarf_isccp
  - epct_plugin_umarf_xrit
```

where:

- etc_dir is the directory where the EUMETSAT Data Tailor configuration is stored
- workspace_dir is the directory where any intermediate product is stored. The default path can be customised by editing the epct.yaml file in directory listed in etc_dir
- customisations_log_dir is the directory where the logs for the customisation processes are stored
- installed_plugins and registered_backends list the installed customisation plugins and backends, which together determine the actual customisation capabilities of the EUMETSAT Data Tailor.

### 2.2.6 Integrating the CLI in a shell script

#### 2.2.6.1 A simple example: configuring and executing a customisation in a script

The following script provides a simple example on how to configure the customisation and executing it.

```
# Native HRSEVIRI files are processed using
# a customisation defined in the script, and the results placed
# in a predefined output directory
# Call signature:
# ./sample-customise.sh $INPUT_FILE

input_file="$1"

output_folder="test-output"

# Run a customisation, passing the configuration as the standard input
epct run-chain -y -o "$output_folder" "$input_file" <<EOF
  filter: hrseviri_natural_color
  format: geotiff
  id: projection_plate_carree_quicklook
  name: Projection Plate-Carree with quick-look
  product: HRSEVIRI
  projection: geographic
  quicklook:
    filter:
      bands:
      - channel_3
      - channel_2
      - channel_1
    format: png_rgb
    nodatacolor: transparent
    resample_method: cubic
    stretch_method: min_max
  roi: western_europe
EOF
```

#### 2.2.6.2 A more advanced example: integrate the EUMETSAT Data Tailor CLI and EBPA

An example of integration between *EUMETSAT Data Tailor CLI* and *E Batch Processing Agents (EBPA) – Prototype* in a shell script is described in Exercise 5 of EUMETSAT Product Customisation Toolbox - Installation Guide & Exercises.

#### 2.2.6.3 Another advanced example: Configuring the EUMETSAT Data Tailor for a EUMETCast Station (CLI)

An example on how to use EUMETSAT Data Tailor on a EUMETCAST Station using is provided in Exercise 4 of EUMETSAT Product Customisation Toolbox - Installation Guide & Exercises (Tables 1 and 2). The scripts simulate-customise.sh (Table 2 in the exercise) needs to be adapted as follows:

```
# Every second, process the first MSG native file found in input directory.
#
# Output files replace input files. Native files are processed using
# natural-color-disc customisation chain for MSG HRSEVIRI products
# Call signature:
# ./simulate-customise.sh $INPUT_FOLDER

input_folder="$1"
# Indefinite loop
while true; do
    # Placed here so that script doesn't idle indefinitely without input
    sleep 1s
      # Look up first MSG native file found
    input_file=$(ls $input_folder/*nat 2>/dev/null | head -n 1) # If no file
is found, return to beginning
    [[ -z $input_file ]] && continue
    # Execute chain. If successful, delete input file.
    epct run-chain -c natural-color-disc  -o "$input_folder" "$input_file" \
    && rm "$input_file"
done
```

### 2.2.7 Short review of the CLI sub-commands

Other useful sub-commands are listed below:

- `epct config`: it shows general configurations. Optionally, it is possible to reset the configuration. Available options:
  - `--force-etc-reset` Dangerous. Clear configuration folder and restore defaults It will also create a backup of the previous configuration in the configuration folder.
  - `--help` Show the help message and exit.
- `epct customisations`: it returns the list of all customisations. Optionally, it is possible to get information on a specific customisation.
  Main options:
  - `-u`, `--process-uuid TEXT` a valid customisation process UUID
  - `-t`, `--today` only show the customisations run today
  - `-s`, `--status [queued\|running\|done\|suspended\|killed\|failed\|inactive]`
  - only show the customisations in the specified status
  - `--start_time`, `--st TEXT` only show customisations started after this time
  - (time must be expressed as `YYYYMMDDThhmmssZ`)
  - `--stop_time`, `--sp TEXT` only show customisations started before this time
  - (time must be expressed as `YYYYMMDDThhmmssZ`)
- `epct log`: Display the log for a process with `PROCESS_UUID UUID`.
  Available options:
  - `--last_lines`, `--ll INTEGER` Number of lines must to be read starting from the end
- `epct manage-process`: Manage a running process to see its status or to send it a signal (`kill`, `suspend`, `resume`). Available options:
  - `-i`, `--pid` Desired PID.
  - `-n`, `--process_id` A EUMETSAT Data Tailor process-ID.
  - `-s`, `--signal` Available actions: kill, suspend, resume or status.

  - `--help` Show the help message and exit.
- `epct aggregation-analyse`: Show the result of the aggregation analysis for a given product and for the chosen input files. Available options:
  - `-p`, `--product` Desired product key (e.g.: AVHRRL1).
  - `-b`, `--band` Desired band ID.
  - `-a`, `--aggregate-by` Aggregation types (e.g.: orbit, time, disk).
  - `--help` Show the help message and exit.

## 2.3 Using the EUMETSAT Data Tailor API

### 2.3.1 The EUMETSAT Data Tailor API: a description of the key entry points

The EUMETSAT Data Tailor provides an Application Programming Interface to make its functionalities available as Python function to external Python programs. Other programming languages are not supported at the moment.
The API allows to:

- request the execution of a product customisation;
- manage saved customisation configurations;
- manage saved layer filters, ROI and quick-look configurations;
- request information on the possible outcome of an aggregation;
- manage a launched process
- explore the EUMETSAT Data Tailor configuration and retrieve the current capabilities in terms of:
  - supported input product types
  - available output formats for each product type
  - available customisation functions for a given product type and output format.
- explore the customisation logs.

The API is implemented by the `api` module.

As an example, to request the execution of the format conversion to NetCDF4 for the `AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_20180120004248Z` product in the current directory and to place the output files in the existing `results` directory, the Python script would contain the following lines:

```
from epct import api

INPUT_FILENAME = 'AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_2018012
0004248Z'
chain_config = {'product': 'AVHRRL1', 'format': 'netcdf4'}

output_files =  api.run_chain(
    [INPUT_FILENAME],
    chain_config=chain_config,
    target_dir='results'
)
```

And the `output_files` list would contain the paths to the output files (only one in the example).
The main entry points to the Python API are provided in the following table:

| **Category** | **Entry Point** | **Description** |
|---|---|---|
| Customisation | `run_chain` | Run a customisation with a given configuration, specified as a dictionary on the specified input files. Returns the list of the paths to the generated products. |
| | `run_chain_to_xarray` | Run a customisation with a given configuration, specified as a dictionary on the specified input files. Returns an *xarray* dataset. |
| | `submit_customisations` | Request the asynchronous execution of the same customisation on a set of products of the same type. Returns a dictionary with the Customisation UUIDs as the keys and the Customisation task objects as the values. |
| | `customisations` | Returns information on current and past customisations as a dictionary with the Customisation UUIDs as the keys and the information as the value. |
| | `log` | Returns log information for the customisation with the specified UUID as a multi-line string. |
| | `manage-process` | Manage a running process to see its status or to send it a signal. |
| Customisation configuration management | `read` | Returns a dictionary with the available configuration options for a given product type. |
| | `delete` | Delete a user configuration with a given ID for the following SECTIONs: “chain”, “filter”, “roi” and “quicklook”. It returns the removed ID if successful |
| | `save` | Save user configurations for the following sections: “chain”, “filter”, “roi” and “quicklook”. |
| Data Tailor configuration | `config` | Returns the general configuration as it appear on disk, as a dictionary It also allows to reset the EUMETSAT Data Tailor configuration. |
| | `info` | Return general information about EUMETSAT Data Tailor (version, installed components, defaults) as a dictionary |
| Miscellaneous | `aggregation_analyse` | Show the result of the aggregation analysis for a given product and for the chosen input files, as a serialised dictionary (stream) |

Note: Test data is available in the test-data directory of the EUMETSAT Data Tailor source code.

### 2.3.2 Accessing the API

To use the API in a terminal, the `epct-desktop` `conda` environment must be activated first; conda will be available as a command as it has been already installed (see "Supported platforms and installation").

To activate the environment, execute the following command:

```
conda activate epct-desktop
```

### 2.3.3 Running a simple customisation

In this section we will see hot to run a configuration chain using the API `run_chain`. The signature of this Python function is:

```
def run_chain(
    product_paths,
    product=None,
    chain_name=None,
    chain_config=None,
    config={},
    target_dir='.',
    workspace_dir=etc.WORKSPACE_DIR,
    sensing_start=None,
    sensing_stop=None,
    olda_identifier=None,
    check_olda_cache=None,
    dry_run=False
):
```

This function executes a customisation according to a configuration specified as a dictionary or a key-name on a set of products and return the list of the paths to the generated products.

Note that the paths to the input products could also be URL pointing to the products in OLDA (GLBSST only at the moment).

In the following, a few examples to be executed within from a Python interactive shell are described. >>> and ... are the shell prompts and does not have to be copied. Strings in <...> are placeholders, to be replaced by the user.

#### 2.3.3.1 Example: Run a customisation involving several customisation functions, including aggregation

This example shows how to run a customisation involving the ROI Extraction, Layer Filter, Re-projection and Aggregation and Format Conversion to NetCDF-4 functions, with the configuration specified directly as a dictionary.

```
>>> from epct import api

# define the configuration of the functional chain to apply:
>>> chain_config = {
...         'name': 'sample_chain',
...         'product': 'AMSAL1',
...         'format': 'netcdf4',
...         'roi': {'NSWE': [62, 23, -15, 10]},
...         'projection': 'geographic',
...         'aggregation': 'orbit',
...         'filter': 'terrain_elevation'
...     }

# specify the input product paths and run the chain
>>> input_products = [
...     '<my_input_dir>/AMSA_xxx_1B_M01_20180319100122Z_20180319100426Z_N_O_2
0180319105613Z',
...     '<my_input_dir>/AMSA_xxx_1B_M01_20180319100426Z_20180319100722Z_N_O_2
0180319105956Z',
... ]

# run the chain
>>> output_files = api.run_chain(
...     product_paths=input_products,
...     chain_config=chain_config,
...     target_dir='<my_output_dir>'
... )

# get the output file paths
>>> output_files
```

Note that the `output_files` list contains only one path, as we have aggregated the two products.

#### 2.3.3.2 Example: Run a functional chain described by an existing configuration specified in a YAML file

This example shows how to customise an AVHRRL1 product as described in an existing configuration for AVHRRL1 products as specified in the YAML configuration file `format-geotiff.yaml`.

```
>>> from epct import api
>>> from yaml import safe_load

# specify the input product paths
>>> input_products = [
...     '<my_input_dir>/AVHRRL1 AVHR_xxx_1B_M01_20180120003103Z_20180120003403
Z_N_O_20180120004248Z',
... ]

>>> config_path = '<my_input_dir>/format-geotiff.yaml'

# load the configuration in a dictionary
>>> with open(config_path, 'r') as f:
...       chain_config = yaml.safe_load(f)

# run the chain
>>> output_files = api.run_chain(
    product_paths=input_products,
    chain_config=chain_config,  # the key-name of the desired existing config
uration
    target_dir='<my_output_dir>'
)

# get the output file paths
>>> output_files
```

#### 2.3.3.3 Example: Run a customisation applying a ROI defined by a Shapefile and return the result as an xarray dataset

This example shows how to run a customisation involving the ROI Extraction using a Shapefile (central_germany.zip) and retrieve the result as an xarray object, with the configuration specified directly as a dictionary.

```
>>> from epct import api

# define the configuration of the functional chain to apply:
>>> chain_config = {
...         'name': 'sample_chain',
...         'product': 'HRSEVIRI',
...         'format': 'netcdf4',
...         'projection': 'geographic’,
...     }

# specify the input product paths and load the shapefile
>>> input_products = [
...     '<my_input_dir>/MSG3-SEVI-MSG15-0100-NA-20160701121240.370000000Z-201
60701121258-1263454-1.nat',
...     ]

>>> with open('central_germany.zip', 'rb') as f:
...     shapefile_stream = f.read()


# run the chain and return the result as an `xarray` object
>>> output_xarray_dataset = run_chain_to_xarray(
...    product_paths=input_products,
...    chain_config=chain_config,
...    target_dir='<my_output_dir>',
...    shapefile_stream=shapefile_stream
... )
```

### 2.3.4 Explore and manage the EUMETSAT Data Tailor configuration through the API

This section describes how to explore and manage the EUMETSAT Data Tailor configuration through the API.

If you are not familiar with such configuration, the Appendix "Understanding the EUMETSAT Data Tailor" configuration provides a useful overview.

#### 2.3.4.1 Reading the configuration via API

The key function to retrieve available configuration is read, which allows to query the general configuration and/or product configurations.

##### 2.3.4.1.1 Retrieving product types

The following provides an example on how to see all product types:

```
>>> from epct import api
>>> api.read('products')
```

The output will be a dictionary with the product keys as the keys and the product properties as the values.

##### 2.3.4.1.2 Retrieving ROIs and other product-independent configuration

To see all the pre-configured ROIs execute the following:

```
>>> from epct import api
>>> rois = api.read('rois')
```

where `rois` returns a dictionary with the ROI ids as the keys and the ROI properties as the values. For example the details of the ROI with id `albania` would be:

```
>>> rois['albania']
{'NSWE': [42.688247, 39.624998, 19.304486, 21.02004], 'name': 'Albania', 'id'
: 'albania'}
```

The same syntax can be used to retrieve product independent configuration for "sections" other than `rois`. To see available sections, invoke `api.read(None)` in the Python shell or script.

##### 2.3.4.1.3 Retrieving product-dependent information

Adding the keyword product followed by the product type the sub-command shows all the available configurations for a given product type. For example to see supported output formats for the product type `HRSEVIRI`, call:

```
>>> api.read('formats', product='HRSEVIRI')
```

which returns the dictionary of the supported formats with the format ids as keys, e.g.:

```
{
     "geotiff": {
          "name": "GeoTiff",
          "description": "GeoTIFF is a public domain metadata standard ...",
          "ext": ".tif",
          "id": "geotiff"
     },
     "hdf4": { ...},
}
```

To show the contents of the `natural_color_disc` predefined configuration for HRSEVIRI:

```
>>> api.read('chains/natural_color_disc', product='HRSEVIRI')
```

which returns a dictionary like:

```
{
     "filter": "hrseviri_natural_color",
     "format": "geotiff",
     "id": "natural_color_disc",
     "name": "Natural color disc",
     "product": "HRSEVIRI"
}
```

To expand all configuration sections and get more details set the keyword argument expanded to `True`:

```
>>> api.read('chains/natural_color_disc', product='HRSEVIRI', expanded=True)
```

which returns a dictionary like:

```
{
 "filter": {
      "name": "Natural color",
      "product": "HRSEVIRI",
      "bands": [
       {
        "name": "NIR1.6",
        "number": 3,
        "id": "channel_3"
       },
       {
        "name": "VIS0.8",
        "number": 2,
        "id": "channel_2"
       },
       {
        "name": "VIS0.6",
        "number": 1,
        "id": "channel_1"
       }
      ],
      "id": "hrseviri_natural_color"
 },
 "format": {
      "name": "GeoTiff",
      "description": "GeoTIFF is a public domain metadata standard ...",
      "ext": ".tif",
      "id": "geotiff"
 },
 "id": "natural_color_disc",
 "name": "Natural color disc",
 "product": "HRSEVIRI",
 "source_text": "product: HRSEVIRI\nname: Natural color disc\nfilter: hrsevir
i_natural_color\nformat: geotiff"
}
```

#### 2.3.4.2 Save a piece of configuration

Users can save a piece of configuration using the API api.save, which saves a new configuration for the following sections: `chain`, `filter`, `roi` and `quicklook`. Users can pass the configuration as a dictionary; if it is in a YAML file, it can be converted to a dictionary using `yaml.safeload` as in a previous example.
For example, to save a new ROI with the name "`ROI test`" and with the following bounding-box: Nord=20 deg, South=10 deg, West=30 deg and East=50 deg:

```
>>> from epct import api

# the "name" is visualized inside the GUI
>>> roi_config = {'name': 'ROI test', 'NSWE': [20, 10, 30, 50]}

>>> api.save('roi', config_to_save=roi_config)
```

On success, the last command returns a dictionary where the key is the id of the saved ROI.

#### 2.3.4.3 Deleting a piece of configuration

To remove an existing piece of configuration, the API `api.delete` is used, e.g. to remove the ROI saved in the previous paragraph:

```
>>> api.delete('roi', 'roi_test')
```

Which on success returns the id of the deleted ROI.

### 2.3.5 A more complex example: integrating the EUMETSAT Data Tailor API with Pytroll in a Python script

This section illustrates how to tackle a simple scenario where the EUMETSAT Data Tailor API is used alongside Pytroll. The main steps are (the tool to use to compute operations is in parentheses):

- identify good passes from a set of products (*Pyorbital*);
- select the corresponding files from a folder (*trollsift*);
- reproject the selected file(s) (*EUMETSAT Data Tailor*).

The steps are described in the following paragraphs.

#### 2.3.5.1 Identify good passes over Madrid for Metop-B satellite

To identify good passes of a sensor (satellite) over a defined location (latitude, longitude, altitude) and time (start_search_time, time_interval), we will use Pyorbital as follows:

```
>>> from datetime import datetime
>>> from pyorbital.orbital import Orbital

# Madrid
>>> latitude = 40.41
>>> longitude = -3.7
>>> altitude = 52  # meters

>>> start_search_time = datetime(2019, 2, 1, 8, 0)
>>> time_interval = 12  # hours

# set satellite's orbit
# if a TLE file is available, use "tle_file" option
>>> tle_file = '<my_laytest_metopb_tle.txt'
>>> orbit_tle = Orbital('Metop-B', tle_file=tle_file)
```

Available info for good passes over a specified location are:

- rise time of the satellite;
- fall time of the satellite;
- max elevation time of satellite;
- azimuth and elevation of the satellite in a specified time.

Using the information above we can deduce the visibility time of the satellite from a specified location.

```
# Metop B passes using TLE
>>> passes = orbit_tle.get_next_passes(
        utc_time=start_search_time,length=time_interval, lat=latitude,
        lon=longitude, alt=altitude
    )

>>> for p in passes:
        print('Rise: ', p[0])
        print('Fall: ', p[1])
        print('MaxElevation: ', p[2])
        az, el = orbit_tle.get_observer_look(p[2], longitude, latitude,
            altitude)
        print('Azimuth: ',  az)
        print('Elevation: ',  el,)
        print('Visibility time: ', p[1] - p[0], '\n---\n')
```

### 2.3.5.2 Select Metop-B product files correspondent to good passes

In this step, we want to select the input product files for which:

- `satellite Rise time < sensing start time < satellite Fall time`

and possibly:

- `sensing end time < satellite Fall time`

i.e. the images with the target visibility as long as possible.

Using trollsift module it is possible setting the name convention for a specific sensor/satellite to automatically get acquisition info from the data file names. For example:

```
>>> import glob
>>> import os
>>> import trollsift

>>> data_dir = 'avhrrl1/'
>>> files = glob(data_dir + '*.nat')

>>> eum_fmt = '{instrument_id}_{product_type}_{processing_level}_{spacecraft_id}_'
>>> eum_fmt += '{sensing_start:%Y%m%d%H%M%S}Z_{sensing_end:%Y%m%d%H%M%S}Z_'
>>> eum_fmt += '{processing_mode}_{disposition_mode}_{processing_time:%Y%m%d%H%M%S}Z'
>>> data = {f: trollsift.parse(os.path.join(data_dir, eum_fmt), f) for f in files}
```

In this way the user can easily access to info as start-sensing-time and stop-sensing time of the available products and check the target visibility for each product.

#### 2.3.5.3 Reproject the selected file(s) using the EUMETSAT Data Tailor

To re-project the selected files(s) users have only to define the right functional chain and run the processing using the EUMETSAT Data Tailor API `run_chain`. So, here the user can set:

- a name for the processing chain;
- the output format of the result;
- the extent of the ROI or the name of a predefined ROI;
- the geographic projection of the output;
- type of aggregation.

```
from epct import api

# define chain processing task
>>> chain_name = 'Metop-B chain'
>>> output_format = 'netcdf4'
>>> roi = [62, 23, -35, 10]  # custom ROI
>>> projection = 'geographic'
>>> aggregation = 'orbit'
>>> chain_dict = {
        'name': chain_name,
        'product': 'AVHRRL1',
        'format': output_format,
        'roi': {'NSWE': roi},
        'projection': projection,
        'aggregation': aggregation
    }

# here we launch the processing chain
>>> selected_files = <list_of_input_files>
>>> out = api.run_chain(
        product_paths=selected_files,
        chain_config=chain_dict,
        target_dir='<my_output_dir>'
    )
```

### 2.3.6 Another advanced example: Configuring the EUMETSAT Data Tailor for a EUMETCast Station

An example on how to use EUMETSAT Data Tailor on a EUMETCAST Station is provided in Exercise 4 of EUMETSAT Product Customisation Toolbox - Installation Guide & Exercises (tables 3 and 4). The script simulate-customise.py (table 4) needs to be adapted as follows:

```
#!/usr/bin/env/python
# Every second, process the first MSG native file found in input directory.
#
# Output files replace input files. Native files are processed using
# natural-color-disc customisation chain.
#
# Call signature:
```

```
# ./simulate-eumetcast.py $INPUT_FOLDER
import argparse import os
import subprocess import time
# Parse CLI input
parser = argparse.ArgumentParser(description="Every second, process the first "
                                             "MSG native file found in input "
                                             "directory.")
parser.add_argument("input_folder")
args = parser.parse_args()
# Indefinite loop
while True:
    # Placed here so that script doesn't idle indefinitely without input
    time.sleep(1)
    # Look up MSG native file in input folder
    input_files = [os.path.join(args.input_folder, x)
                   for x in os.listdir(args.input_folder)
                   if os.path.splitext(x)[1] == ".nat"] # If no files found, return
to beginning
    if not input_files:
        continue
    # Execute chain.
    input_file = input_files[0]
    command = ["epct", "chain-run", "-c", "natural-color-disc", "-p",
               "HRSEVIRI", "-o", args.input_folder, input_file]
    failed = subprocess.call(command)
    # If successful, delete input file.
    if not failed:
        os.remove(input_file)
```

### 2.3.7 Short review of other API entry points

Other useful APIs are listed below:

- `api.config`: lists all the available configurations. With the `force_etc_reset` option, it restores the default configuration, removing the one in use.
- `api.info`: retrieves the information on the current EUMETSAT Data Tailor installation
- `api.customisations`: returns a dictionary with all customisations, or with the details of a specific one.
- `api.log`: Display the log for a process with `PROCESS_UUID` UUID.
- `api.manage_process`: manages a running process to see its status or to send it a signal (`kill`, `suspend`, `resume`).
- `api.aggregation_analyse`: shows the result of the aggregation analysis for a given product and for the chosen input files. An example checking the feasibility of an aggregation by *orbit* for two *AVHRRL1* PDU products is provided below.

```
>>> from epct import api

>>> product = 'AVHRRL1'
>>> format = 'geotiff'
>>> input_products = [
        '<my_input_dir>/AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N
_O_20180120004248Z',
        '<my_input_dir>/AVHR_xxx_1B_M01_20180120003403Z_20180120003703Z_N
_O_20180120013050Z',
    ]
>>> aggregation_type = 'orbit'

>>> api.aggregation_analyse(product, format, input_products, aggregate_by
=aggregation_type)
Out: {
        'extent': {
            'Orbits in selection': ['27708-27709'],
            'Days in selection': ['2018-01-20'],
            'Eligibility': [
                "AGGREGATION 'BY ORBIT' IS POSSIBLE.",
                "AGGREGATION 'BY DAY' IS POSSIBLE."
            ]
        },
        'outcome': {
            '27708-27709_seq1': {
                'orbit': '27708-27709',
                'nr_elements_of_sequence': 2,
                'products': [
                    'AVHR_xxx_1B_M01_20180120003103Z_20180120003403Z_N_O_
20180120004248Z',
                    'AVHR_xxx_1B_M01_20180120003403Z_20180120003703Z_N_O_
20180120013050Z'
                ],
                'start_sensing_time': '2018-01-20T00:31:03',
                'stop_sensing_time': '2018-01-20T00:37:03',
                'data_amount_to_read': '53.1 MB'
            }
        }
    }
```

## 2.4 Troubleshooting

### 2.4.1 Reset of the configuration

The default configuration could messed up and the EUMETSAT Data Tailor stops being functional. This may happen e.g.:

- during an incomplete/partial software update;
- if configurations are edited by hand and some error is introduced which is difficult to track afterwards;
- if lots of experimental configurations are introduced, and the user needs to revert to the default state.

A specific CLI command (and the correspondent API function) is provided to reset the configuration to the default state:

```
epct config --force-etc-reset
```

Warning
The option `--force-etc-reset` of the `epct config` CLI (or `force_etc_reset` keyword of the corresponding API config) is **dangerous**: this command clears the configuration folder and restore defaults. So, **any user customization about functional chains, layer filters, ROIs and quick-look configurations will be lost!**

### 2.4.2 Hard drive memory issue

The EUMETSAT Data Tailor uses a directory, named ``*workspace_dir*``, to storage, during the execution of a processing chain, intermediate products and logs. The default path to the ``*workspace_dir*`` is set automatically and it is therefore platform-dependent. The path to a different ``*workspace_dir*`` storage can be optionally specified through the external interfaces, overriding the default.
When a new processing is launched (by any interface), the EUMETSAT Data Tailor performs the following tasks:
1. generates a unique process name, i.e. a process ID
2. creates the directory, named ``*workspace_dir*``/``*process_ID*``, where the intermediate products will be stored
3. initialize the process-specific log
4. performs consistency checks
5. invokes the module which governs the execution of a configuration chain
6. the output product(s) is (are) moved from the ``*workspace_dir*``/``*process_ID*`` into the target directory specified by the user.

**At the end of the process the EUMETSAT Data Tailor does not remove the processing directory where intermediate products and logs are stored.**

Warning: The EUMETSAT Data Tailor will never empty its ``*workspace_dir*`` where all intermediate products and logs for each processing are stored. Among intermediate products there are also the OLDA (OnLine Data Access) products downloaded before the processing. So, at some point the user will need to remember to manually remove all or some working directories inside the ``*workspace_dir*`` or his hard drive will get full!

### 2.4.3 RAM issue

The user inadvertently can require processes which could consume a large amount of the host computer resources; this could potentially freeze the computer. The scenario may in principle occur when a user selects several input products and tries to customize them. While no specific recovery strategy is designed for this scenario, several mitigation strategies are put in place to reduce the probability for it to happen:
- a few functionalities declare and enforce limits on the amount of data which can be processed by a single process. As an example, aggregation of EPS products is limited to a full orbit;
- when using the CLI, only customization processes generating one output each can be run. As a consequence, the resource footprint of each process is limited, and if several processes are executed at the same time, the standard process manager of the operating system takes care of controlling the resources (it may e.g. kill some processes, but as processes are independent, the other can continue);
- when using the GUI, even if several processes are requested, only fixed-size batches are executed, once a process concludes, the next one is executed.

## APPENDIX A UNDERSTANDING THE EUMETSAT DATA TAILOR CONFIGURATION

The EUMETSAT Data Tailor allows concatenating several customisation functions (layer filtering, aggregation, re-projection, etc.), by simply specifying in the customisation configuration which functions should be used. The Data Tailor then selects the order in which they are executed, transparently to the user.

This solution offers a simple and transparent way to configure a configuration; this benefits from the human-readable syntax of the configuration files, which is described below.

To store the configuration of a customisation, the EUMETSAT Data Tailor uses YAML files, in which the selections of functions and the related parameters are specified by means of key-value pairs separated by a colon ":". In the case of Data Tailor, the allowable values for both keys and values are just very simple keywords which define the desired processing steps (e.g. `aggregation`, `projection`) and the corresponding chosen options (e.g. "`aggregation`": "`orbit`", "`format`": "`geotiff`"). The following is a basic example of a YAML configuration file:

```
name: Natural color
product: AVHRRL1
filter: natural-color
format: geotiff
projection: geographic
ROI: western-europe
```

This example illustrates a few features of the EUMETSAT Data Tailor configuration files:

- **a customisation configuration can be configured in a plain-text file**: in the example, a configuration named Natural Color, valid for AVHRRL1 products, applies a predefined layer filter (`natural-color`), performs a re-projection to the geographic projection, selects a pre-defined Region of Interest ("`western-europe`") and applies a format conversion to the GeoTIFF format. Such definition is highly readable, and in principle can be modified with a text editor, provided the users know the specification;
- **a "master" YAML file can refer to other pre-defined pieces of configuration**: in the example, the layer filter `natural-color` will apply the configuration for that filter, allowing it to be reused across configurations and removing the need for replicating the details in each master file. This also improves the readability of the master configuration file. The same approach is used in the example for the definition of the `ROI`, of the projection and of the `format`;
- **a customisation configuration can be reused for several input products**. Reuse is one of the key advantages in using a file-based configuration.

The configuration can also be specified by providing the detailed parameters for one or more customisation functions. For example, the configuration in the example above could be written alternatively as:

```
name: Natural color
product: AVHRRL1
filter:
    bands:
          - w1.606-3.722
          - w0.842
          - w0.633
format: geotiff
projection: geographic
ROI: western-europe
```

The example illustrates more features of the YAML-based approach:

- a detailed configuration is still valid;
- the functions to be activated for a given customisation are at the “first level”, their detailed configuration can be described on several lines;
- a master configuration can mix detailed specifications with predefined filters: in the example, the projection is described using a pre-defined projection.

As said, a configuration could be created or modified by a user with a text editor, the recommended approach is the use the tools provided by the external interfaces (GUI, CLI, API), which take care of ensuring integrity and consistency.

The mandatory configuration parameters are `product` and `format`.

*This Document is Public*

# APPENDIX B ADVANCED: CUSTOMISING THE EUMETSAT DATA TAILOR INSTALLATION

The EUMETSAT Data Tailor configuration is stored in a set of plain-text YAML file, which are stored in the Data Tailor configuration directory.

To find out where is the configuration directory, you can e.g. execute from a terminal:

```
epct info
```

and look for the value of etc_dir.

The key files which describe the configuration of the installation are:

- **`epct.yaml`: general configuration, including:**
  - `WORKSPACE_DIR` the directory where the temporary files are stored; leave empty to use the default
  - `root_path`: the top-level directory served by the Web Service Interface; leave it to `/` to access the whole filesystem
  - `naming_convention`: the default naming convention, which can be customised with the fields described in the commented lines
  - `api_base_path`: the base path of the Web Service Interface.
- **`epct-webui.yaml`: configuration of the GUI, in particular:**
  - `api_url`: the URL where the Web Service Interface
  - `localhost`: the address where the GUI is served
  - `port`: the port where the GUI is accessible.

If the configuration of the GUI is changed, the GUI must be restarted.

The configuration of the Web Service Interface is currently not managed through YAML files; however, the host and port can be configured as required by `waitress-serve`, e.g.:

```
waitress-serve --host=localhost --port=8001 --call epct_restapi:app
```

# APPENDIX C ADVANCED: INSTALLING OR REMOVING CUSTOMISATION PLUGINS

Customisation plugins provide customisation capabilities for a set of input products and output formats.

They can be installed and removed at run-time; the EUMETSAT Data Tailor capabilities will change accordingly.

To install a customisation plugin (assuming the corresponding Conda package has been built on the target machine), use `conda install` in a terminal with the `epct-desktop` environment active. E.g. to install an `epct_plugin_umarf` plugin, execute:

```
conda install -c local -c anaconda epct_plugin_umarf
```

After the command, you can check that the plugin is installed with `epct info`.

You can also note that the supported formats for the MSG HRSEVIRI L1.5 now include ISCCP formats, executing:

```
epct read -p HRSEVIRI formats
```

Note: some customisation plugins rely on external executables to be present on the target machine, otherwise they may be installed, but not providing capabilities. Refer to the plugin installation instructions for details.

You can remove a plugin with `conda uninstall`, e.g.:

```
conda uninstall epct_plugin_umarf
```

You can check with `epct info` that the plugin has been removed and with e.g. `epct read -p HRSEVIRI` formats that the ISCPP formats are no longer supported.