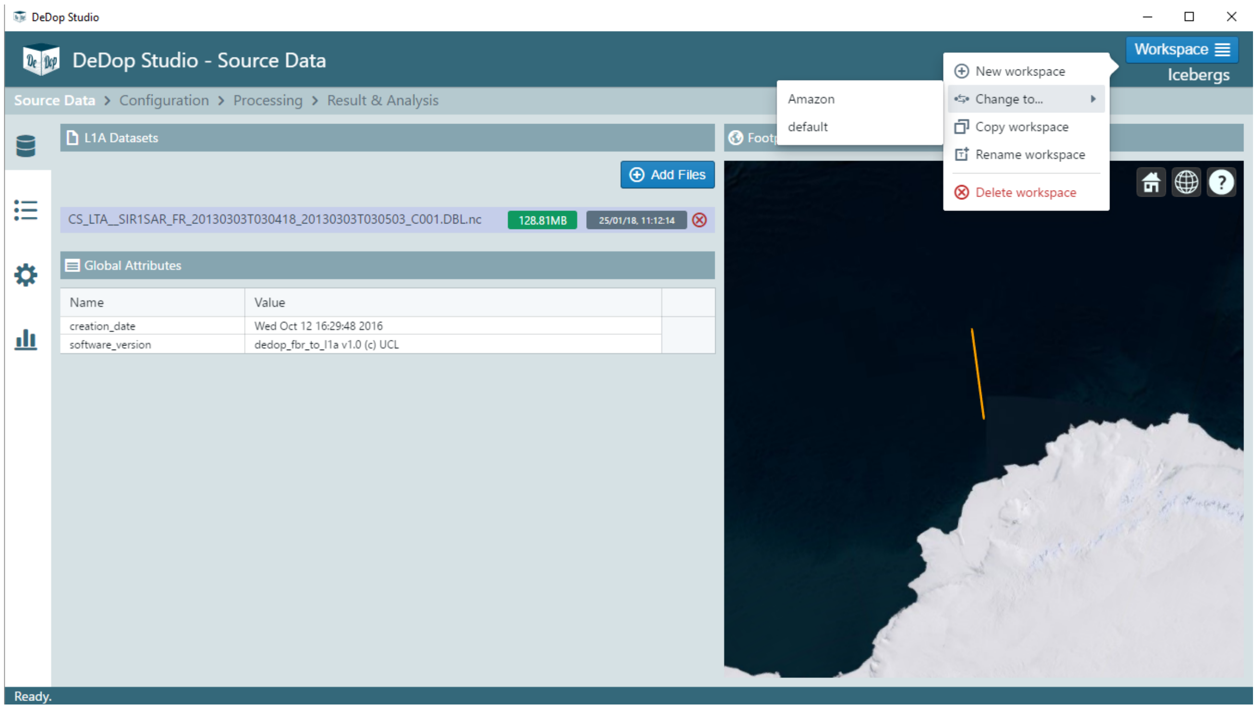
DeDop Documentation

Release 1.5.1.dev1

DeDop Development Team

Feb 22, 2019

Table of Contents



CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL.nc 128.81MB 25/01/18, 11:12:14

Name	Value
creation_date	Wed Oct 12 16:29:48 2016
software_version	dedop_fbr_to_l1a v1.0 (c) UCL

CHAPTER 1

Introduction

DeDop: a User Configurable Tool for Processing Delay Doppler Altimeter Data.

1.1 Delay Doppler Altimeter Data

The recent development of SAR altimetry, or more properly Delay Doppler altimetry, as first implemented on CryoSat-2 and now also operated on Sentinel-3, opens an exciting new era for the scientific community. This new approach offers to scientists an opportunity to develop new processing schemes and derive new and improved products, and so maximise the benefits of the measurements available from upcoming missions. Historically, in conventional altimetry, the understanding of the Level 1b processor was the responsibility of the instrument engineers with system expertise. This was logical as the subsequent levels of processing only needed the results produced by the Level 1b processor (Level 1b product) and the information contained in it. There were not many different correct ways of processing the raw data up to Level 1b.

However, this approach is no longer appropriate for SAR Altimetry or Altimeter Delay Doppler Processing. The links between the Level 1B processing and the Level 2 processing, particularly the retracking of the waveforms, are very strong. For instance, different ways of performing the delay Doppler processing lead to different L1B waveform shapes,and peculiarities of the *Delay Doppler Processor* (DeDop processor) have a noticeable effect on the L1B waveform leading to changes in the geophysical retrievals.

Due to this strong link between the Level 1b processing and the final geophysical retrievals, it is important that the SAR Altimetry scientific community gains a much better understanding the Level 1B processor, and is involved in new developments. For various reasons (e.g. the novelty of the processing, previous unavailability of adequate documentation, restricted availability of the low levels of data), this understanding is currently limited, both in general terms of what a DeDop processor is, and in the different options chosen for the different missions.

1.2 The DeDop^3 Project

The DeDop^3 project provides the scientific community with the means to understand and use low level altimetry data and to become involved in how these data are processed. This is achieved by providing them with a Fully Adaptable and Configurable DeDop and a friendly user interface (a tool) to help them to interact with DeDop processor. The proposed DeDop has different options from which the user will be able to choose according to their particular field of interest. Examples of the new options are: surface focusing (particularly relevant for special targets like coasts, rivers or lakes), any kind of weighting along and across track, different azimuth processing approaches, stack masking, new stacking algorithms (e.g. ACDC), Sigma-0 at stack level, etc.

DeDop is open source and available in such a way that users are able to explore the code, understand its possibilities, and to modify it to their own needs.

The tool also comes with various demonstrations of new features that can be investigated and retrieved when using these lower data processing levels. They are presented as successful case studies.

1.3 The DeDop Tool

The DeDop tool comprises both a command-line interface, the *DeDop Shell*, and a graphical user interface, the *DeDop Studio*. The tool's primary goal is to make it easy to configure and run and to provide a number of analysis functions to inspect and compare the L1B results. The tool's target users are community scientists wishing to learn, modify or extend the DeDop processor configuration and/or code and then use the tool for comparisons between outputs of varying configurations generated by the DeDop.

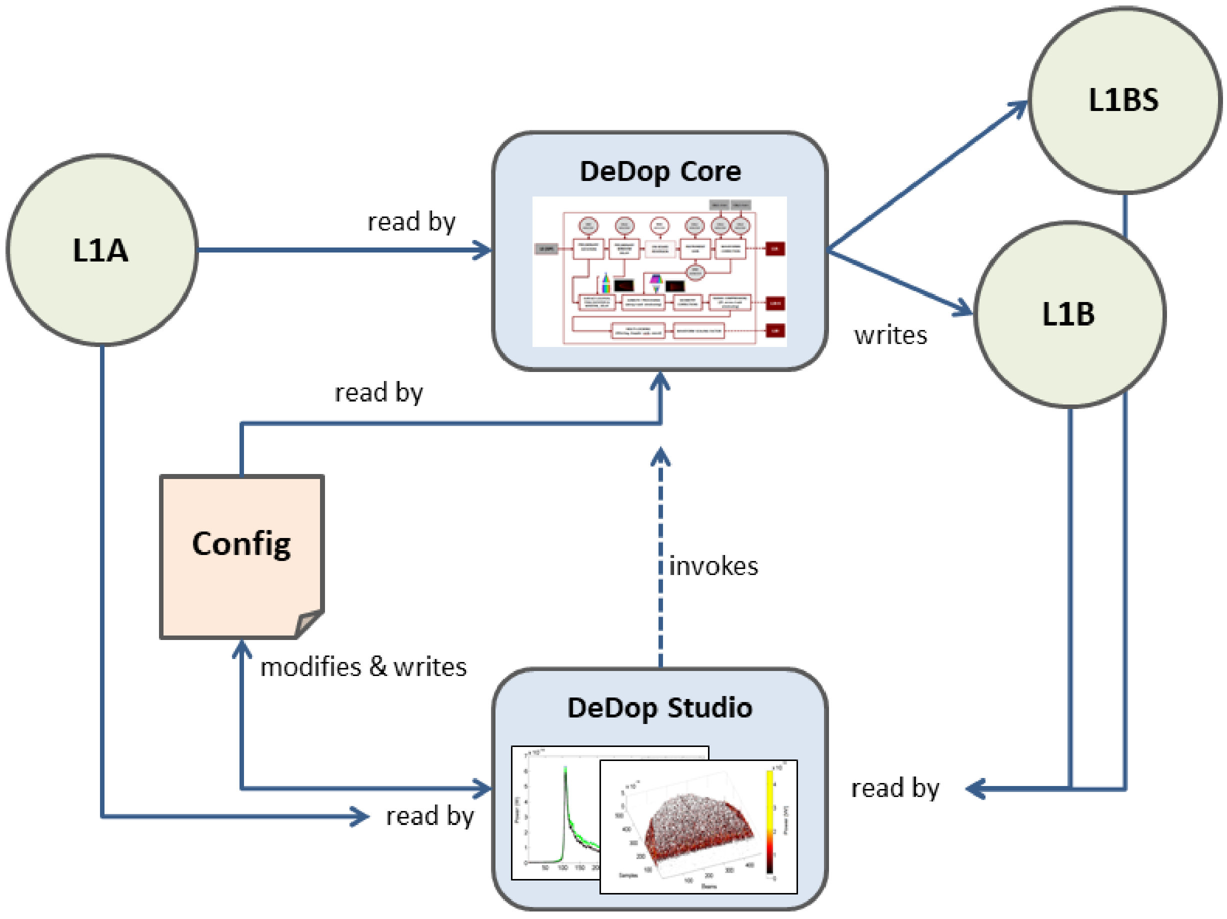


Fig. 1: The DeDop and the Tool context

The image above shows two components: DeDop Studio and DeDop Core. DeDop Core consists of **DeDop processor**, **DeDop Shell**, and **DeDop webapi**. With DeDop Shell, users will be able to do all the operations using a command-line interface. Information on how to use DeDop Shell can be found in *DeDop Shell*. With DeDop Studio, users can also perform the same operations as in DeDop Shell (modifies & writes config, read L1A data, etc.) and in the end it invokes DeDop processor in DeDop Core via the webapi interface. More information about DeDop Studio can be found in *DeDop Studio*.

CHAPTER 2

Quick Start

This chapter is a tutorial for users new to the DeDop Shell.

Refer to *Setup* for installing DeDop.

2.1 Using DeDop Shell

In the new terminal window, type:

```
$ dedop -h
```

to list the available DeDop sub-commands. You can get help on sub-commands as well:

```
$ dedop run -h
```

2.1.1 Processing L1A to L1B

To perform L1A to L1B processing using the Delay Doppler Processor (DDP) you need to add one or more L1A input files to your current DeDop *workspace*. For example, download Amazon or Iceberg L1A data. And then, add the file(s) to your dedop workspace:

```
$ dedop input add some/path/to/CS_LTA__SIR1SAR_FR_20150331T034023_20150331T034235_
↪C001.DBL.nc
created workspace "default"
current workspace is "default"
adding inputs: done
one input added
```

From the output, you can see that a new workspace named `default` and a new DDP configuration also named `default` Now run the processor with default settings:

```
$ dedop run
created DDP configuration "default" in workspace "default"
current DDP configuration is "default"
processing ~/.dedop/workspaces/default/inputs/CS_LTA__SIR1SAR_FR_20150331T034023_
↪20150331T034235_C001.DBL.nc using "default"
processing: [##########------------------------------] 25%
...
processing took 0:05:03.159575
```

If the command succeeded, the L1B output files can be found in `workspaces/default/configs/default/outputs`, which is by default located in the DeDop user data directory. On Unix and MacOS this directory is `~/.dedop` while on Windows it is `C:/Users/<username>/.dedop`. The location of your workspaces directory can be changed by configuration. Please refer to the *Configuration Parameters*.

If the processor run was successful you can observe new L1B and L1BS files:

```
$ dedop output ls
2 outputs created with config "default" in workspace "default":
  1: L1BS_CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL_default.nc
  2: L1B_CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL_default.nc
```

To inspect and interact with the generated L1B file:

```
$ dedop output inspect L1B_CS_LTA__SIR1SAR_FR_20150331T034023_20150331T034235_C001.
↪DBL_default.nc
wrote notebook file "C:\Users\dedop-user\.dedop\workspaces\default\notebooks\inspect-
↪L1B_CS_LTA__SIR1SAR_FR_20150331T034023_20150331T034235_C001.DBL_default.nc.ipynb"
calling: start "DeDop - inspect - [...AR_FR_20150331T034023_20150331T034235_C001.DBL_
↪default.nc]" /Min "C:\Users\dedop-user\.dedop\temp\dedop-notebook-server.bat"
A terminal window titled "DeDop - inspect - [...AR_FR..." has been opened.
Close that window or press CTRL+C within it to terminate the Notebook session.
```

NOTE: the command above was executed in Windows machine. Command output in Unix or MacOS maybe different but comparable.

This command should open up a web browser window that displays an interactive *Jupyter Notebook*. Also a second terminal window should have opened up. This hosts the Jupyter Notebook server process. If you are not familiar with Jupyter Notebook, read the introduction here. Try to run the commands in the Notebook by selecting **Cell > Run All**. Here is a sample inspect Notebook to show how it looks like for a successful run of `inspect` notebook (some graphics may not be able to be displayed by GitHub). When you are done with the Notebook, simply close the Notebook tab in the browser as well as the second terminal window.

If you like to perform your analysis in batch mode rather than using the interactive Notebook, you can do so by writing your own analysis script in Python. An example is given in DeDop's source code repository: inspect-script.py.

This script can by run with the Python interpreter bundled with the DeDop Core installation. To generate a multi page PDF, with all the figures use an output filename that has the `.pdf` extension:

```
$ python inspect-script.py "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\default\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_default.nc" inspect-out.pdf
```

To generate a directory of figure images, run the script with a directory name or with `dir` as 3rd argument:

```
$ python inspect-script.py "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\default\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_default.nc" inspect-out dir
```

2.1.2 Changing the processor configuration

From the last steps above, the current DDP configuration should be still `default`. Verify:

```
$ dedop status
configuration location:     ~/.dedop/config.py
workspaces location:        ~/.dedop/workspaces
workspaces total size:      150 MiB
workspace names:            default
current workspace:          default
current DDP configuration:  default
```

We will now create a new configuration `myconf`, type:

```
$ dedop config add myconf
created DDP configuration "myconf" in workspace "default"
current DDP configuration is "myconf"
```

From the output, you can see that a new DDP configuration named `myconf` has been created and is now the current one. To modify the configuration, type:

```
$ dedop config edit
```

Now three DDP configuration files should have been opened in your default text editor. Their format is JSON. You may change any DDP configuration settings now, for example, in the `CHD.json` (the *characterisation definition file*) change the value of the parameter `uso_freq_nom_chd` from its default value `10e6` to `7.5e6`:

```
"uso_freq_nom_chd": {
     "value": 7.5e6,
     "description": "USO nominal frequency",
     "units": "Hz"
},
```

Save the configuration file in your text editor.

Now run the processor with the modified DDP configuration `myconf`:

```
$ dedop run
```

We can now compare the L1B outputs in an interactive Jupyter Notebook:

```
$ dedop output compare -C default L1B_CS_LTA__SIR1SAR_FR_20150331T034023_
↪20150331T034235_C001.DBL_myconf.nc L1B_CS_LTA__SIR1SAR_FR_20150331T034023_
↪20150331T034235_C001.DBL_default.nc
wrote notebook file "C:\Users\dedop-user\.dedop\workspaces\default\notebooks\compare-
↪L1B_CS_LTA__SIR1SAR_FR_20150331T034023_20150331T034235_C001.DBL_myconf.nc-L1B_CS_
↪LTA__SIR1SAR_FR_20150331T034023_20150331T034235_C001.DBL_default.nc-1.ipynb"
calling: start "DeDop - compare - [...1T034235_C001.DBL_myconf.nc] [...T034235_C001.
↪DBL_default.nc]" /Min "C:\Users\dedop-user\.dedop\temp\dedop-notebook-server.bat"
A terminal window titled "DeDop - compare - [...1T034..." has been opened.
Close that window or press CTRL+C within it to terminate the Notebook session.
```

When you pass just file *names* to the `dedop output compare` command, DeDop must know to which configurations they refer to. The first filename corresponds to the *current* DDP configuration or the one given by the `-c` option. The second filename corresponds to a DDP configuration given by the `-C` (upper case!) option. Here is a sample compare Notebook to show how it looks like for a successful run of `compare` notebook (some graphics may not be able to be displayed by GitHub).

Instead of specifying the configuration name, you can also pass any two L1B *file paths* to the `dedop output compare` command:

```
$ dedop output compare "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\default\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_default.nc" "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\myconf\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_myconf.nc"
```

Again, if you like to perform your analysis in batch mode, you can do so by writing your own comparison analysis script in Python. An example is given in DeDop's source code repository: compare-script.py.

This script can be run with the Python interpreter bundled with the DeDop Core installation. To generate a multi page PDF use an output filename that has the `.pdf` extension:

```
$ python compare-script.py "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\default\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_default.nc" "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\myconf\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_myconf.nc" compare-out.pdf
```

To generate a directory of figure images, run the script with a directory name or with `dir` as 4th argument:

```
$ python compare-script.py "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\default\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_default.nc" "C:\Users\dedop-user\.
↪dedop\workspaces\default\configs\myconf\outputs\L1B_CS_LTA__SIR1SAR_FR_
↪20150331T034023_20150331T034235_C001.DBL_myconf.nc" compare-out dir
```

CHAPTER 3

User Manual

3.1 Setup

3.1.1 DeDop Core Installation

From Binaries

DeDop Core is distributed as pre-compiled binaries which can be retrieved from here. For the windows Windows platforms there is a dedicated installer executable. For MacOS and Unix a shell script file is provided. All platform distributions have a bundled Python interpreter, which makes it rather large in size (roughly 300-400 MB). As mentioned before, this package also contains DeDop processor and DeDop webapi. To install DeDop Core in Windows, double click the `exe` file.

In MacOS, run the following commands:

```
chmod u+x DeDop-core-1.x.x-MacOSX-x86_64.sh
./DeDop-core-1.x.x-MacOSX-x86_64.sh
```

In Unix, run the following commands:

```
chmod u+x DeDop-core-1.x.x-Linux-x86_64.sh
./DeDop-core-1.x.x-Linux-x86_64.sh
```

During installation, you will be requested the installation location as well as a preference to add this location to your PATH. When unsure, just follow the default (no).

Run the following command to start DeDop Shell:

```
<dedop_core_installation_dir>\Scripts\dedop-shell.bat    # Windows
<dedop_core_installation_dir>/bin/dedop-shell.command    # MacOS
<dedop_core_installation_dir>/bin/dedop-shell.sh         # Unix
```

From Source

DeDop is programmed in Python so you first need to setup a suitable Python environment. We recommend using a Miniconda Python3 environment to create an isolated environment that is independent from the default Python in your machine. Refer to this page for the instruction on how to install Miniconda on your machine. It is recommended to install Miniconda with Python 3.x since DeDop Core uses Python 3.

To install DeDop Core from source, first you need to checkout the DeDop Core GitHub repository:

```
git clone https://github.com/DeDop/dedop-core.git
```

Step into the newly created source directory:

```
cd dedop-core
```

After installing Miniconda, open a terminal window and create `dedop` environment with all the required dependencies as listed in *environment.yml* by typing:

```
<miniconda_installation_dir>/bin/conda env create --file environment.yml
```

To activate this environment, type:

```
source <miniconda_installation_dir>/bin/activate dedop      # Linux, MacOS
<miniconda_installation_dir>\Scripts\activate dedop         # Windows
```

Install DeDop in the Python environment *dedop*:

```
python setup.py install
```

After the installation is finished, start DeDop Shell by typing:

```
dedop --help
```

If you plan to run DeDop Studio, please run the following command:

```
dedop-webapi --version
```

This is necessary to trigger the creation of *~/.dedop/<version_num>/dedop-location* file. In the installation from binary, this is not required because at the end of the installation, this file is automatically created.

3.1.2 DeDop Studio Installation

As illustrated in *this diagram*, DeDop Studio is dependent on DeDop Core in performing any processing tasks. For this reason, make sure that DeDop Core has been installed in your computer before performing DeDop Studio installation. Failure to do that will result in the DeDop studio failing to startup.

From Binaries

DeDop is distributed as pre-compiled binaries which can be retrieved from DeDop Studio release page. For the windows Windows platforms there is an easy one-click installer executable. For MacOS, the installer is available as dmg file and zip file, while for Unix, the installer is available as tar.gz, zip, and AppImage files. All DeDop Studio installers (except the Unix tar.gz and zip files) are light-weight and executed by double clicking them. They don't require any extra user input.

Please note that in Windows, you may need to accept the warning at the beginning in regards to installing a third-party software. Similarly in MacOS, you may need to allow installation of apps downloaded from *App Store and identified developers*. More information about it can be found in Gatekeeper page.

After the installation has been completed, you will find an executable file named *DeDop-Studio*, which you can search using start menu (Windows) or Spotlight (MacOS).

From Source

Pre-requisites

The following software needs to be installed on your machine before you can start installing DeDop Studio from source:

- nodejs v6.9.1
- npm v3.10.8 (comes with nodejs)

Go to here for downloading nodejs and here for the installation guide.

How-to-install

Clone dedop-studio repository:

```
git clone https://github.com/DeDop/dedop-studio.git
```

Do npm install:

```
cd dedop-studio
npm install
```

Create a *dedop-config.js* inside *dedop-studio* directory and put the location of `dedop-webapi.exe` (Windows) or `dedop-webapi` (MacOS and Linux) under `webAPIConfig` field. The location of `dedop-webapi` will be where the dedop environment is, eg.:

```
<miniconda_installation_dir>\envs\dedop\Scripts\dedop-webapi.exe     # Windows with␣
↪DeDop Core installation from source
<miniconda_installation_dir>/envs/dedop/bin/dedop-webapi             # MacOS & Unix␣
↪with DeDop Core installation from source

<dedop-core_installation_dir>\Scripts\dedop-webapi.exe               # Windows with␣
↪DeDop Core installation from binary
<dedop-core_installation_dir>/bin/dedop-webapi                       # MacOS & Unix␣
↪with DeDop Core installation from binary
```

More information about the can be found in `dedop-config.template.js`. Sample values for `webAPIConfig` in different OS's are provided here:

```
# Windows
webAPIConfig: {
  command: "C:\\Miniconda3\\envs\\dedop\\Scripts\\dedop-webapi.exe",
  servicePort: 2999,
  processOptions: {}
}

# MacOS
```

(continues on next page)

(continued from previous page)

```
webAPIConfig: {
  command: "/Users/userName/miniconda3/envs/dedop/bin/dedop-webapi",
  servicePort: 2999,
  processOptions: {}
}

# Linux
webAPIConfig: {
  command: "/home/userName/miniconda3/envs/dedop/bin/dedop-webapi",
  servicePort: 2999,
  processOptions: {}
}
```

Compile:

```
npm run compile
```

Start:

```
npm start
```

3.2 Workspace and Configuration Concept

3.2.1 Workspace

Workspace in DeDop Shell refers to a space in the file system in which all the requires parts for processing are located. They include source files, configurations, output files, as well as the Jupyter notebooks. It is possible to have multiple workspaces and by default they are located under `$USER_DIR/.dedop/workspaces`. For example:

```
C:\\Users\\dummy_user\\.dedop\\workspaces  # Windows
/home/dummy_user/.dedop/workspaces         # Linux
/Users/dummy_user/.dedop/workspaces        # MacOS
```

When DeDop Shell runs for the first time, there is no workspace available. An automatic creation of default workspace can be triggered by running `dedop input add some/path/to/your/L1A.nc` or `dedop run` command as described *here*.

Another (more recommended) way to add a new workspace is by running the following command:

```
$ dedop w add workspace_name
```

Upon successful operation, the following responses shall be returned:

```
created workspace "workspace_name"
current workspace is "workspace_name"
```

This means that the new workspace has been successfully created and made the current workspace, which means that, unless explicitly changed, whatever operations being performed after this will happen inside this workspace.

Inside `workspaces` directory, there is a file called `.current`, inside which the name of the current workspace can be found. When there is no current workspace (eg. because there is no more workspace after a deletion), this file will be empty.

More information and example on workspace management operations, please go to *here*.

3.2.2 Configuration

DeDop configuration refers to a set of configurations (Configuration, Characterization, Constants), which in the file system are stored as CNF.json, CHD.json, and CST.json. Most of the time, users will need to modify only the **Configuration** (CNF.json). A configuration is represented by a directory with a name of the configuration name and is located under `configs` directory inside a workspace directory. It is possible to have multiple configurations under each workspace and it is recommended to have a new configuration for different set of configuration values because in the end the generated output products have a one-to-one relationship with the configuration. This enables a particular output to be easily reproduced in the future, if required.

As in `workspaces` directory, there is a `.current` file inside `configs` directory. It works following exactly the same concept: this file has the information on what is the current configuration. In the case of no current configuration, this file will be empty.

More information and example on configuration management operations, please go to *here*.

3.3 DeDop Shell

3.3.1 Overview

DeDop Shell comprises a single command-line executable, which is called `dedop` and is available after installing the DeDop Shell on your computer. With this tool, users can have a complete interaction with the input data, processor, configuration, and finally the output products. If you have not done so, see section *Setup* to install and initialize DeDop Shell.

In this section, you will find a complete manual on how to use DeDop Shell. This can be sub-divided into the following parts:

- *Workspace Management*
- *Source File Management*
- *Processor Configuration Management*
- *Running the Processor*
- *Analyse the Results*

3.3.2 Workspace Management

Add a new workspace

When DeDop Shell runs for the first time, there is no workspace available. An automatic creation of default workspace can be triggered by running `dedop input add some/path/to/your/L1A.nc` or `dedop run` command as described *here*.

To add a new workspace is by running the following command:

```
$ dedop w add workspace_name
```

Upon successful operation, the following responses shall be returned:

```
$ created workspace "workspace_name"
$ current workspace is "workspace_name"
```

This means that the new workspace has been successfully created and made the current workspace, which means that, unless explicitly changed, whatever operations being performed after this will happen inside this workspace.

Remove a workspace

To remove a workspace, run one of the following commands:

```
$ dedop w remove                      # CASE 1
$ dedop w remove workspace_name       # CASE 2
```

In both cases, a confirmation prompt will appear to really ensure that this action is intentional. To disable this prompt, use the optional argument `--yes` (eg. `dedop w remove --yes`).

In **CASE 1**, the tool will remove the current workspace. It will then change the current workspace to the next workspace. When no more workspaces exist, it will just be assumed that there is no current workspace.

In **CASE 2**, the tool will remove the specified workspace. If the specified name is also the current workspace name, the above scenario will be in effect.

`remove` has an alias `rm`.

Copy a workspace

To copy a workspace, run one of the following commands:

```
$ dedop w copy                                        # CASE 1
$ dedop w copy workspace_to_be_copied                 # CASE 2
$ dedop w copy workspace_to_be_copied new_workspace   # CASE 3
```

In **CASE 1**, without specifying which workspace to be copied, it will default to the current workspace. The new workspace name is also not specified, so in this case a new name with the format of `current_workspace_name + _ + unique_number` is created. The unique number will be incremented until a unique combination is constructed. For example, for a current workspace `workspace1`, it will create a new workspace named `workspace1_1`.

In **CASE 2**, the workspace to be copied is specified, but not the name of the new workspace. It will then follow the same rule as in **CASE 1**, by creating a new workspace with a name that is of format `workspace_to_be_copied + _ + unique_number`.

In **CASE 3**, both the workspace to be copied and the name of the new workspace are specified. It is very clear that it will copy the specified workspace to a new workspace with the specified name.

One thing that is worth mentioning is that copying a workspace does not automatically change the current workspace.

`copy` has an alias `cp`.

Rename a workspace

To rename a workspace, run one of the following commands:

```
$ dedop w rename new_workspace_name                                # CASE 1
$ dedop w rename workspace_to_be_renamed new_workspace_name    # CASE 2
```

In **CASE 1**, the current workspace will be renamed to `new_workspace_name`. The current workspace will be automatically changed after the renaming.

In **CASE 2**, the specified workspace will be renamed to `new_workspace_name`. There is **NO** change on the current workspace.

`rename` has an alias `rn`.

Get current workspace

It is sometimes useful to know in which workspace we are working on at the moment. To get that information, run the following:

```
$ dedop w current
```

If there is a current workspace, the name of the current workspace will be returned. Otherwise, `no current workspace` will be returned.

`current` has an alias `cur`.

List workspaces

To list available workspaces, run the following command:

```
dedop w list
```

`list` has an alias `ls`.

3.3.3 Source File Management

After creating a workspace, the next step is to manage the L1A source files.

Add new L1A source file

To add a new L1A file(s) into this workspace, run the following command:

```
$ dedop i add /path/to/file1 /path/to/file2 /path/to/file3
```

What this command does is copying those files into the current workspace directory. When successful, those files will be located inside `inputs` directory under the current workspace directory. Every workspace will have its own `inputs` directory, which in the end can be used as a source file for running multiple processes with different configurations.

Remove L1A source file

To remove the previously-added L1A file(s), run one of the following commands:

```
$ dedop i remove                                           # CASE 1
$ dedop i remove file_name1 file_name2                     # CASE 2
$ dedop i remove -w workspace_name                         # CASE 3
$ dedop i remove -w workspace_name file_name1 file_name2   # CASE 4
```

In all cases, a confirmation prompt will appear to really ensure that this action is intentional. To disable this prompt, use the optional argument `--quiet` (eg. `dedop i remove --quiet`).

In **CASE 1**, all previously-added source files in the current workspace will be removed.

In **CASE 2**, the specified files in the current workspace will be removed.

In **CASE 3**, all previously-added source in the specified workspace will be removed.

In **CASE 4**, the specified files in the specified workspace will be removed.

`remove` has an alias `rm`.

List all L1A source files

To list all source files that have been added, run one of the following commands:

```
$ dedop i list                      # CASE 1
$ dedop i list -w other_workspace  # CASE 2
$ dedop i list L1A*                 # CASE 3
```

In **CASE 1**, the tool will return a list of all source files in the current workspace.

In **CASE 2**, the tool will return a list of all source files in the specified workspace.

In **CASE 3**, the tool will return a list of all source files that match the given regex `L1A*`.

`list` has an alias `ls`.

3.3.4 Processor Configuration Management

The next step before running an actual process is to manage the configurations.

Add a new configuration

To add a new configuration, run one of the following commands:

```
$ dedop c add new_config_name                      # CASE 1
$ dedop c add -w other_workspace new_config_name  # CASE 2
$ dedop c add --cryosat-adapted new_config_name   # CASE 3
```

In all cases, a new folder named `new_config_name` is created under a workspace and it consists of three default configuration files `CHD.json`, `CNF.json`, and `CST.json`. The generated configurations are by default for `Sentinel-3` processing unless when `--cryosat-adapted` is specified.

In **CASE 1**, a new configuration will be created under the current workspace directory.

In **CASE 2**, a new configuration will be created under the specified workspace directory.

In **CASE 3**, a new configuration suited for Adapted Cryosat-2 FBR data will be created under the current workspace directory.

Remove a configuration

To remove a configuration, run one of the following commands:

```
$ dedop c remove                                   # CASE 1
$ dedop c remove config_name                       # CASE 2
$ dedop c remove -w other_workspace config_name   # CASE 3
```

In all cases, a confirmation prompt will appear to really ensure that this action is intentional. To disable this prompt, use the optional argument `--yes` (eg. `dedop c remove --yes`). Removing a configuration means deleting a configuration folder including its contents (all the CHD, CNF, and CST files).

In **CASE 1**, the current configuration in the current workspace will be removed. It will then change the current configuration to the next configuration. When none left, it will go into a state where there are no current configurations.

In **CASE 2**, the specified configuration in the current workspace will be removed. There is no change of current configuration if it does not involve current configuration.

In **CASE 3**, the specified configuration inside a specified workspace will be removed.

`remove` has an alias `rm`.

Modify a configuration

To modify a configuration, run one of the following commands:

```
$ dedop c edit                                  # CASE 1
$ dedop c edit config_name                      # CASE 2
$ dedop c edit -w other_workspace config_name   # CASE 3
```

In all cases, it will launch a text editor and open all three configuration files. The text editor to be launched is OS-dependent and it is configurable on the *Tool Configuration* with the key name *launch_editor_command*.

In **CASE 1**, the text editor will open all the configuration files of the current configuration under the current workspace.

In **CASE 2**, the text editor will open all the configuration files of the specified configuration under the current workspace.

In **CASE 3**, the text editor will open all the configuration files of the specified configuration under the specified workspace.

When you are finished, just save the files and close the editor.

`edit` has an alias `ed`.

Copy a configuration

To copy a configuration, run one of the following commands:

```
$ dedop c copy                                                               # CASE 1
$ dedop c copy config_name_to_be_copied                                      # CASE 2
$ dedop c copy config_name_to_be_copied new_config_name                      # CASE 3
$ dedop c copy -w other_workspace config_name_to_be_copied new_config_name   # CASE 4
```

In **CASE 1**, neither the configuration to be copied nor the new configuration name is specified, so in this case a new name with the format of `current_config_name + _copy_ + unique_number` is created. The unique number will be incremented until a unique combination is constructed. For example, for a current config `config1`, it will create a new config named `config1_copy`, `config1_copy_2`, `config1_copy_3`, and so on.

In **CASE 2**, the configuration to be copied is specified, but not the name of the new config. It will then follow the same rule as in **CASE 1**, by creating a new config with a name that is of format `current_config_name + _copy_ + unique_number`.

In **CASE 3**, the specified configuration will be copied as `new_config_name` inside the current workspace

In **CASE 4**, the specified configuration will be copied as `new_config_name` inside the specified workspace

As in workspace management, copying a configuration does **NOT** automatically change the current configuration.

`copy` has an alias `cp`.

Rename a configuration

To rename a configuration, run one of the following commands:

```
$ dedop c rename new_config_name                                          # CASE 1
$ dedop c rename config_to_be_renamed new_config_name                     # CASE 2
$ dedop c rename -w other_workspace config_to_be_renamed new_config_name  # CASE 3
```

In **CASE 1**, the current config name will be renamed to `new_config_name`. The current configuration will also be changed to `new_config_name`.

In **CASE 2**, the specified config name in the current workspace will be renamed to `new_config_name`.

In **CASE 3**, the specified config name in the specified workspace will be renamed to `new_config_name`.

`rename` has an alias `rn`.

Show configuration info

To display information about the configuration such as current configuration path, list of files, as well as the file sizes, run the following command:

```
$ dedop c info                                  # CASE 1
$ dedop c info other_config                     # CASE 2
$ dedop c info -w other_workspace config_name   # CASE 3
```

In **CASE 1**, information for the current configuration in the current workspace will be displayed.

In **CASE 2**, information for the specified configuration in the current workspace will be displayed.

In **CASE 3**, information for the specified configuration in the specified workspace will be displayed.

`info` has an alias `i`.

Get current configuration

To get the current configuration name, run the following:

```
$ dedop c current
```

If there is a current configuration, the name of the current configuration will be returned. Otherwise, `no current DDP configuration` will be returned.

It is also possible to get the current configuration in the other workspace by adding this parameter `-w other_workspace_name` in the command.

`current` has an alias `cur`.

List configurations

To list available configurations, run one of the following commands:

```
$ dedop c list
```

As before, to list available configurations in the other workspace, just add `-w other_workspace_name` in the command.

`list` has an alias `ls`.

Upgrade configurations

A new version of DeDop Core sometimes comes with new versions of configuration files. In order to update your configurations, run the following command:

```
$ dedop c upgrade
```

Failure to use the latest version of configurations may result in processing errors.

`upgrade` has an alias `up`.

Show configuration version

To display the current configuration version, run the following command:

```
$ dedop c version
```

`version` has an alias `v`.

3.3.5 Running the Processor

Once the L1A source files have been added and configurations have been created, it is time to run the processing. To do that, use the following command:

```
$ dedop run
```

This command calls a processor to process L1A files to L1B (and possible L1BS). More information on how the processor works, go to here. By default, the command above will process every single L1A files inside the `inputs` directory under the current workspace, unless `--inputs [L1A_FILE [L1A_FILE ...]]` flag is specified.

The default behaviour is that the current configuration is used for processing. However, when `--all-configs` flag is set, it will process the input files with all available configurations in the current workspace, producing an output for each configuration. The output products will be located inside `outputs` directory under each configuration directory. To specify other locations for the outputs, the flag `--output DIR` can be used.

When the flag `--skip-l1bs` is added to the command above, the process will generate only L1B files.

3.3.6 Analyse the Results

After the processing has been finished, we can now compare the L1B outputs in an interactive Jupyter Notebook:

```
$ dedop output compare -C default L1B_myconf.nc L1B_default.nc
```

When you pass just file *names* to the `dedop output compare` command, DeDop must know to which configurations they refer to. The first filename corresponds to the *current* DDP configuration or the one given by the `-c` option. The second filename corresponds to a DDP configuration given by the `-C` (upper case!) option. You can also pass file *paths* to the `dedop output compare` command in which case the configuration names are ignored.

3.3.7 Tool Configuration

Configuration File

When DeDop is run for the first time it will create a file `config.py` in the directory `.dedop` of the current user's home directory. All DeDop tools use this file to read special software configuration parameters.

This is not to be confused with the *processor configurations* referred to in the dedicated section above.

Unixes and Darwin: On Unixes and Darwin (OS X), the full path to the DeDop tools configuration file is usually:

```
/home/<username>/.dedop/config.py
```

where `/home/<username>` is also given by ~ or `$HOME` in a terminal or shell.

Windows: On Windows 7+, the full path to the DeDop tools configuration file is usually:

```
C:/Users/<username>/.dedop/config.py
```

where `C:/Users/<username>` is also given by `%USERPROFILE%` on the Windows command-prompt.

To force writing a new DeDop tools configuration file use:

```
$ dedop --new-conf
```

This may be useful after DeDop software updates. It will ensure that you get the latest configuration parameters supported by a given DeDop version.

Configuration Parameters

Given here are the current DeDop tools configuration parameters:

Parameter name	Description	Default value
`workspaces_dir`	Path where the DeDop Shell stores your workspaces.	`'~/.dedop/workspaces'`
`launch_notebook_command`	An OS-specific shell command string used to launch a new Jupyter notebook server.	*OS-specific*
`launch_notebook_in_new_terminal`	Whether launching the notebook creates a new terminal window.	`False`
`launch_editor_command`	An OS-specific shell command string used to launch a text editor for the processor configuration files.	*OS-specific*

3.3.8 Command Reference

The following examples shall help you understand the basic concepts behind the various `dedop` commands.

Delay Doppler Altimeter Data (DeDop) command-line interface, version 1.5.1.dev1

```
usage: dedop [-h] [--version] [--traceback] [--license] [--docs] COMMAND ...
```

Positional Arguments

COMMAND	Possible choices: workspace, w, config, c, input, i, output, o, run, r, status, s, notebook, man, copyright, license

One of the following commands. Type "COMMAND -h" to get command-specific help.

Named Arguments

--version	show program's version number and exit
--traceback	show (Python) stack traceback for the last error Default: False
--license	show software license and exit Default: False
--docs	show software documentation in a browser window Default: False

Sub-commands:

workspace (w)

Manage DeDop workspaces.

```
dedop workspace [-h] COMMAND ...
```

Positional Arguments

COMMAND	Possible choices: add, remove, rm, copy, cp, rename, rn, current, cur, list, ls One of the following commands. Type "COMMAND -h" for help.

Sub-commands:

add

Add new workspace

```
dedop workspace add [-h] WORKSPACE
```

Positional Arguments

WORKSPACE	Name of the workspace

remove (rm)

Remove workspace

```
dedop workspace remove [-h] [-y] [WORKSPACE]
```

Positional Arguments

WORKSPACE	Name of the workspace

Named Arguments

-y, --yes	Do not ask for confirmation. Default: False

copy (cp)

Copy workspace

```
dedop workspace copy [-h] [WORKSPACE] [NEW_NAME]
```

Positional Arguments

WORKSPACE	Name of the workspace
NEW_NAME	Name of the new workspace

rename (rn)

Rename workspace

```
dedop workspace rename [-h] [WORKSPACE] NEW_NAME
```

Positional Arguments

WORKSPACE	Name of the workspace
NEW_NAME	New name of the workspace

current (cur)

Current workspace

```
dedop workspace current [-h] [WORKSPACE]
```

Positional Arguments

WORKSPACE	Name of the workspace

list (ls)

List workspaces

```
dedop workspace list [-h]
```

config (c)

Manage DeDop DDP configurations.

```
dedop config [-h] COMMAND ...
```

Positional Arguments

COMMAND	Possible choices: add, remove, rm, edit, ed, copy, cp, rename, rn, info, i, current, cur, list, ls, upgrade, up, version, v One of the following commands. Type "COMMAND -h" for help.

Sub-commands:

add

Add new DDP configuration

```
dedop config add [-h] [-w WORKSPACE] [--cryosat-adapted] CONFIG
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace
--cryosat-adapted	Initialise configuration with parameters for processing reconditioned Cryosat-2 FBR data Default: False

remove (rm)

Remove DDP configuration

```
dedop config remove [-h] [-y] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-y, --yes	Do not ask for confirmation. Default: False
-w, --workspace	Name of the workspace

edit (ed)

Edit DDP configuration

```
dedop config edit [-h] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

copy (cp)

Copy DDP configuration

```
dedop config copy [-h] [-w WORKSPACE] [CONFIG] [NEW_NAME]
```

Positional Arguments

CONFIG	Name of the DDP configuration
NEW_NAME	Name of the new DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

rename (rn)

Rename DDP configuration

```
dedop config rename [-h] [-w WORKSPACE] [CONFIG] NEW_NAME
```

Positional Arguments

CONFIG	Name of the DDP configuration
NEW_NAME	New name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

info (i)

Show DDP configuration info

```
dedop config info [-h] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

current (cur)

Current DDP configuration

```
dedop config current [-h] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

list (ls)

List DDP configurations

```
dedop config list [-h] [-w WORKSPACE]
```

Named Arguments

-w, --workspace	Name of the workspace

upgrade (up)

Upgrade DDP configurations to the newest versions

```
dedop config upgrade [-h] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

version (v)

Show DDP configuration versions

```
dedop config version [-h] [-w WORKSPACE] [CONFIG]
```

Positional Arguments

CONFIG	Name of the DDP configuration

Named Arguments

-w, --workspace	Name of the workspace

input (i)

Manage L1A inputs.

```
dedop input [-h] COMMAND ...
```

Positional Arguments

COMMAND	Possible choices: add, remove, rm, list, ls One of the following commands. Type "COMMAND -h" for help.

Sub-commands:

add

Add new inputs

```
dedop input add [-h] [-w WORKSPACE] [-q] L1A_FILE [L1A_FILE ...]
```

Positional Arguments

L1A_FILE	L1A input file to add to workspace.

Named Arguments

-w, --workspace	Name of the workspace
-q, --quiet	Suppress output of progress information. Default: False

remove (rm)

Remove inputs

```
dedop input remove [-h] [-w WORKSPACE] [-q] [L1A_FILE [L1A_FILE ...]]
```

Positional Arguments

L1A_FILE	L1A input file to add to workspace.

Named Arguments

-w, --workspace	Name of the workspace
-q, --quiet	Suppress output of progress information. Default: False

list (ls)

List inputs

```
dedop input list [-h] [-w WORKSPACE] [WC]
```

Positional Arguments

WC	Wildcard pattern.

Named Arguments

-w, --workspace	Name of the workspace

output (o)

Manage and analyse L1B outputs.

```
dedop output [-h] COMMAND ...
```

Positional Arguments

COMMAND	Possible choices: clean, cl, list, ls, open, op, inspect, ins, compare, cmp One of the following commands. Type “COMMAND -h” for help.

Sub-commands:

clean (cl)

Clean outputs folder of current configuration or CONFIG

```
dedop output clean [-h] [-w WORKSPACE] [-c CONFIG] [-q]
```

Named Arguments

-w, --workspace	Name of the workspace.
-c, --config	Name of the DDP configuration.
-q, --quiet	Suppress output of progress information. Default: False

list (ls)

List outputs folder of current configuration or CONFIG

```
dedop output list [-h] [-w WORKSPACE] [-c CONFIG] [WC]
```

Positional Arguments

WC	Wildcard pattern.

Named Arguments

-w, --workspace	Name of the workspace.
-c, --config	Name of the DDP configuration.

open (op)

Open outputs folder of current configuration or CONFIG

```
dedop output open [-h] [-w WORKSPACE] [-c CONFIG]
```

Named Arguments

-w, --workspace	Name of the workspace.
-c, --config	Name of the DDP configuration.

inspect (ins)

Inspect L1B product

```
dedop output inspect [-h] [-w WORKSPACE] [-c CONFIG] L1B_FILENAME
```

Positional Arguments

L1B_FILENAME	The filename or path of the a L1B product. If only a filename is given, it must exist in outputs of the given workspace/configuration.

Named Arguments

-w, --workspace	Name of the workspace.
-c, --config	Name of the DDP configuration.

compare (cmp)

Compare two L1B products

```
dedop output compare [-h] [-w WORKSPACE] [-c CONFIG] [-W WORKSPACE_2]
                     [-C CONFIG_2]
                     L1B_FILENAME_1 [L1B_FILENAME_2]
```

Positional Arguments

L1B_FILENAME_1	The filename or path of the first L1B product. If only a filename is given, it must exist in outputs of the given workspace/configuration.
L1B_FILENAME_2	The filename or path of the second L1B product. If only a filename is given, it must exist in outputs of the given second or first workspace/configuration. If omitted, the first filename or path is used.

Named Arguments

-w, --workspace	Name of the workspace.
-c, --config	Name of the DDP configuration.
-W, --workspace-2	The workspace of the second L1B product.
-C, --config-2	The DDP configuration of the second L1B product

run (r)

Run the Delay Doppler Processor (DDP).

```
dedop run [-h] [-s] [-q] [-w WORKSPACE] [-c CONFIG]
          [-i [L1A_FILE [L1A_FILE ...]]] [-o DIR] [-a]
```

Named Arguments

-s, --skip-l1bs	Skip generation of L1B-S files. Default: False
-q, --quiet	Suppress output of progress information. Default: False
-w, --workspace	Use WORKSPACE, defaults to current workspace.
-c, --config	Use CONFIG in workspace, defaults to current DDP configuration.
-i, --inputs	L1A input files. Defaults to all L1A files in workspace.
-o, --output	Alternative output directory.
-a, --all-configs	Run all DDP configurations in workspace. Cannot be used with option -c Default: False

status (s)

Print DeDop status information.

```
dedop status [-h] [-l]
```

Named Arguments

-l, --long	show extended status information Default: False

notebook

Open a Jupyter Notebook for DeDop.

```
dedop notebook [-h]
```

man

Open DeDop user manual in browser window.

```
dedop man [-h]
```

copyright

Print DeDop copyright information.

```
dedop copyright [-h]
```

license

Print DeDop license information.

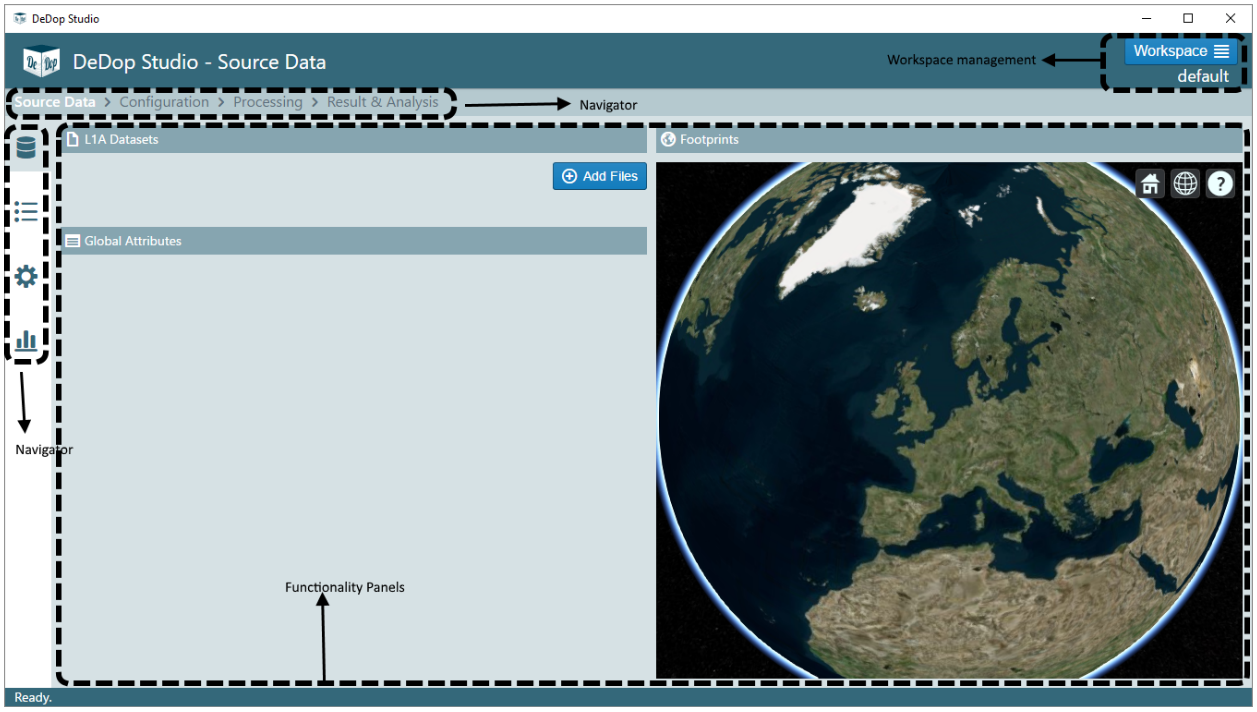
```
dedop license [-h]
```

3.4 DeDop Studio

DeDop Studio is a desktop application with a primary aim of providing a user-friendly experience for processing L1A files. It was designed based on the already-existing *DeDop Shell*, so the behavior is almost the same with its command line counterpart. In order to be able to use DeDop Studio, one must have a specific version of DeDop Core installed. The table below shows the dependency information between DeDop Core and DeDop Studio.

DeDop Studio	. . . depends on DeDop Core
v1.2.0	v1.2.0
v1.1.0	v1.1.0
v1.0.0	v1.0.0

The image below shows the first screen of DeDop Studio. At the top there is a screen title that will be updated once the user navigates to another screen. On the right side of this top section, there is a section for **Workspace management**. This is the area where user can perform workspace-related operations, such as adding, removing, copying, renaming, changing, and deleting a workspace. Right under the screen title, there are breadcrumbs showing the available four screens of this application. They also correspond to the four icons that are lined up vertically on the left side of the screen. These components act as the **navigator** of the application, to give the user an idea on where he/she is in the application, and also in the workflow. Both the workspace management and the navigator are components that are static, which means they will be visible on all the screens. **Functionality panels**, on the other hand, is the main area of the application where users can interact and it will change depending on which screen is selected.



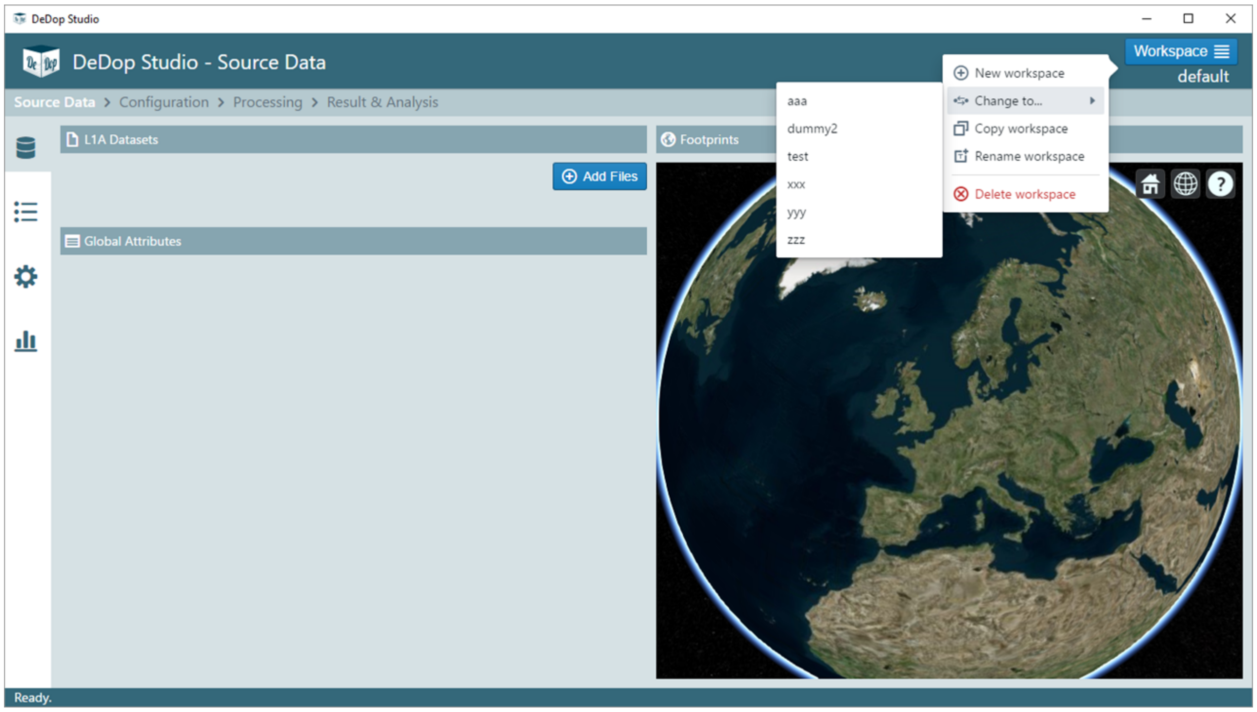
3.4.1 Workspace Management

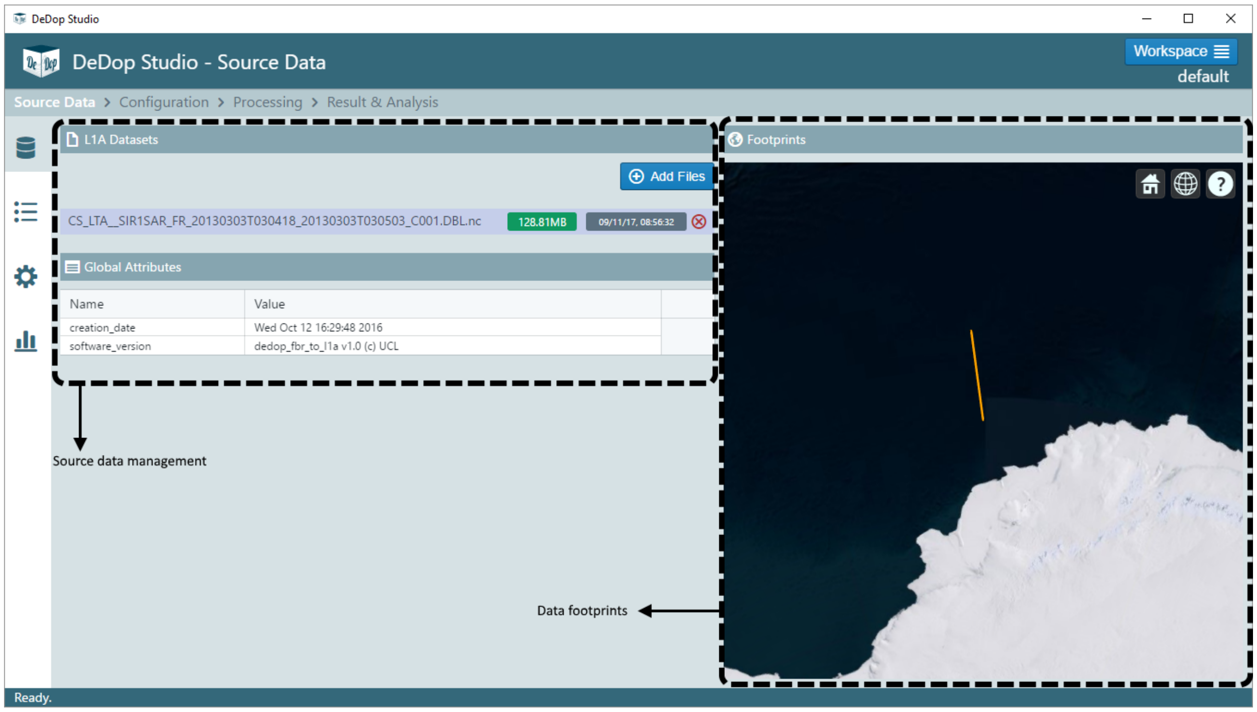
Clicking the blue Workspace button at the top right will display a workspace management popover menu. With this menu, one can perform workspace-related operations. Under this button, the current workspace name is displayed. For example, from the image above, we can see that the current workspace is **default**.

What happened behind the scene when any workspace-related operations (in fact, most of the operations) are triggered at DeDop Studio is that it makes some modifications on `.dedop` directory, as the case in DeDop Shell.

3.4.2 L1A Source File Management

On the **Source Data** screen, there are three panels: L1A Datasets, Global Attributes, and Footprints. In **L1A Datasets** panel, initially users can add an L1A file into the current workspace. Clicking the `Add Files` button will open a file selector window, using which the user can select a single file from the file system. When the adding process is finished, the newly-added file, together with all previously-added files, will be displayed as a list. When a single file is clicked, the **Global Attributes** panel will display the global attributes of the netCDF file. At the same time, **Footprints** panel will display the footprints of the selected netCDF file in a Cesium globe. Note that this display on the globe is only possible when the source file is a conformant L1A data.



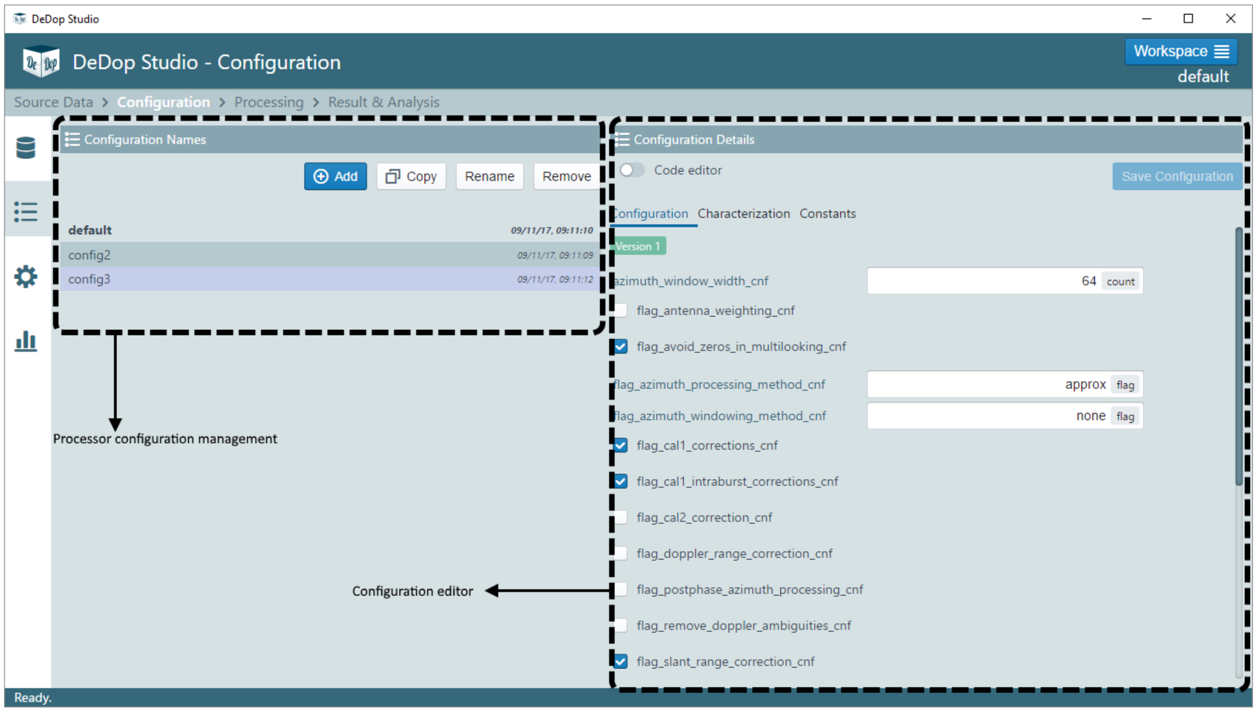


CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL.nc 128.81MB 09/11/17, 08:56:32

Name	Value
creation_date	Wed Oct 12 16:29:48 2016
software_version	dedop_fbr_to_l1a v1.0 (c) UCL

One more thing that happens when a file is selected is that, DeDop Studio stores this information and this will be selected as the default file (although still changeable) when a process is about to be started. This is explained in more detail in *Processing Screen*.

3.4.3 Processor Configuration Management



default	09/11/17, 09:11:10
config2	09/11/17, 09:11:09
config3	09/11/17, 09:11:12

azimuth_window_width_cnf	64 count
flag_antenna_weighting_cnf	☐
flag_avoid_zeros_in_multilooking_cnf	✓
flag_azimuth_processing_method_cnf	approx flag
flag_azimuth_windowing_method_cnf	none flag
flag_cal1_corrections_cnf	✓
flag_cal1_intraburst_corrections_cnf	✓
flag_cal2_correction_cnf	☐
flag_doppler_range_correction_cnf	☐
flag_postphase_azimuth_processing_cnf	☐
flag_remove_doppler_ambiguities_cnf	☐
flag_slant_range_correction_cnf	✓

There are two panels in **Configuration** screen: Configuration Names and Configuration Details. In **Configuration Names** panel, users can do configuration-related operations such as add, copy, rename, remove, or change the current configuration. Single-clicking a configuration name on the list will display the contents of that particular configuration on **Configuration Details** panel. To select a configuration as the current configuration, double-click it. Another way is to select it from the drop-down list in **Run Settings** panel inside *Processing Screen*.

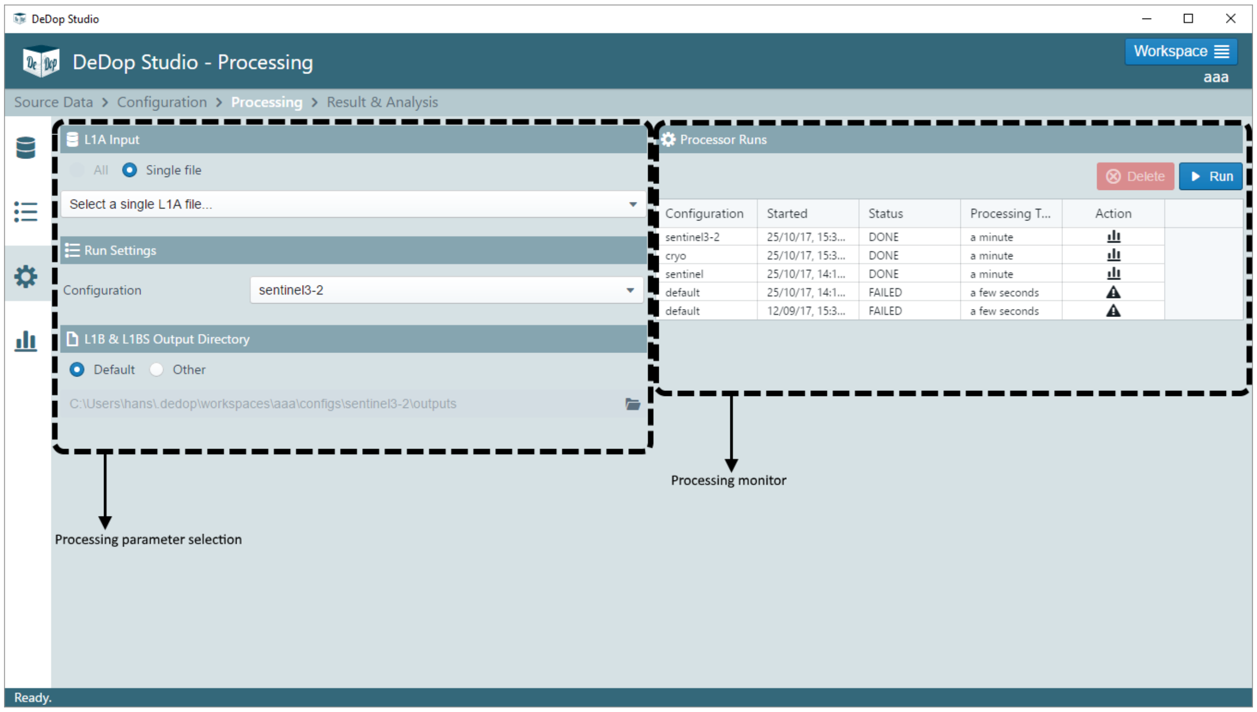
In **Configuration Details** panel, users can modify the configuration files in two ways: using HTML components or using code editor. In HTML mode, configuration items are listed in a form appearance either as a label-texinput (when the expected value is of type number or string) or a checkbox (when the expected value is of type boolean). When the code editor mode is active (by activating `Code editor` toggle at the top), a JSON code editor will appear. Here, users can modify the values of the configurations in JSON format.

In both modes, as soon as a value has been changed, the `Save Configuration` button is enabled. Click this button if the change(s) is to be made, otherwise this change(s) will not be reflected on the actual configuration files.

By default (in DeDop Studio v1.2.0), the modification of `Characterization` and `Constants` configurations are disabled. You will notice that under these tabs, the HTML elements are disabled and the Code Editor is read-only.

3.4.4 Running the Processor

There are 4 panels under **Processing** screens: L1A Input, Run Settings, L1B & L1BS Output Directory, and Processor Runs. The first two panels form a summary on which L1A dataset and which configuration the users have chosen on **Source Data** and **Configuration** screens, respectively. A small note on **L1A Input** panel: at the moment only


DeDop Studio
DeDop Studio - Processing
Workspace
aaa
Source Data > Configuration > Processing > Result & Analysis
L1A Input
All
Single file
Run Settings
L1B & L1BS Output Directory
Default
Other
Processor Runs
Delete
Run
Processing monitor
Processing parameter selection
Ready.

Select a single L1A file...

Configuration: sentinel3-2

C:\Users\hans\.dedop\workspaces\aaa\configs\sentinel3-2\outputs

Configuration	Started	Status	Processing T...	Action
sentinel3-2	25/10/17, 15:3...	DONE	a minute	📊
cryo	25/10/17, 15:3...	DONE	a minute	📊
sentinel	25/10/17, 14:1...	DONE	a minute	📊
default	25/10/17, 14:1...	FAILED	a few seconds	⚠
default	12/09/17, 15:3...	FAILED	a few seconds	⚠

processing of single files are available. In **L1B & L1BS Output Directory**, users can select another output directory for the product results.

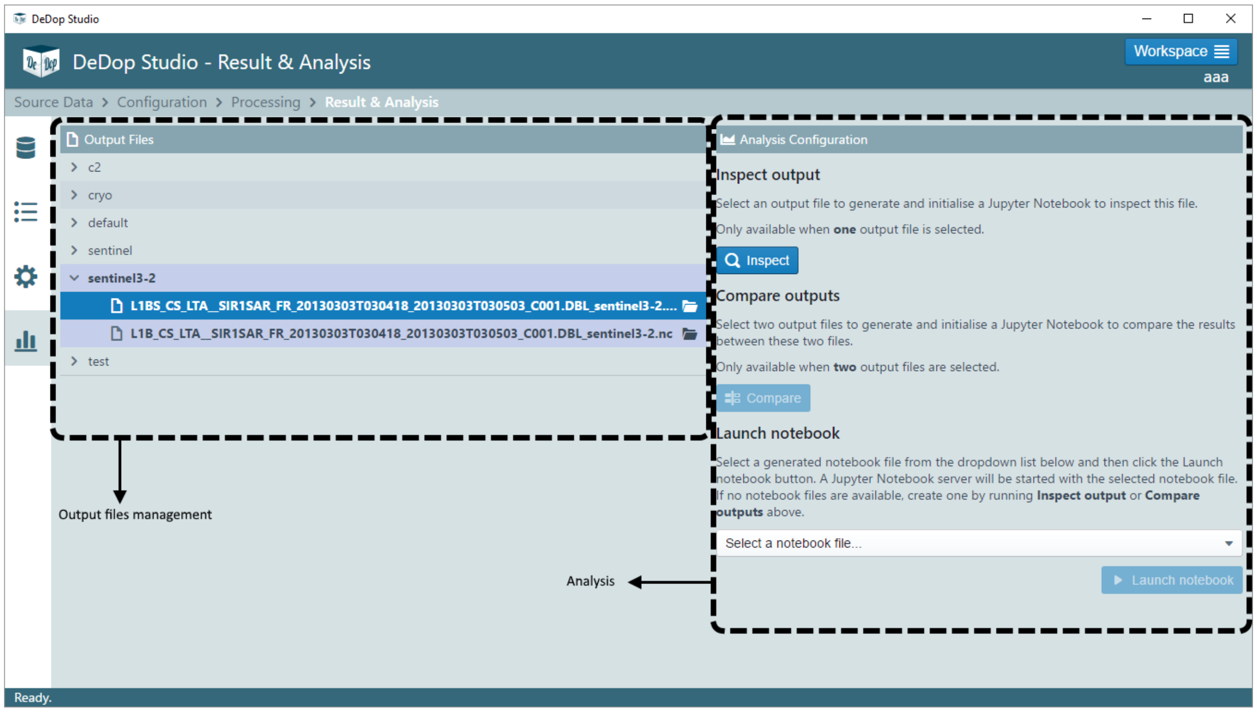
To initiate a processing, click `Run` button inside **Processor Runs** panel. DeDop Studio will check whether all the required parameters (input file, configuration, and output directory) have been selected. When any of them are missing, a dialog box will pop up with the information on which field you have to fill up. Otherwise, the processing will be started, marked by a new entry on the table. You can monitor the progress of the processing and at the moment, because the tool is capable only running one process at a time, the `Run` button is blocked, displayed as a spinner, as long as a process is running.

In the end, a process can either be successful or fail. This status is indicated by the icon under `Action` column on the table in **Processor Runs** panel. On the image above, for example, the first three rows indicate that the processes are successful and when icon is clicked, the page will transition to screen 4, *Result & Analysis*. In the case of failure, mouse over on the icon to show a short description of the error as a tooltip text.

When DeDop Studio is closed, the information in the table is preserved, by storing the data into `dedop-prefs.json`. During the next startup of DeDop Studio, this information is loaded and used to populate the table.

3.4.5 Analysing L1B Results

The purpose of this screen is to manage what to do with the result products after a processing. There are two panels: Output Files and Analysis Configuration. In **Output Files**, users can navigate to different output files in the current workspace directory, grouped by configurations. By clicking a file name, the said file is selected. This has an implication on which actions are available, depending on how many files are selected. If only one file is selected, the `Inspect` button on **Analysis Configuration** panel is enabled. When two files are selected, the `Compare` button is enabled. Clicking one of these buttons will trigger a creation of a Jupyter notebook suitable for inspecting or comparing the file(s) that have been selected, and to start a local instance of Jupyter notebook server. This behavior is consistent with DeDop Shell command *dedop output compare/inspect*. The dropdown list at the bottom of **Analysis Configuration** is used to select a notebook file that has been previously created through `Inspect` and `Compare` buttons.


DeDop Studio
DeDop Studio - Result & Analysis
Workspace
aaa
Source Data > Configuration > Processing > Result & Analysis
Output Files
c2
cryo
default
sentinel
sentinel3-2
L1BS_CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL_sentinel3-2....
L1B_CS_LTA__SIR1SAR_FR_20130303T030418_20130303T030503_C001.DBL_sentinel3-2.nc
test
Output files management
Analysis Configuration
Inspect
Compare
Select a notebook file...
Launch notebook
Analysis
Ready.

Inspect output

Select an output file to generate and initialise a Jupyter Notebook to inspect this file.

Only available when **one** output file is selected.

Compare outputs

Select two output files to generate and initialise a Jupyter Notebook to compare the results between these two files.

Only available when **two** output files are selected.

Launch notebook

Select a generated notebook file from the dropdown list below and then click the Launch notebook button. A Jupyter Notebook server will be started with the selected notebook file. If no notebook files are available, create one by running **Inspect output** or **Compare outputs** above.

3.5 Auxilliary Files

3.5.1 Configuration Parameters

The configuration of the processor is controlled via the Configuration file (CST.json). This file is a JSON document that contains several parameters to enable, disable, or otherwise control various features of the processor. The CNF.json file can be editted either via the Dedop Studio interface, or from the command-line interface with the command `dedop config edit`.

This page contains descriptions of the configuration parameters, their purposes and possible values.

Corrections Flags

- **flag_cal2_correction_cnf**
  - value: boolean [truelfalse]
  - description: Flag that activates the CAL2 corrections: Deactivated (false); Activated (true)
- **flag_uso_correction_cnf**
  - value: boolean [truelfalse]
  - description: Flag that activates the USO correction: Deactivated (false); Activated (true)
- **flag_cal1_corrections_cnf**
  - value: boolean [truelfalse]
  - description: Flag that activates the CAL1 corrections: Deactivated (false); Activated (true)
- **flag_cal1_intraburst_corrections_cnf**
  - value: boolean [truelfalse]

- description: Flag that activates the CAL1 intraburst corrections: Deactivated (false); Activated (true)

Surface Focusing

- **flag_surface_focusing_cnf**
  - value: boolean [true|false]
  - description: Flag that activates the surface focussing: Deactivated (false); Activated (true)
- **surface_focusing_lat_cnf**
  - value: float [-90, 90]
  - units: degrees north
  - description: Location of the surface focusing target (latitude) (ignored unless flag_surface_focusing_cnf is ‘true’)
- **surface_focusing_lon_cnf**
  - value: float [-180, 180]
  - units: degrees east
  - description: Location of the surface focusing target (longitude) (ignored unless flag_surface_focusing_cnf is ‘true’)
- **surface_focusing_alt_cnf**
  - value: float
  - units: metres
  - description: Location of the surface focusing target (altitude) (ignored unless flag_surface_focusing_cnf is ‘true’)

Azimuth Processing

- **flag_azimuth_processing_method_cnf**
  - value: string [‘approx’|’exact’]
  - description: Flag that indicates the azimuth processing method: Approximate (‘approx’); Exact (‘exact’)
- **flag_postphase_azimuth_processing_cnf**
  - value: boolean [true|false]
  - description: Flag that enables the post-phase azimuth processing: Deactivated (false); Activated (true)
- **flag_azimuth_windowing_method_cnf**
  - value: string [‘none’|’boxcar’|’hamming’|’hanning’]
  - description: Flag the sets the azimuth windowing method: Disabled (‘none’); Boxcar (‘boxcar’); Hamming (‘hamming’); Hanning (‘hanning’)
- **azimuth_window_width_cnf**
  - value: integer [32, 64]
  - description: Width of Azimuth window (minimum value: 32, maximum value: 64)

Geometry Corrections

- **flag_doppler_range_correction_cnf**
  - value: boolean [true|false]
  - description: Flag that activates the Doppler range correction in the geometry corrections: Deactivated (false); Activated (true)
- **flag_slant_range_correction_cnf**
  - value: boolean [true|false]
  - description: Flag that activates the slant range correction in the geometry corrections: Deactivated (false); Activated (true)
- **flag_window_delay_alignment_method_cnf**
  - value: integer [0|1|2|3|4]
  - description: Flag to indicate the window delay alignment method: Surface dependent (0); Beam max integrated power (1); Satellite position above surface (2); Look angle 0 (3); Doppler angle 0 (4)

Stack Masking

- **flag_stack_masking**
  - value: boolean [true|false]
  - description: Flag that activates the Stack Masking algorithm: Activated (true); Deactivated (false)
- **flag_remove_doppler_ambiguities**
  - value: boolean [true|false]
  - description: Flag that indicates if the Doppler ambiguities will be removed: No (false); Yes (true)

Multilooking

- **flag_avoid_zeros_in_multilooking**
  - value: boolean [true|false]
  - description: Flag that indicates if the samples set to zero in the beams will be avoided when averaging in multi-looking: No (false); Yes (true)
- **flag_surface_weighting**
  - value: boolean [true|false]
  - description: Flag that activates the surface weighting: Deactivated (false); Activated (true)
- **flag_antenna_weighting**
  - value: boolean [true|false]
  - description: Flag that activates the antenna weighting: Deactivated (false); Activated (true)

General

- **zp_fact_range**
  - value: integer
  - description: Zero padding factor used during range compression
- **n_looks_stack**
  - value: integer
  - description: Maximum number of looks in 1 stack

Region of Interest

NB: these parameters are optional, any or all of them may be omitted from the configuration file. If you do not wish to apply a Region-of-Interest (RoI) filter, you should not include these parameters in the configuration file. If you wish to enable the RoI filter, add whichever parameters you intend to use to describe the RoI.

- **min_lat**
  - value: float [-90, 90]
  - units: degrees north
  - description: minimum latitude beneath which input records will be excluded
- **max_lat**
  - value: float [-90, 90]
  - units: degrees north
  - description: maximum latitude above which input records will be excluded
- **min_lon**
  - value: float [-180, 180]
  - units: degrees east
  - description: minimum longitude beneath which input records will be excluded
- **max_lon**
  - value: float [-180, 180]
  - units: degrees east
  - description: maximum longitude above which input records will be excluded

3.5.2 Characterisation Parameters

The Characterisation file (CHD.json) describes the parameters of the instrument from which the input data being processed have been obtained. the default characterisation parameter values are suitable for processing data from Sentinel-3. In general, it should not be nessicary to edit this file. If you wish to process Cryosat-2 data that has been adapted into the Sentinel-3 L1A format, a suitable set of values can be created either through the *Dedop Studio* interface, or by adding the `--cryosat-adapted` flag to the `dedop config add` command.

Parameter Descriptions

- **mean_sat_alt_chd**
  - value: float
  - units: metres
  - description: mean altitude of the satellite
- **N_samples_sar_chd**
  - value: integer
  - description: number of samples per each SAR pulse
- **N_ku_pulses_burst_chd**
  - value: integer
  - description: number of ku-band pulses per burst
- **freq_ku_chd**
  - value: float
  - units: Hz
  - description: emitted frequency in Ku-band
- **pulse_length_chd**
  - value: float
  - units: s
  - description: pulse length
- **bw_ku_chd**
  - value: float
  - units: Hz
  - description: Ku-band bandwidth
- **power_tx_ant_ku_chd**
  - value: float
  - units: dB
  - description: Antenna SSPA RF Peak Transmitted Power in Ku band
- **antenna_gain_ku_chd**
  - value: float
  - units: dB
  - description: Antenna gain for Ku-band
- **uso_freq_nom_chd**
  - value: float
  - units: Hz
  - description: USO nominal frequency
- **alt_freq_multiplier_chd**

  - value: float
  - description: Factor to convert from USO frequency to altimeter frequency
- **prf_sar_chd**
  - value: float
  - units: Hz
  - description: pulse repetition frequency
- **brf_sar_chd**
  - value: float
  - units: Hz
  - description: burst repetition frequency
- **antenna_weights_chd**
  - value: float array(250)
  - description: array of antenna weights
- **antenna_angles_chd**
  - value: float array(250)
  - units: radians
  - description: array of antenna angles
- **antenna_angles_spacing_chd**
  - value: float
  - units: radians
  - description: spacing between antenna angles

NB: the parameters `antenna_weights_chd`, `antenna_angles_chd`, and `antenna_angles_spacing_chd` are optional, and do not need to be included in the CHD file unless the `flag_antenna_wieghting_cnf` flag in the CNF file is set to `true`.

3.5.3 Constants Parameters

The Constants file (CST.json) contains the values of various physical constants required by the Dedop processor. Although this file can be editted either via the Dedop Studio interface or with the command `dedop config edit`, it should generally not be nessicary to do so.

Constants

- **c_cst**
  - value: float
  - description: speed of light
- **pi_cst**
  - value: float
  - description: Pi number

- **semi_major_axis_cst**
  - value: float
  - description: semi-major axis of WGS84 ellipsoid
- **semi_minor_axis_cst**
  - value: float
  - description: semi-minor axis of WGS84 ellipsoid
- **earth_radius_cst**
  - value: float
  - description: radius of the earth
- **flat_coeff_cst**
  - value: float
  - description: flattening coefficient of the WGS84 ellipsoid
- **sec_in_day_cst**
  - value: float
  - description: number of seconds in a day

3.6 Input/Output Files

3.6.1 L1A Input Files

File Description

The Dedop processor reads Level 1A Sentinel-3 format data files. These files are in the netCDF4 (.nc) format, and must have the correct dimensions and variables as described in this page. If the format of the input file being processed does not match the format described here, Dedop may be unable to read the file.

Input files can be added to a workspace via the command `dedop input add`, or passed directly to the processor at run-time by using the `-i` parameter of the `dedop run` command.

Dimensions

- **echo_sample_ind**
  - description: the number of samples per echo
- **sar_ku_pulse_burst_ind**
  - description: the number of Ku-band pulses in a SAR burst
- **sar_c_pulse_burst_ind**
  - description: the number of C-band pulses in a SAR burst
- **ltm_max_ind**
  - description: Maximum number of LTM Cal1 or Cal2 tables
- **time_l1a_echo_sar_ku**

  - description: the along-track time of SAR bursts

- **time_l1a_echo_plrm**
  - description: the along-track time of PLRM bursts

Variables

- **echo_sample_ind**
  - description: the echo sample index
- **sar_ku_pulse_burst_ind**
  - description: the SAR Ku-band burst index
- **sar_c_pulse_burst_ind**
  - description: the SAR C-band burst index
- **ltm_max_ind**
  - description: Maximum number of LTM Cal1 or Cal2 tables
- **time_l1a_echo_sar_ku**
  - description: the time of the SAR Ku-band burst
- **UTC_day_l1a_echo_sar_ku**
  - description: the UTC day number of the burst
- **UTC_sec_l1a_echo_sar_ku**
  - description: the number of seconds of the current UTC day of the burst
- **UTC_time_20hz_l1a_echo_sar_ku**
  - description: UTC of the 20 Hz measurement
- **isp_coarse_time_l1a_echo_sar_ku**
  - description: ISP coarse time
- **isp_fine_time_l1a_echo_sar_ku**
  - description: ISP fine time
- **flag_time_status_l1a_echo_sar_ku**
  - description: time status flag
- **sral_fine_time_l1a_echo_sar_ku**
  - description: ISP SRAL fine datation
- **lat_l1a_echo_sar_ku**
  - description: the latitude (degrees north) for the current burst
- **lon_l1a_echo_sar_ku**
  - description: the longitude (degrees east) for the corrent burst
- **surf_type_l1a_echo_sar_ku**
  - description: the surface type flag
- **burst_count_prod_l1a_echo_sar_ku**

  - description: bursts counter within the product
- **seq_count_l1a_echo_sar_ku**
  - description: the source sequence counter index
- **burst_count_cycle_l1a_echo_sar_ku**
  - description: bursts counter within the tracking cycle
- **nav_bul_status_l1a_echo_sar_ku**
  - description: navigation bulletin status
- **nav_bul_source_l1a_echo_sar_ku**
  - description: navigation bulletin source identifier
- **oper_instr_l1a_echo_sar_ku**
  - description: the instrument flag
- **SAR_mode_l1a_echo_sar_ku**
  - description: SAR mode identifier
- **cl_gain_l1a_echo_sar_ku**
  - description: tracking configuration - closed loop gain
- **acq_stat_l1a_echo_sar_ku**
  - description: tracking configuration - acquisition status
- **dem_eeprom_l1a_echo_sar_ku**
  - description: tracking configuration - DEM EEPROM read access
- **weighting_l1a_echo_sar_ku**
  - description: altimeter configuration - weighting function
- **loss_track_l1a_echo_sar_ku**
  - description: loss of track criterion
- **h0_nav_dem_l1a_echo_sar_ku**
  - description: altitude command H0 computed with nav DEM
- **h0_applied_l1a_echo_sar_ku**
  - description: applied altitude command H0
- **cor2_nav_dem_l1a_echo_sar_ku**
  - description: altitude command COR2 computed with nav DEM
- **cor2_applied_l1a_echo_sar_ku**
  - description: applied altitude command COR2
- **dh0_l1a_echo_sar_ku**
  - description: distance error computed on the echo of the cycle (N-2) in open loop mode
- **agccode_ku_l1a_echo_sar_ku**
  - description: AGCCODE for ku band
- **agccode_c_l1a_echo_sar_ku**

  - description: AGCCODE for c band
- **alt_l1a_echo_sar_ku**
  - description: the altitude of the current burst
- **orb_alt_rate_l1a_echo_sar_ku**
  - description: the altitude rate of the current burst
- **x_pos_l1a_echo_sar_ku**
  - description: the ECEF x-coordinate of the current burst
- **y_pos_l1a_echo_sar_ku**
  - description: the ECEF y-coordinate of the current burst
- **z_pos_l1a_echo_sar_ku**
  - description: the ECEF z-coordinate of the current burst
- **x_vel_l1a_echo_sar_ku**
  - description: the ECEF x-velocity of the current burst
- **y_vel_l1a_echo_sar_ku**
  - description: the ECEF y-velocity of the current burst
- **z_vel_l1a_echo_sar_ku**
  - description: the ECEF z-velocity of the current burst
- **roll_sat_pointing_l1a_echo_sar_ku**
  - description: the roll (degrees) of the satellite at the current burst
- **pitch_sat_pointing_l1a_echo_sar_ku**
  - description: the pitch (degrees) of the satellite at the current burst
- **yaw_sat_pointing_l1a_echo_sar_ku**
  - description: the yaw (degrees) of the satellite at the current burst
- **roll_sral_mispointing_l1a_echo_sar_ku**
  - description: SRAL mispointing angle - roll
- **pitch_sral_mispointing_l1a_echo_sar_ku**
  - description: SRAL mispointing angle - pitch
- **yaw_sral_mispointing_l1a_echo_sar_ku**
  - description: SRAL mispointing angle - yaw
- **range_ku_l1a_echo_sar_ku**
  - description: the receiving window range
- **int_path_cor_ku_l1a_echo_sar_ku**
  - description: the internal path correction
- **uso_cor_l1a_echo_sar_ku**
  - description: the USO correction
- **cog_cor_l1a_echo_sar_ku**

  - description: the Centre-of-Gravity correction
- **agc_ku_l1a_echo_sar_ku**
  - description: the AGC correction
- **scale_factor_ku_l1a_echo_sar_ku**
  - description: the sigma-0 scaling factor
- **sig0_cal_ku_l1a_echo_sar_ku**
  - description: internal calibration correction on Sigma0 for ku band
- **i_meas_ku_l1a_echo_sar_ku**
  - description: the i-component of the measured waveform
- **q_meas_ku_l1a_echo_sar_ku**
  - description: the q-component of the measured waveform
- **gprw_meas_ku_l1a_echo_sar_ku**
  - description: ku band samples of the normalized GPRW (cal2)
- **cal2_ku_ind_l1a_echo_sar_ku**
  - description: the CAL2 index
- **burst_power_cor_ku_l1a_echo_sar_ku**
  - description: the power of the burst
- **burst_phase_cor_ku_l1a_echo_sar_ku**
  - description: the phase of the burst
- **cal1_ku_ind_l1a_echo_sar_ku**
  - description: the CAL1 index

3.6.2 L1B Output Files

File Description

The L1B file is the primary output created by the dedop processor. Each record in the L1B file represents information about a surface position beneath the satellite track.

Dimensions

- **time_l1b_echo_sar_ku**
  - description: Number of L1B ECHO_SAR_Ku measurements
- **echo_sample_ind**
  - description: Number of samples in a waveform
- **max_multi_stack_ind**
  - description: Maximum number of multilook beams per stack

Variables

- **time_l1b_echo_sar_ku**
  - description: UTC Seconds since 2000-01-01 00:00:00.0
- **UTC_day_l1b_echo_sar_ku**
  - description: UTC Days since 2000-01-01 00:00:00.0
- **UTC_sec_l1b_echo_sar_ku**
  - description: Seconds since the start of the current UTC day
- **GPS_time_l1b_echo_sar_ku**
  - description: GPS time
- **isp_coarse_time_l1b_echo_sar_ku**
  - description: ISP coarse time
- **isp_fine_time_l1b_echo_sar_ku**
  - description: ISP fine time
- **sral_fine_time_l1b_echo_sar_ku**
  - description: ISP SRAL fine datation
- **lat_l1b_echo_sar_ku**
  - description: latitude (degrees north)
- **lon_l1b_echo_sar_ku**
  - description: longitude (degrees east)
- **alt_l1b_echo_sar_ku**
  - description: altitude of satellite
- **orb_alt_rate_l1b_echo_sar_ku**
  - description: orbital altitude rate
- **flag_time_status_l1b_echo_sar_ku**
  - description: time status flag
- **time_time_corr_val_l1b_echo_sar_ku**
  - description: time correlation validity flag
- **flag_man_pres_l1b_echo_sar_ku**
  - description: manoeuvre presence flag (no manoeuvre/ongoing manoeuvre)
- **flag_man_thrust_l1b_echo_sar_ku**
  - description: manoeuvre thrust flag (no thrust/ongoing thrust)
- **flag_man_plane_l1b_echo_sar_ku**
  - description: manoeuvre plane flag (in plane/out of plane)
- **flag_gnss_status_l1b_echo_sar_ku**
  - description: validity flag for the navigation message from the gnss receiver
- **x_pos_l1b_echo_sar_ku**

  - description: satellite altitude - x component
- **y_pos_l1b_echo_sar_ku**
  - description: satellite altitude - y component
- **z_pos_l1b_echo_sar_ku**
  - description: satellite altitude - z component
- **x_vel_l1b_echo_sar_ku**
  - description: satellite velocity - x component
- **y_vel_l1b_echo_sar_ku**
  - description: satellite velocity - y component
- **z_vel_l1b_echo_sar_ku**
  - description: satellite velocity - z component
- **nav_bul_status_l1b_echo_sar_ku**
  - description: navigation bulletin status (ok/ko)
- **nav_bul_source_l1b_echo_sar_ku**
  - description: navigation bulletin source identifier (gps/doris)
- **nav_bul_coarse_time_l1b_echo_sar_ku**
  - description: navigation bulletin coarse time
- **nav_bul_fine_time_l1b_echo_sar_ku**
  - description: navigation bulletin fine time
- **seq_count_l1b_echo_sar_ku**
  - description: sequence count
- **isp_time_status_echo_sar_ku**
  - description: ISP time status
- **oper_instr_l1b_echo_sar_ku**
  - description: operating instrument
- **SAR_mode_l1b_echo_sar_ku**
  - description: LRM mode identifier
- **cl_gain_l1b_echo_sar_ku**
  - description: tracking configuration - closed loop gain
- **acq_stat_l1b_echo_sar_ku**
  - description: tracking configuration - acquisition status
- **dem_eeprom_l1b_echo_sar_ku**
  - description: tracking configuration - DEM EEPROM read access
- **weighting_l1b_echo_sar_ku**
  - description: altimeter configuration - weighting function
- **loss_track_l1b_echo_sar_ku**

  - description: loss of track criterion
- **h0_nav_dem_l1b_echo_sar_ku**
  - description: altitude command H0 computed with nav DEM
- **h0_applied_l1b_echo_sar_ku**
  - description: applied altitude command H0
- **cor2_nav_dem_l1b_echo_sar_ku**
  - description: altitude command COR2 computed with nav DEM
- **cor2_applied_l1b_echo_sar_ku**
  - description: applied altitude command COR2
- **dh0_l1b_echo_sar_ku**
  - description: distance error computed on the echo of the cycle (N-2) in open loop mode
- **agccode_ku_l1b_echo_sar_ku**
  - description: AGCCODE for ku band
- **surf_type_l1b_echo_sar_ku**
  - description: altimeter surface type
- **range_ku_l1b_echo_sar_ku**
  - description: corrected range for ku band
- **uso_cor_l1b_echo_sar_ku**
  - description: USO frequency drift correction
- **int_path_cor_ku_l1b_echo_sar_ku**
  - description: internal path correction for ku band
- **range_rate_l1b_echo_sar_ku**
  - description: range rate
- **agc_ku_l1b_echo_sar_ku**
  - description: corrected AGC for ku band
- **scale_factor_ku_l1b_echo_sar_ku**
  - description: scaling factor for sigma0 evaluation for ku band
- **agc_cor_ku_l1b_echo_sar_ku**
  - description: correction for instrumental errors on AGC for ku band
- **sig0_cal_ku_l1b_echo_sar_ku**
  - description: internal calibration correction on Sigma0 for ku band
- **nb_stack_l1b_echo_sar_ku**
  - description: number of waveforms summed in stack
- **max_stack_l1b_echo_sar_ku**
  - description: maximum power of stack
- **stdev_stack_l1b_echo_sar_ku**

  - description: standard deviation of stack

- **skew_stack_l1b_echo_sar_ku**
  - description: skewness of stack
- **kurt_stack_l1b_echo_sar_ku**
  - description: kurtosis of stack
- **beam_ang_l1b_echo_sar_ku**
  - description: look angles in stack
- **beam_form_l1b_echo_sar_ku**
  - description: flag on beam formation quality in stack
- **i2q2_meas_ku_l1b_echo_sar_ku**
  - description: I2+Q2 measurement for ku band

3.6.3 L1B-S Output Files

File Description

In addition to the L1B file, the dedop processor can also optionally write an L1B-S output file. This file contains much of the same information as the L1B file, but is extended to include stack-level (per beam) data. The processor will produce an L1B-S file by default, but due to the large size of the file you may wish to dissable this. To prevent the processor from creating an L1B-S file, add the `--skip-l1bs` argument to the `dedop run` command.

Dimensions

- **time_l1bs_echo_sar_ku**
  - description: Number of L1Bs ECHO_SAR_Ku measurements
- **echo_sample_ind**
  - description: Number of samples in a waveform
- **max_multi_stack_ind**
  - description: Maximum number of multilook beams per stack

Variables

- **time_l1bs_echo_sar_ku**
  - description: UTC Seconds since 2000-01-01 00:00:00.0
- **UTC_day_l1bs_echo_sar_ku**
  - description: UTC Days since 2000-01-01 00:00:00.0
- **UTC_sec_l1bs_echo_sar_ku**
  - description: Seconds since the start of the current UTC day
- **lat_l1bs_echo_sar_ku**
  - description: Surface location (degrees North)

- **lon_l1bs_echo_sar_ku**
    - description: Surface location (degrees East)
- **surf_type_l1bs_echo_sar_ku**
    - description: Altimeter surface type (open ocean or semi-enclosed seas/enclosed seas/enclosed seas or lakes/continental ice/land)
- **records_count_l1bs_echo_sar_ku**
    - description: Record index
- **alt_l1bs_echo_sar_ku**
    - description: Altitude of surface (m)
- **orb_alt_rate_l1bs_echo_sar_ku**
    - description: Orbital altitude rate (m/s)
- **x_pos_l1bs_echo_sar_ku**
    - description: Satellite ECEF x-coordinate
- **y_pos_l1bs_echo_sar_ku**
    - description: Satellite ECEF y-coordinate
- **z_pos_l1bs_echo_sar_ku**
    - description: Satellite ECEF z-coordinate
- **x_vel_l1bs_echo_sar_ku**
    - description: Satellite ECEF x-velocity
- **y_vel_l1bs_echo_sar_ku**
    - description: Satellite ECEF y-velocity
- **z_vel_l1bs_echo_sar_ku**
    - description: Satellite ECEF z-velocity
- **meas_x_pos_l1bs_echo_sar_ku**
    - description: Surface location - ECEF x-coordinate
- **meas_y_pos_l1bs_echo_sar_ku**
    - description: Surface location - ECEF y-coordinate
- **meas_z_pos_l1bs_echo_sar_ku**
    - description: Surface location - ECEF z-coordinate
- **roll_sat_pointing_l1bs_echo_sar_ku**
    - description: Satellite pointing angle - roll (degrees)
- **pitch_sat_pointing_l1bs_echo_sar_ku**
    - description: Satellite pointing angle - pitch (degrees)
- **yaw_sat_pointing_l1bs_echo_sar_ku**
    - description: Satellite pointing angle - yaw (degrees)
- **roll_sral_mispointing_l1bs_echo_sar_ku**

  - description: SRAL mispointing angle - roll (degrees)
- **pitch_sral_mispointing_l1bs_echo_sar_ku**
  - description: SRAL mispointing angle - pitch (degrees)
- **yaw_sral_mispointing_l1bs_echo_sar_ku**
  - description: SRAL mispointing angle - yaw (degrees)
- **range_ku_l1bs_echo_sar_ku**
  - description: Corrected range for ku band (m)
- **int_path_cor_ku_l1bs_echo_sar_ku**
  - description: Internal path correction for ku band
- **uso_cor_l1bs_echo_sar_ku**
  - description: USO frequency drift correction
- **cog_cor_l1bs_echo_sar_ku**
  - description: Distance antenna-CoG correction
- **agccode_ku_l1bs_echo_sar_ku**
  - description: AGCCODE for ku band
- **agc_ku_l1bs_echo_sar_ku**
  - description: corrected AGC for ku band
- **scale_factor_ku_l1bs_echo_sar_ku**
  - description: scaling factor for sigma0 evaluation for ku band
- **sig0_cal_ku_l1bs_echo_sar_ku**
  - description: internal calibration correction on Sigma0 for ku band
- **snr_ku_l1bs_echo_sar_ku**
  - description: snr estimation for ku band
- **i2q2_meas_ku_l1bs_echo_sar_ku**
  - description: multilooked I2+Q2 measurement for ku band
- **nb_stack_l1bs_echo_sar_ku**
  - description: number of waveforms summed in stack
- **max_stack_l1bs_echo_sar_ku**
  - description: maximum power of stack
- **max_loc_stack_l1bs_echo_sar_ku**
  - description: Location of the maximum power of stack
- **stdev_stack_l1bs_echo_sar_ku**
  - description: standard deviation of stack
- **skew_stack_l1bs_echo_sar_ku**
  - description: skewness of stack
- **kurt_stack_l1bs_echo_sar_ku**

  - description: kurtosis of stack
- **beam_ang_l1bs_echo_sar_ku**
  - description: look angles in stack
- **beam_form_l1bs_echo_sar_ku**
  - description: flag on beam formation quality in stack
- **burst_start_ind_l1bs_echo_sar_ku**
  - description: Burst start index for stack building
- **burst_stop_ind_l1bs_echo_sar_ku**
  - description: Burst stop index for stack building
- **iq_scale_factor_l1bs_echo_sar_ku**
  - description: Dynamic scale factor for I/Q waveforms i_echoes_ku_l1bs_echo_sar_ku and q_echoes_ku_l1bs_echo_sar_ku
- **i_echoes_ku_l1bs_echo_sar_ku**
  - description: Fully calibrated ku band echoes, i measurements aligned within the stack
- **q_echoes_ku_l1bs_echo_sar_ku**
  - description: fully calibrated ku band echoes, q measurements aligned within the stack
- **start_look_angle_stack_l1bs_echo_sar_ku**
  - description: start look angle in stack
- **stop_look_angle_stack_l1bs_echo_sar_ku**
  - description: stop look angle in stack
- **start_beam_ang_stack_l1bs_echo_sar_ku**
  - description: start doppler beam angle in stack
- **stop_beam_ang_stack_l1bs_echo_sar_ku**
  - description: stop doppler beam angle in stack
- **power_var_stack_l1bs_echo_sar_ku**
  - description: power variations within the stack

CHAPTER 4

Frequently Asked Questions

4.1 Why does the layout of DeDop Studio looks different in my computer?

DeDop Studio was designed at a window size of 1366 x 768, which is at the moment the most common laptop screen resolution. This means that all the component placements were decided based on what looks best at 1366 x 768 resolution. Panel placements may be different in lower resolution, but they should all still be fully functional. Please drop us a message if your experience is greatly affected by this.

CHAPTER 5

API Reference

Warning: This chapter it generated from Python source code and not yet very useful. Also, the API described here is not yet stable.

5.1 Input / Output

5.2 Configuration

Processor configuration management.

class `dedop.conf.`**`CharacterisationFile`**(*cst: dedop.conf.constants.ConstantsFile*, *filename: str = None*, ***kwargs*)

class for loading the Characterisation File

`antenna_angles`
array of antenna angles

`antenna_angles_spacing`
spacing between antenna angles

`antenna_gain_ku`
antenna gain for ku-band

`antenna_weights`
array of antenna weights

`brf_sar`
burst repetition frequency

`bw_ku`
Ku-band bandwidth

`chirp_slope_ku`
the chirp slope for the Ku-band (derived parameter)

freq_ku
emitted frequency in Ku-band

mean_sat_alt
mean altitude of the satellite

n_ku_pulses_burst
number of ku-band pulses per burst

n_samples_sar
number of samples per each SAR pulse

power_tx_ant_ku
antenna SSPA RF peak transmitted power in Ku band

prf_sar
pulse repetition frequency

pulse_length
pulse length

uso_freq_nom
USO frequency nominal value

wv_length_ku
the Ku-band wavelength (derived parameter)

class dedop.conf.**ConstantsFile**(*filename: str = None*, ***kwargs*)
class for loading the Constants file

c
speed of light

earth_radius
radius of the earth

flat_coeff
flattening coefficient of the WGS84 ellipsoid

pi
Pi number

sec_in_day
number of seconds in a day

semi_major_axis
semi-major axis of WGS84 ellipsoid

semi_minor_axis
semi-minor axis of WGS84 ellipsoid

class dedop.conf.**ConfigurationFile**(*filename: str = None*, ***kwargs*)
class for loading the Configuration File

class dedop.conf.**AuxiliaryFileReader**(*filename: str = None*, ***kwargs*)
class for reading auxiliary files

class dedop.conf.**AuxiliaryParameter**(*parameter_name: str*, *doc_string: str = None*, *param_type: type = None*, *cast_type=None*, *optional: bool = False*, *default_value=None*)
this is a variable descriptor class. It exists to link internal auxiliary parameter names with those used in the JSON documents.

This also enables the code in the AuxiliaryFileReader base class to distinguish easily which members of the child classes that define parameters - this allows us to e.g: throw a warning if an input file defines an unexpected parameter

5.3 Model

5.4 Processor

5.5 Web Socket Service

5.6 Utilities

Common utilities used throughout the DeDop project.

class `dedop.util.`**`Parameter`**(*name*, *default_value=None*, *data_type=None*, *description=None*, *value_set=None*, *units=None*, *position=0*)

The Parameter class is used to describe, validate, and convert parameter values.

`data_type`

The parameter's data type. Must be an instance of the type class.

`default_value = None`

The parameter's default value. Default value is None.

`description = None`

The parameter's description. Default value is None.

`static get_parameters`(*clazz*)

Get a dictionary that maps names to Parameter descriptors or nested Parameter dictionaries. The dictionary is collected from the given clazz' hierarchy. Per-class parameter descriptor dictionaries are retrieved by looking up the class attribute 'parameters' which must be a dictionary if it exists. :param clazz: A class :return: A dictionary of possibly nested Parameter instances

`is_bound_to_value_set = None`

True, if the parameter value must be one of the value_set values. Default value is False.

`name`

The parameter's name.

`position = None`

The parameter's position which can be used for sorting if it is a positional value. Default value is 0.

`units = None`

The parameter value's physical units. Default value is None.

`value_set = None`

The parameter's value set. Default value is None.

CHAPTER 6

Indices and Tables

- genindex
- modindex
- search

Python Module Index

Index