ATLAS (**A**u**T**omated **L**idar Quality **A**nalysis **S**oftware) manual

Author: Siomos N.

Version: 1
Created on: 12.03.2025

Table of Contents

1 Installation

1.1 Introduction

The **recommended way to install ATLAS** is by creating a conda environment

=> see section "Installing ATLAS with conda"

Expert python users that don't want to install conda and know what they are doing can alternatively try one of these approaches

=> see next section "Installing ATLAS without conda"

1.2 Installing ATLAS without conda

ATLAS can be installed by using the following command in a **python 3.9** environment:

pip install atlas_actris

Currently using python 3.9 is mandatory because other versions might lead to dependency issues.

Linux users that happen to have python 3.9 installed in their system and have admin rights can install pip on the system and create a virtual environment with *venv*. Please note that *venv* uses the system's python, so this approach will probably fail to work if the system is using something other than python 3.9. In the future ATLAS support will be added for a range of python versions.

Windows users can install python 3.9 e.g. from here:

https://www.python.org/downloads/release/python-390/

and then use *venv* to create a virtual environment. Please make sure that *pip* is installed and available in the environment.

After the environment is installed, activate it and use the following command:

pip install atlas_actris

The ATLAS source code will be installed in a folder named atlas_dev within your environment folder. The quickest and easiest way to find it is to just search for it. However, users are strongly discouraged of making changes directly on the source code installed by pip.

If changes must be made then it is recommended to download and work on the ATLAS source from Github (https://github.com/nikolaos-siomos/ATLAS) within the created environment, so that the dependencies are met. The source code installed by pip should be used **only for executing** the main scripts using configuration and setting ascii files stored out of the python environment.

1.3 Installing ATLAS with conda

The recommended way to install ATLAS is through conda, because it is possible to create environments with a specific python version installed.

Users with any form of **conda already installed** (e.g. through anaconda, miniconda, or miniforge) can skip steps 1 and 2.

In this tutorial we will use miniforge (see: What is the difference between miniconda and miniforge?),

but the process of installing conda through anaconda or miniconda is very similar. Please follow the steps below.

Step 1: Download the miniforge installer

The installer for different operating systems is provided in the following link:

https://conda-forge.org/download/

Step 2: Install miniforge

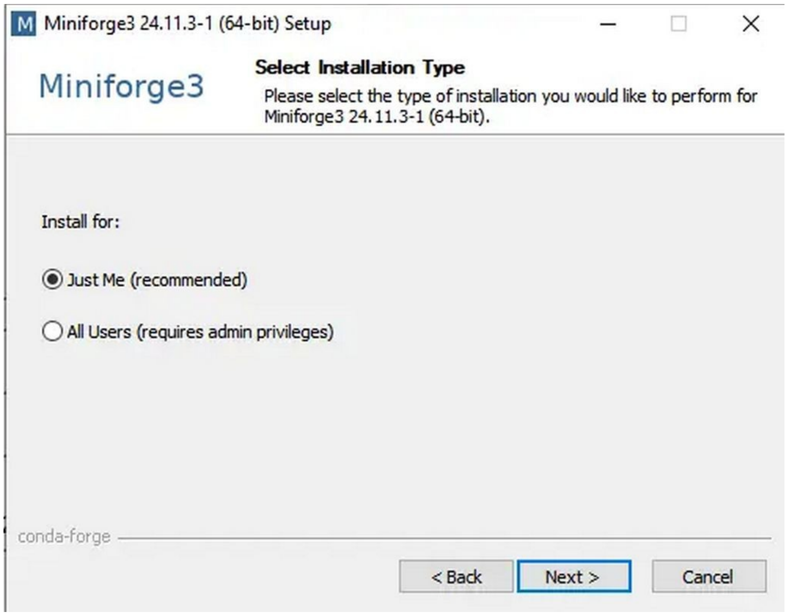
Please follow the instructions on the webpage to execute the installer.

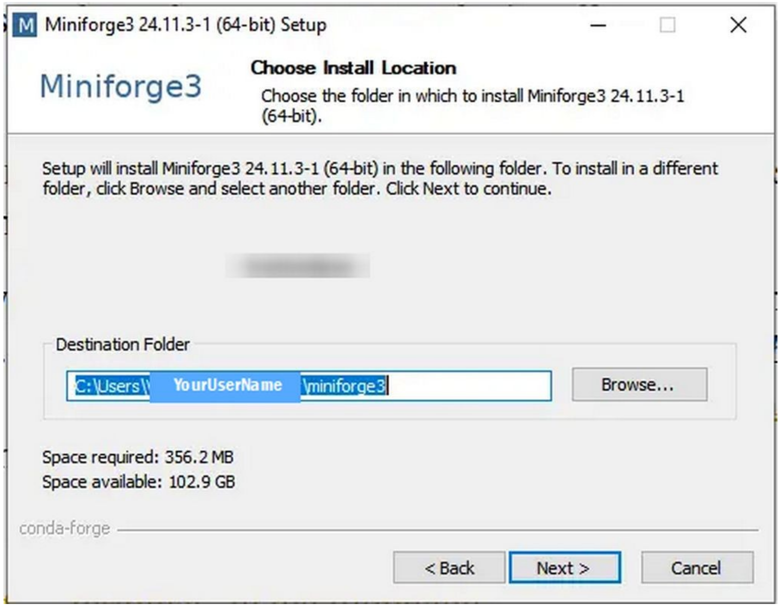
Linux:

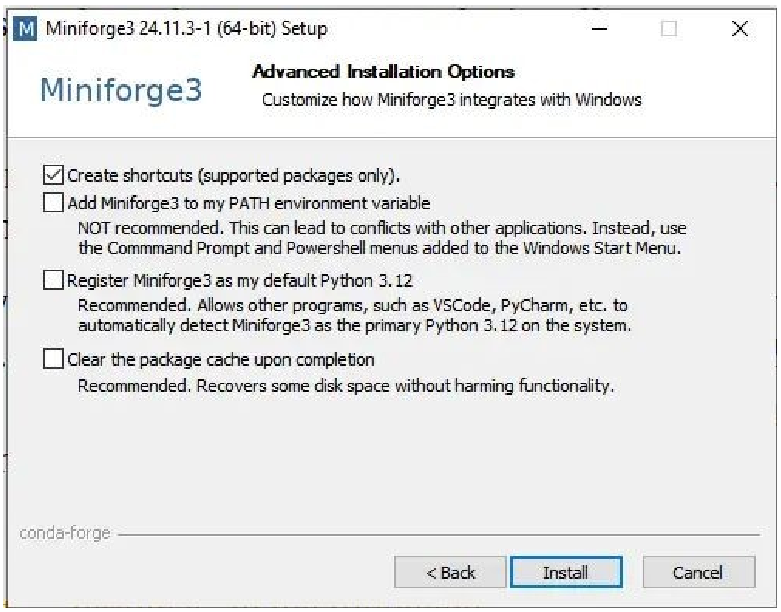
At the last stage of installation, users will be asked if they want to have the *base* environment activated by default on their shell (terminal).

Selecting *yes* in most cases will make life easier. Linux users should keep in mind that if you already have enabled this option (e.g. from a previous anaconda installation) the new *base* environment from miniforge **will supersede** the old *base* environment that was used until now in the terminal by default. If this is not what you want then please select *no* at this stage.

Windows 10:




Miniforge3 24.11.3-1 (64-bit) Setup
Miniforge3
Choose Install Location
Choose the folder in which to install Miniforge3 24.11.3-1 (64-bit).
Setup will install Miniforge3 24.11.3-1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Users\ YourUserName \miniforge3
Browse...
Space required: 356.2 MB
Space available: 102.9 GB
conda-forge
< Back
Next >
Cancel


Miniforge3 24.11.3-1 (64-bit) Setup
Miniforge3
Advanced Installation Options
Customize how Miniforge3 integrates with Windows
Create shortcuts (supported packages only).
Add Miniforge3 to my PATH environment variable
NOT recommended. This can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the Windows Start Menu.
Register Miniforge3 as my default Python 3.12
Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Miniforge3 as the primary Python 3.12 on the system.
Clear the package cache upon completion
Recommended. Recovers some disk space without harming functionality.
conda-forge
< Back
Install
Cancel

<u>**Step 3**: Activate the base environment</u>

Linux:

People that selected *yes* at the end of the previous step should already have an activated base environment next time they open a new terminal.

People that selected *no* at the end of the previous step can activate the base environment in the terminal using the following command

source <path to the activate script inside the miniforge folder>

For example:

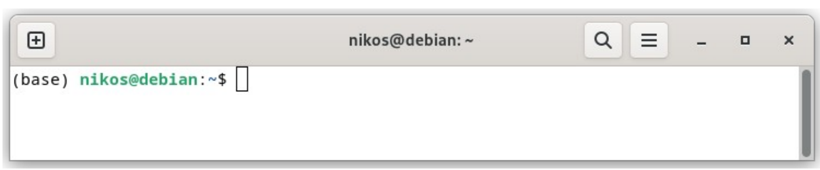
source /home/nikos/miniforge3/bin/activate

Windows 10:

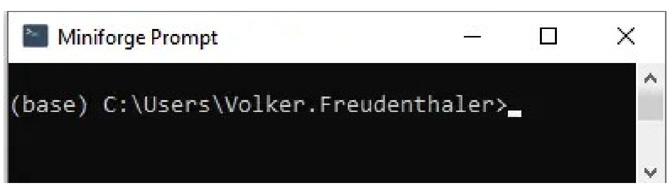
In the Start Menu there is a new item "Miniforge Prompt". If you click on it, you open a Windows command prompt (terminal) with the (*base)* environment already activated.

No matter which approach was selected, the users should see "(base)" written on their terminal like in the example below:

Linux:

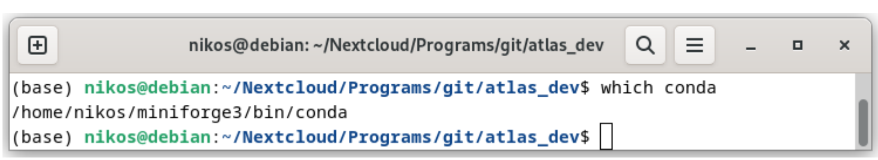


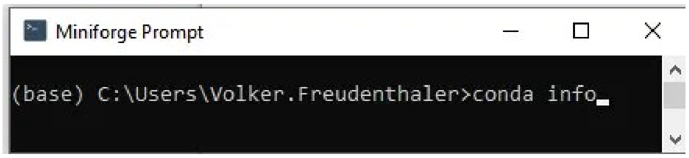
Windows:



To verify that you are using the right *base* environment please type the following command while being inside the base environment:

Linux:



Windows: 

It should return a path within the miniforge folder (or your expected conda installation folder). For example:

Linux: */home/nikos/miniforge3/bin/conda*

Windows: active env location : C:\Users*YourUserName*\miniforge3

Step 4: Create a new environment

In the base environment type:

Linux: *(base) user@linux_distro:>conda create -n atlas_box python=3.9 pip*

Windows: *(base) C:\Users\YourUserName>conda create -n atlas_box python=3.9 pip*

This will create a new environment called “atlas_box” with python 3.9 and *pip* installed. The users can also select a different name for their environment. Please note that if the name “atlas_box” is already assigned to an existing environment, a new name must be selected. This is relevant to users that are using an existing conda installation with an *atlas_box* environment.

It is convenient to install an integrated development environment (IDE) to better edit and view scripts (optional). In the example below the Spyder IDE will also be installed when the atlas-box environment is created:

conda create -n atlas_box python=3.9 pip spyder

If you missed this, you can install spyder later in the atlas_box environment by typing:

conda install -n atlas_box spyder

Note that in both case the spyder from the conda-forge repository channel will be installed if miniforge was installed. In an anaconda or miniconda installation, spyder from the anaconda channel will be installed.

<u>Step 5</u><u>: Activate the new environment</u>

The environment can be activated by typing:

conda activate atlas_box

For example:

<u>Step 6</u><u>: Install or update ATLAS</u>

Install ATLAS from the pip repository (https://pypi.org/project/atlas-actris/) by typing the following command in the activated *atlas_box* environment:

pip install atlas_actris

For example:

To install a specific version of ATLAS (not recommended) you can type instead:

pip install atlas_actris==0.4.9

The ATLAS source code will be installed in a folder named atlas_dev within your environment folder. The quickest and easiest way to find it is to just search for it. However, users are strongly discouraged of making changes directly on the source code installed by pip.

If changes must be made then it is recommended to download and work on the ATLAS source from Github (https://github.com/nikolaos-siomos/ATLAS) within the created environment, so that the dependencies are met. The source code installed by pip should be used **only for executing** the main scripts using configuration and setting ascii files stored out of the python environment.

<u>Step 7 (optional)</u><u>: Update ATLAS</u>

If ATLAS is already installed and a new update came out, it is possible to update through pip by typing:

pip install atlas_actris --upgrade

For example:

Please note that it is not possible to install/update ATLAS using conda. Conda is used only for the creating of the environment and the installation of pip and python3.9.

<u>Important note for ATLAS developers</u>: Upgrading ATLAS through pip will **delete any local changes** in the scripts of the ATLAS source folder, which is installed within the conda environment. Please avoid making any changes to that folder. Users that are interested in developing their own scripts or want to contribute to the development of ATLAS are strongly encouraged to still install ATLAS through pip (so that the dependencies are correctly installed) but download and work on the source code from the ATLAS Github repository: https://github.com/nikolaos-siomos/ATLAS

2 Using ATLAS

Executing ATLAS (__master__.py) is quite straightforward given that the necessary input files (settings file, configuration file) are properly configured and the paths to the raw data files are defined correctly.

ATLAS can be called either directly from the command-line using an argument list or by using a script. Developers interested in creating their own scripts or users that prefer using the command-line can have a look to the ATLAS_developer_manual.pdf file for a complete explanation of the available arguments.

In all other cases, calling ATLAS with the official calling script (__call_atlas_interactive__.py) is recommended. As of ATLAS version 0.5.0, templates calling scripts are provided in the *calling_scripts* folder. In this document we will explain how to use the __call_atlas_user_version__.py script.

As of version 0.5.0 the following official modules are available in the ATLAS source folder:

- __master__.py – the main ATLAS module used for processing and visualizing the QA tests
- __call_atlas_interactive__.py – the calling script used for calling the __master__ module
- __signal_viewer__.py – a module that can be used to inspect the raw and/or range-corrected signals. It can only be used after using ATLAS once because it relies on its output

2.1 Calling ATLAS

In this section we will describe how to use __call_atlas_interactive__.py script.

Step 1: Prepare the input data

The script is located within the ATLAS folder and can be executed as a main program. Before execution it, it is important to order the input lidar data in the following way:

- /<main_data_folder> or alternatively /<main_data_folder>/<scc_station_id>
    - /<scc_lidar_id>_<scc_version_id>_<scc_config_id>_<data_identifier>
        - /nrm/<raw_files>
        - /tlc
            - /north/<raw_files>
            - /east/<raw_files>
            - /south/<raw_files>
            - /west/<raw_files>
        - /tlc_rin
            - /inner/<raw_files>
            - /outer/<raw_files>
        - /pcb
            - /+45/<raw_files>
            - /-45/<raw_files>
        - /drk
    - /configurations
        - configuration_file_<scc_lidar_id>_<scc_version_id>_<scc_config_id>.ini
    - /settings
        - settings_file_<scc_lidar_id>_<scc_version_id>_<scc_config_id>.ini
    - /radiosondes
        - /<scc_lidar_id>

This is the default folder structure that the calling script expects. More specifically:

- **<main_data_folder>**: The main data folder where he parent folders can be placed directly under this folder
- **<main_data_folder>/<scc_station_id>**: For analyzing measurements from multiple stations it might be convenient to place the parent folders not directly under the main_data_folder but within a subfolder named according to the scc_station_id (e.g. aky, brc, mun)
- **<scc_lidar_id>_<scc_version_id>_<scc_config_id>_<data_identifier>**: The parent folder where folders containing the QA tests will be placed. The first three main parts correspond to the SCC lidar ID, the SCC version ID, and the SCC configuration ID.

  The SCC configuration ID provided here will be used to link with the configuration and settings files. The SCC lidar ID will be used to link with the radiosondes folder.

  It is mandatory that all 4 parts of the name, separated by the "_" character, are provided.

  The data identifier part can be used to separate between measurements of the same SCC configuration. It is highly recommended that the date is at least provided with an yyyymmhh format. After the date part custom information can follow separated by "_" (e.g. no_filter, test, air_condition_off etc)
- **nrm**: The folder containing the raw files of the Rayleigh measurement
- **tlc**: The folder containing the 4 sector folders of the quadrant telecover test (north, east, south, west). Each folder includes all raw files of the corresponding type. For example, if 4 iterations are performed, the north folder will include all north files regardless of the temporal gaps
- **tlc_rin**: The folder containing the 2 sector folders of the ring telecover test (inner, outer). Each folder includes all raw files of the corresponding type.
- **pcb**: The folder containing the 2 folders of the polarization calibration measurement (+45, -45). Each folder includes all raw files of the corresponding type. Note that +45 and -45 are general notations. If, for instance, a half-waveplate is used for the polarization calibration measurement the -22.5 files should go to the -45 folder
- **drk**: The folder containing either a common short dark to be used for all other QA tests or an independent long dark measurement. Using a corresponding short dark measurement for each QA test is not currently supported automatically but it can be achieved, if necessary, by splitting the QA tests in multiple parent folders

  Note that sometimes more that one QA measurements of the same might be available under the same parent folder. In such cases, it is still possible to point ATLAS to the desired directory by configuring the initialization file of __call_atlas_interactive__. This option will be discussed below in more detail.
- **configurations**: The folder where all ATLAS configuration files are placed.

- **configuration_file_<scc_lidar_id>_<scc_version_id>_<scc_config_id>.ini**: The configuration file of a specific SCC configuration. The SCC lidar, version, and configuration IDs must be included in the name as displayed. More information about setting up this file can be found in section 3.
- **settings**: The folder where all ATLAS configuration files are placed.
- **settings_file_<scc_lidar_id>_<scc_version_id>_<scc_config_id>.ini**: The settings file of a specific SCC configuration. The SCC lidar, version, and configuration IDs must be included in the name as displayed. More information about setting up this file can be found in section 4.
- **radiosondes**: The folder includes subfolders named according to the SCC lidar ID. The corresponding radiosondes files are placed inside. This is done because certain stations might have lidar systems installed in different locations which would require different radiosondes. More information about the supported radiosonde formats can be found in section 5.

An example with 4 parent folders that correspond to 3 different SCC configurations (100, 101, 102) and 2 lidar systems (140, 150) is displayed below. Multiple parent folders that correspond to the same configuration can be placed under the same main data folder.

- /mydisk
    - /140_9_100_20250319
    - /140_9_100_20250320
    - /140_9_100_20250320_filter_test
    - /140_9_101_20250320
    - /150_10_102_20250319
    - /configurations
        - configuration_file_140_9_100.ini
        - configuration_file_140_9_101.ini
        - configuration_file_150_10_102.ini
    - /settings
        - settings_file_140_9_100.ini
        - settings_file_140_9_101.ini
        - settings_file_150_10_102.ini
    - /radiosondes
        - /140
        - /150

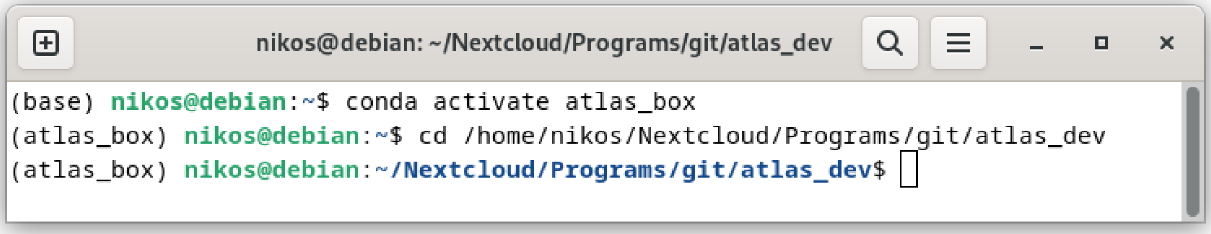
Step 2: Run the calling script

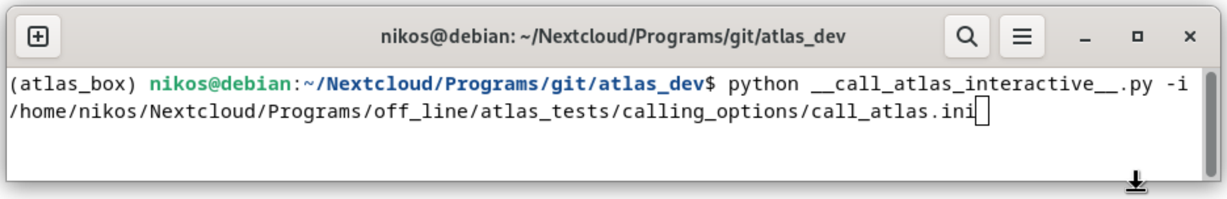
In order to run the script either use the command-line or spyder.

Running in the command-line:

- Activate the environment where ATLAS was installed (check section 1 for more details)
- Within the environment change directory to the ATLAS source folder and type the following command:

  python __call_atlas_interactive__.py -i <path to the initialization file>

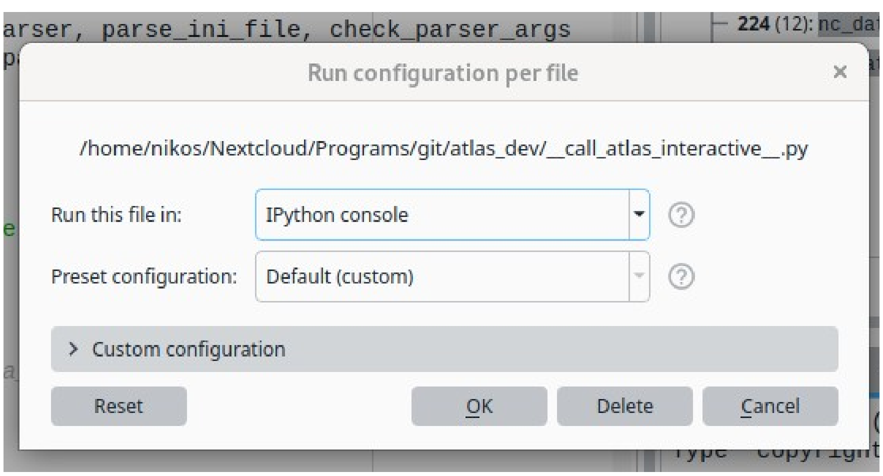




The program requires a single argument which you can provide but typing either -*i*, as in the example, or --*ini_file*

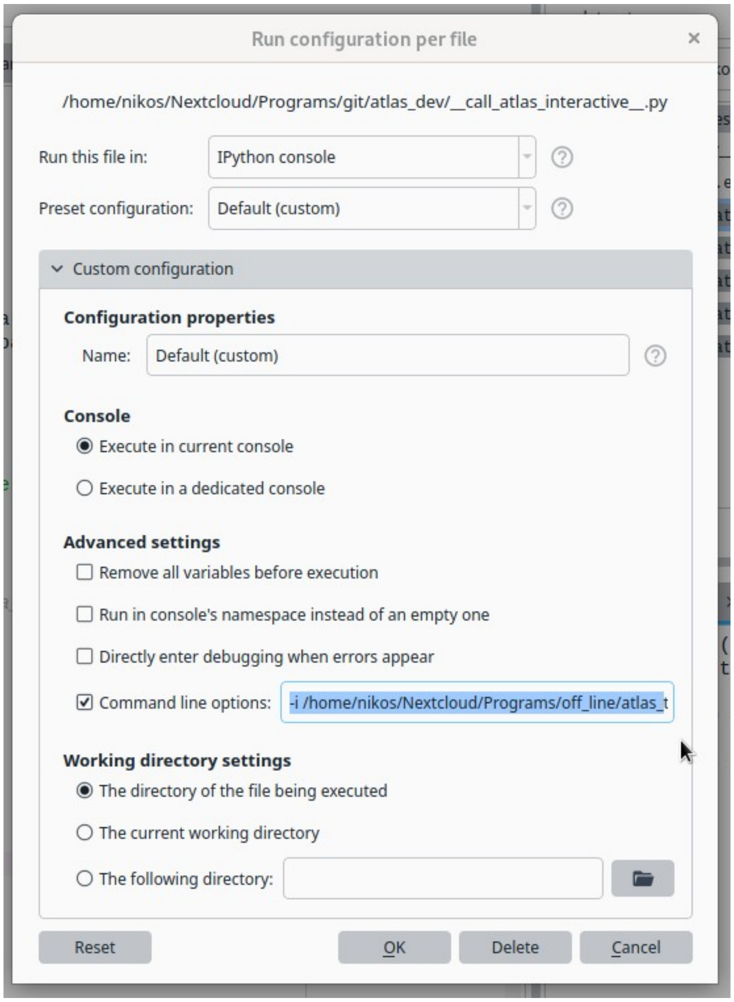
Running through Spyder:

- Activate the environment where ATLAS was installed (check section 1 for more details)
- Load spyder by typing *spyder* in the environment
- Press Ctrl + F6 to load the *Configuration per file* tab or navigate there from the *Run* drop-down menu.



- Expand the Custom configuration part, tick the Command line Options checkbox and fill it in as follows:

  -*i <path to the initialization file>*



- Close the *Configuration per file* tab and press F5 to run the program.

The provided absolute path to the initialization file points to the .ini needed to run the clalling script. The structure of the initialization file will be explained in detail below

2.2 Preparing the initialization file

The initialization script consist of 4 sections:

- autodetect_paths
- options
- explicit_paths (optional)
- explicit_folders (optional)

A template of the initialization file can be found in the templates folder.

[autodetect_paths]

The user should strictly follow the folder structure described in section 2.1 otherwise the paths will not be determined correctly. The parent folders should be named as: <scc_lidar_id>_<scc_version_id>_<scc_config_id>_<date>_<label>, with label being optional. The label can be anything describing the data, it can also be omitted. For example: '438_20250101' or '437_20230320_no_filter'

Mandatory fields

main_data_folder: The folder containing the parent folders and all other necessary information such as the configurations, settings, and radiosondes. These folders can be placed either directly in the main_data_folder or one directory level deeper. This can be achieved by using the **scc_station_id** field (explained below)

Optional fields

scc_station_id: The SCC station ID. Provide it if the parent folders must be placed under the <main_data_folder>/<station_id> directory. Leave empty if the parent folders must be placed placed in the <main_data_folder> directory

scc_configuration_id: The SCC configuration ID. It will be used for identifying the parent folder, the configuration files and the settings files. If not provided, the available configurations will be inferred from the existing parent folders and the user will be prompted to select one after running the calling script

data_identifier: A unique pattern to identify the parent folder among folders that correspond to the same SCC configuration. Recommendation: Provide at least the date in yyyymmhh format. A custom label can be included as follows: <date>_<label> If not provided and there are more than 1 parent folders available with the same SCC configuration ID, the user will be prompted to provide one after running the calling script

[Options section]

Mandatory fields

file_format: Raw lidar file format. Currently only licel and polly_xt are supported. Choose one of: 'licel', 'polly_xt', 'licel_matlab', 'polly_xt_first', 'licel_old2rack'

- scc: Raw SCC netcdf format
- licel: Official Licel binary format
- polly_xt: PollyXT netcdf format, does not work for 1st generation systems
- licel_matlab: use for the 'brc' and 'run' stations
- polly_xt_first: 'evo' station – first generation PollyXT
- licel_old2rack: 'cbw' station

Optional fields

quick_run: Defaults to False. If set to True the converter and the preprocessing modules will not be called if the algorithm detects output files already produced by them for a specific measurement. This mainly saves time during execution.

process: The user can choose specific QA test(s) to process. Use any of:

- ray: Rayleigh Fit
- tlc: Telecover Test
- pcb: Polarization Calibration
- drk: Long dark (used only for the quicklooks)
- off: Nothing will be processed

Defaults to: ray, tlc, pcb, drk

process_qck: Choose which quicklooks to process. It must be a subset of **process**. Note that each visualizer module (ray, tlc, pcb) can create their own quicklooks. Choose among

- ray: Rayleigh Fit
- tlc: Telecover Test
- pcb: Polarization Calibration
- drk: Long dark
- off: Nothing will be processed

Defaults to: ray, tlc, pcb, drk

slice_rayleigh: Provide temporal limits for the processing of the Rayleigh fit test. Use the following format: HHMM for the limits. For example use: --slice_rayleigh 2300, 0130 to slice between 23:00 UTC and 01:30 UTC (the next day). Defaults to: None, None

expert_analyst: Shortcut of the name of the person analyzing the data

export_all: Export all channels in the QA reports (e.g analog channels for Rayleigh fit test and photon channels for the telecover test)

[explicit_paths]

These paths have to be provided if the default folder structure must be overridden. Providing the paths explicitly will deactivate the autodetect_paths process. The paths of these section must be either provided all together or not provided at all. This section is totally optional and can be omitted in the initialization file.

parent_folder: Absolute path to the parent folder

atlas_configuration_file: Absolute path to the ATLAS configuration file

atlas_settings_file: Absolute path to the ATLAS settings file

radiosonde_folder: Absolute path to the folder that includes the radiosonde files.

[explicit_folders]

These folder names have to be provided if the default parent folder structure must be overridden. Explicit names can be provided for any of the QA test folders. If provided, ATLAS will search for those specific folder name per test rather than the default names (e.g. nrm, tlc, tlc_rin, pcb, drk) in the parent folder. Provide only if it is necessary to deviate from the default ATLAS folder structure. This section is totally optional and can be omitted in the initialization file

nrm: Provide the name of the normal folder inside the parent folder (e.g. nrm_02)

pcb: Provide the name of the pol. cal folder inside the parent folder (e.g. pcb_off)

tlc: Provide the name of the quadrant telecover folder inside the parent folder (e.g. tlc_raman)

tlc_rin: Provide the name of the ring telecover folder inside the parent folder (e.g. tlc_rin_01)

drk: Provide the name of the ring telecover folder inside the parent folder (e.g. tlc_rin_01)

3 The configuration file

The *configuration_file.ini* sample file is provided inside the *./templates* folder. The file consists of two main sections, Lidar and Channels. The file is structured according to the following format:

```
[Lidar]

variable_1 = <single_element_1>

variable_2 = <single_element_2>

.....

[Channels]

variable_A = <element_A1>, <element_A2>, ......, <element_An>

variable_B = <element_B1>, <element_B2>, ......, <element_Bn>

......
```

According to the example, the first section consist of variables that take a single value as input while the later consist of variables that take coma separated values as input. For the latter, the number of values per variable is always equal to the number of signal channels that the user wants to process.

Some fields are optional. The raw file parser attempts to fetch all available metadata from the raw files. Default values are provided when the user either leaves them empty or does not include them in the configuration file at all. For example:

```
variable_1 = <single_element_1>

variable_2 =

variable_3 = <single_element_3>
```

Is equivalent to:

```
variable_1 = <single_element_1>

variable_3 = <single_element_3>
```

The variable_2 will not be taken at all into account and default values will be used instead. Please not that some of the fields are mandatory and there are others that are partly mandatory in a scene that using the default values can be quite error prone. More details on the variables themselves are provided the following sections.

A template of the configuration file can be found in the templates folder.

3.1 Section: System

Mandatory Variables

station_id: The 3 letter ID of the station according to the EARLINET DB. This field will be displayed in the filenames of the exported files.

Example: station_id = mun

lidar_name: The full name of the lidar as defined in the SCC. This field will be displayed in the plots.

Example: lidar_name = POLIS

Mandatory Variables when the --operation_mode argument is set to “labeling” (submitting to CARS)

station_name: The full name of the station as defined in the SCC. This field will be displayed in the plots.

Example: station_name = Munich

lidar_id: The ID of the lidar as defined in the SCC (integer). This field will be reported in the plots and the filenames of the exported files.

Example: lidar_id = 83

version_name: The full name of the version that corresponds to the SCC configuration ID to be tested (integer).

Example: version_name = Version 1.0

version_id: The ID of the version that corresponds to the SCC configuration ID to be tested (integer).

Example: version_id = 84

configuration_name: The full name of the SCC configuration to be tested.

Example: configuration_name = 355Depol_Raman

configuration_id: The ID of the SCC configuration to be tested (integer)

Example: configuration_id = 69

<u>Optional Variables (include only to override the raw file metadata)</u>

altitude: The altitude of the station above sea level (in meters).

Example: altitude = 535

latitude: The station latitude (in degrees).

Example: latitude = 48.148

longitude: The station latitude (in degrees).

Example: longitude = 11.573

zenith_angle: Zenith angle of the lidar (in degrees, 0 at zenith, 90 at horizon).

Example: zenith_angle = 41

azimuth_angle: Azimuth angle of the lidar (in degrees, North = 0, E = 90).

Example: azimuth_angle = 0

3.2 Section: Channels

All variables provided for this section should include exactly as many channels as the channel_id variable. If a variable is not relevant for a specific channel (e.g. dead_time for analog channels) fill with “_”. Note that the number of channels provided has to be equal or less than the total number of channel in the raw input files. A subset of the raw channels can be provided in case the user doesn’t want to process all of them. For Licel systems, the Licel channel ID in combination with the laser identifier in the header (see the Licel manual) are used in order to link with the raw binary files. For PollyXT systems there is no specific identifier per channel. The user can use the channel index (first channel --> 1, second channel --> 2, etc) to link with the raw netcdf files. If nothing is provided, all PollyXT channels available in the raw files will be processed. For example, channel_id = 1, 2, 4 means that the 3rd channel ordered as in the raw netcdf file will not be processed. It is therefore important that all input files **have exactly the same channels** when processing a measurement with ATLAS (e.g. same channels for all QA tests). This is also a good practice for Licel systems.

A detailed explanation of all the variables is provided below with examples. For all examples we have assumed the same system of 12 channels (6 physical) including analog and photon detection at 355 parallel and cross, 387 Raman, 532 parallel and cross, and 607 Raman. Please note that all example values presented here correspond to a fictional system and **should do not be blindly taken and applied to an actual system.**

<u>Mandatory Variables</u>

recorder_channel_id: IDs of each channel according to the raw file header.

- For licel systems: provide the licel id per channel that is going to be include. Currently only BT and BC channels are supported.
- For PollyXT systems provide an ascending number per channel starting at 1 and following the order of the channels in the raw PollyXT files.

A subset of the recorder channels can be provided. Only the channels provided here will be processed

Example: recorder_channel_id = BT0,BC0,BT1,BC1,BT2,BC2,BT3, BC3, BT4, BC4, BT5, BC5

scc_channel_id: IDs of each channel according to the SCC configuration. In the future, linking with the HOI will be done through the scc_channel_id (currently optional). Mandatory Variables when the --operation_mode argument is set to “labeling” (submitting to CARS)

Example: scc_id = 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630

laser*:* The ascending laser number according to the licel manual. Use to link with the licel file. The channel_id is not a unique identifier for licel channels. A single channel can synchronize with more than one lasers. Mandatory when the –file_format argument is set to “licel” or “licel_matlab”.

Example: laser = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

telescope_type: The telescope configuration. Choose among:

- n: near range
- f: far range
- x: single telescope

Example: telescope_type = x, x, x, x, x, x, x, x, x, x, x, x

channel_type: The channel type. Choose one among:

- p: co-polar
- c: cross-polar
- t: total (no depolarization)
- v: vibrational Raman
- r: rotational Raman
- a: Cabannes (HSRL)
- f: fluorescence

Example: channel_type = p, p, c, c, v, v, p, p, c, c, v, v

channel_subtype: The channel subtype. Choose among:

- r: Signal Reflected from a PBS
- t: Signal Transmitted through a PBS
- n: N2 Ramal line
- o: O2 Ramal line
- w: H2O Ramal line

- c: CH4 Ramal line
- h: High Rotational Raman
- l: Low Rotational Raman
- a: Mie (aerosol) HSRL signal
- m: Molecular HSRL signal
- b: Broadband Fluorescence
- s: Spectral Fluorescence
- x: No specific subtype

  Example: channel_subtype = r, r, t, t, n, n, t, t, r, r, n, n

Partly Optional Variables

These variables take default values. It is **highly recommended though to fill most of them in** because might not be valid for the system.

dead_time: The dead time of the photon counting channels (in ns). (for analog channels set _). Defaults to 3.7ns for the non-paralyzable case

Example: dead_time = _, 3.571, _, 4.545, _, 3.704, _, 3.167, _, 3.846, _, 3.846

daq_trigger_offset: Provide only if known, otherwise it defaults to 0. The offset of the data acquisition trigger with respect to the Q-switch per channel in bins. Provide negative values if the acquisition starts before the Q-switch (pre-triggering) and positive values if the acquisition starts after the Q-switch.

Example: daq_trigger_offset = -2020, -2011, -2020, -2011, -2020, -2011, -2025, -2014, -2025, -2014, -2025, -2014

background_low_bin: Starting bin of the background correction averaging range. Defaults to:

- The 600th bin before the end of the profile if the pre-trigger region is ≤400 bins
- The 100th bin if the pre-trigger region is >400 bins

Using default values is currently risky. Please make sure that no spikes are appearing in the pretrigger range that would affect the background correction or manually provide this variable.

Example: background_low_bin = 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100

background_high_bin: ending bin of the background correction averaging range. Defaults to:

- The 100th bin before the end of the profile if the pre-trigger region is ≤400 bins
- The 100th bin before the end of the pre-trigger if the pre-trigger region is >400 bins

Using default values is currently risky. Please make sure that no spikes are appearing in the pretrigger range that would affect the background correction or manually provide this variable.

Example: background_high_bin = 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500

Optional Variables (include only to override the raw file metadata)

acquisition_mode: The mode of the recorded signals per channel. Choose among:

- 0: analog
- 1: photon counting

Example: acquisition_mode = 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1

bins: The total bins of the recorded signals per channel.

Example: bins = 8192, 8192, 8192, 8192, 8192, 8192, 4096, 4096, 4096, 4096, 4096, 4096

detected_wavelength: The wavelength of the detected signal according to the applied Interference Filter of each channel (in nm). This variable is provided by default from the raw file metadata but if accuracy is required for molecular calculations the exact central wavelength should be provided.

Example: detected_wavelength = 354.6, 354.6, 354.6, 354.6, 386.5, 386.5, 532.2, 532.2, 532.2, 532.2, 607.5, 607.5

emitted_wavelength: The wavelength of the originally emitted laser beam per channel. If not provided it defaults to the detected wavelength closest to:

- 355 for channels with detected wavelength <520nm
- 532 for channels with detected wavelength ≥520nm and <1000nm
- 1064 for channels with detected wavelength ≥1000nm

This is valid ONLY for Nd:Yag lasers. In addition, if accurate molecular depolarization ratio calculations are required the emitted wavelength should be provided with sub-nanometer accuracy.

> Example: emitted_wavelength = *354.6, 354.6, 354.6, 354.6, 354.6, 354.6, 532.2, 532.2, 532.2, 532.2, 532.2, 532.2*

channel_bandwidth: Interference filter Bandwidth (in nm). Defaults to: 1nm when the file_format argument is set to licel or licel_matlab. The default value is a very rough guess and it can easily lead to high uncertainty in the molecular depolarization value. Exact values provided from the manufacturers should be used instead. For PollyXT system default values are provided based on the raw file metadata.

> Example: channel_bandwidth = 0.5, 0.5, 0.5, 0.5, 1.0, 1.0, 0.5, 0.5, 0.5, 0.5, 1.0, 1.0

data_acquisition_range: The Data Acquisition Range of each analog channel [mV]. Use _ for photon channels

> Example: data_acquisition_range = 100, _, 100, _, 20, _, 100, _, 100, _, 20, _

analog_to_digital_resolution: The analog to digital resolution (in bits) of each analog channel. Use _ for photon channels.

> *Example: analog_to_digital_resolution = 16, _, 16, _, 12, _, 16, _, 16, _, 12, _*

range_resolution: The range resolution of each channel [m]. It will be used to calculate the Sampling Rate.

> *Example: range_resolution = 3.75, 3.75, 3.75, 3.75, 3.75, 3.75, 7.5, 7.5, 7.5, 7.5, 7.5, 7.5*

laser_repetition_rate: The Laser Repetition Rate [Hz]

> *Example: laser_repetition_rate = 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20*

3.3 Channel Nomenclature

Based on the input provided in the configuration file and the header of the raw files an ID is created per channel in ATLAS and accompanies it during processing and exporting. The ID consists of the following 8 characters:

- first 4 characters correspond to the the **detected_wavelength** variable rounded up to the nearest integer and filled with leading zeros wherever applicable (e.g. 0355, 0607, 1064 etc)
- the 5th character corresponds to the **telescope_type** variable (refer to the respective table)
- the 6th character corresponds to the **channel_type** variable (refer to the respective table)
- the 7th character corresponds to the acquisition mode and can be “a” for analog or “p” for photon. The respective information is provided by the **acquisition_mode** variable
- the 8th character corresponds to the **channel_subtype** variable (refer to the respective table)

For example, channel **0355ftpr** would correspond to a far field photon counting co-polar channel at 355nm detected in the reflected path with respect to the PBS

In another example, channel **0387nvan** would correspond to a near field analog vibrational Raman channel at 387nm the corresponds to scattering from N_2 molecules.

Following the 12 channels defined in the examples of section 4.2, we would get the following channels in the ATLAS plots and netcdf files:

0355xpar, 0355xppr, 0355xcat, 0355xcpt, 0387xvan, 0387xvpn, 0532xpat, 0532xppt, 0532xcar, 0532xcpr, 0607xvan, 0607xvpn

4 The settings file

The *settings_file.ini* sample file is provided inside the *./templates* folder. Similarly to the *config_file.ini* it is also structured in sections and shares a similar syntax. The main difference is that the variables of the settings file do not have a specific number of values per section.

The file consists of seven main sections: converter, preprocessor, quicklooks, rayleigh_fit, telecover, and polarization_calibration. As of version 0.2.0 the settings_file is optional and can be skipped entirely. A relevant warning will be printed so that the user knows that default settings where assumed everywhere.

A template of the settings file can be found in the templates folder.

4.1 Section: converter

debug: If set to True, debugging files will be generated in the *<path_to_the_converter_folder>/debug* folder. Note that the path to the converter folder Defaults to: *<parent_folder>/netcdf* but it can be different if manually provided by the user. These included the metadata gathered from the configuration file, the licel header, and the combination of the two. Defaults to: False

Example: debug = True

trim_overflows: This options determines how overflow values in the raw input files will be treated if detected. Choose among:

- 0: the algorithm will stop and provide a diagnostic error (default)
- 1: the files containing at least one overflow value will be screened out
- 2: overflows will be interpolated (use with care and only only a few bins per profile have overflow values)
- 3: overflows will not be excluded, use this only for debugging purposes

Example: trim_overflows = 2

files_per_sector: The number of telecover files per sector (integer). If provided, the telecover files must be placed in a single folder (this should be *<path_to_parent_folder>/tlc* according to the default folder structure). An automated assignment of the telecover files in different sectors will be attempted serially assuming the following temporal sequence of sectors: north – east – south – west. Note that the telecover test can have more than 1 rounds.

Example: files_per_sector = 5

files_per_ring: The number of telecover files per ring (integer). If provided, the telecover files must be placed in a single folder (this should be *<path_to_parent_folder>/tlc_rin* according to the default folder structure). An automated assignment of the telecover files in different sectors will be attempted serially assuming the following temporal sequence of sectors: inner - outer. Note that the telecover test can have more than 1 rounds.

Example: files_per_ring = 5

rsonde_skip_header: Radiosonde parser option. Number of lines to skip at the beginning of the radiosonde ascii file. Defaults to: 1 (1 line reserved for header info)

Example: rsonde_skip_header = 6 (skip the first 6 lines)

rsonde_skip_footer: Radiosonde parser option. Number of lines to skip at the end of the radiosonde file. Defaults to: 0 (no footer assumed)

Example: rsonde_skip_footer = 10 (skip the last 10 lines)

rsonde_delimiter: Radiosonde parser option. The delimiter that separates columns in the radiosonde file choose one of:

- S: space
- C: comma

Defaults to: S.

Example: rsonde_delimiter = C

rsonde_column_index: Radiosonde parser option. The index (integer) of the column that contains the Height, Pressure, Temperature, and Relative Humidity (optional) information. For example, rsonde_columns = 1, 3, 2, 6 means:

- Height: 1st column in the radiosonde file
- Pressure: 3rd column
- Temperature: 2nd column
- Relative Humidity: 6th column

The relative humidity column is OPTIONAL and can be omitted. Defaults to: 2 1 3

Example: rsonde_column_index = 2, 1, 3, 5

rsonde_column_units: Radiosonde parser option. The units of Height, Pressure, Temperature, and Relative Humidity (optional) columns in the radiosonde file. The number of values must be the same as in **rsonde_column_index**. Supported units for:

- height: m_asl (default), km_asl, m_agl, km_agl
- pressure: hPa (default), Pa, atm
- temperature: K (default), C, Cx10
- relative humidity: percent (default), fraction

If the height units are in agl (above ground level) then the station altitude must be provided in the **rsonde_altitude**

Example: rsonde_column_units = m_asl, hPa, C, percent

rsonde_latitude: The radiosonde station latitude in degrees.

Defaults to None.

Example: rsonde_latitude = 48.25

rsonde_longitude: The radiosonde station longitude in degrees.

Defaults to None.

Example: rsonde_longitude = 11.55

rsonde_altitude: The radiosonde station altitude in meters.

Mandatory if the **radiosonde_column_units** of the altitude is either m_agl or km_agl. Defaults to None. In that case the lowermost altitude reported in the radiosonde file is used instead.

Example: rsonde_longitude = 492.0

rsonde_station_name: The name of the station where the radiosonde measurement was performed. Defaults to: None.

Example: rsonde_station_name = Muenchen-Oberschleissheim

rsonde_wmo_number: The WMO number of the radiosonde station. Defaults to: None

Example: rsonde_wmo_number = 10868

rsonde_wban_number: The WBAN number of the radiosonde station. Defaults to: None

Example: rsonde_wban_number =

4.2 Section: preprocessor

vertical_trimming: If set to True then bins above a certain height above the lidar (20km by default) will be removed. Can speed up computations and save RAM. Defaults to: True

Example: vertical_trimming = False

vertical_limit: The maximum height above the lidar in km above which no calculations will be performed. Solar background calculations are performed prior to vertical signal trimming to enable background calculations up to the maximum signal range. Defaults to: 20km

Example: vertical_limit = 25.0

channels: Provided it to process only selected channels. By default, no channel is excluded if this variable is not provided. The channel name must follow the nomenclature of section 4.3.

Example: channels = 0355xcar, 0355xcpr

In this case only the these 2 channels will be processed

exclude_telescope_type: Provide it to entirely exclude all channels of a specific **telescope_type**. By default, no channel is excluded if this variable is not provided.

Example: exclude_telescope_type = x, n

This will exclude all channels with x and n values provided in their **telescope_type**, that is all near field and unspecified channels

exclude_channel_type: Provide it to entirely exclude all channels of a specific **channel_type**. By default:

- if isday = False --> no channel is excluded if this variable is not provided
- if isday = True --> exclude_channel_type = v, f (vibrational Raman and fluorescence channels excluded)

Example: exclude_channel_type = v, r, o, x

This will exclude all channels with v, r, o, and x values provided in their **channel_type**, that is all vibrational and rotational Raman, co- and cross- circular polarized channels

exclude_acquisition_mode: Provide it to entirely exclude all channels of a specific **acquisition_mode**. By default, no channel is excluded if this variable is not provided.

Example: exclude_acquisition_mode = a

This will exclude all channels with 0 values provided in their **acquisition_mode**, that is all analog channels.

exclude_channel_subtype: Provide it to entirely exclude all channels of a specific **channel_subtype**. By default, no channel is excluded if this variable is not provided.

Example: exclude_channel_subtype = w

This will exclude all channels with w values provided in their **channel_subtype**, that is all water vapor channels.

4.3 Section: quicklooks

The following variables:

- **channels**
- **exclude_telescope_type**
- **exclude_channel_type**
- **exclude_acquisition_mode**
- **exclude_channel_subtype**

where defined in section 5.2 and are also applicable here. The usage is the same with the only exception that exclusions take place in the visualization phase and only for the quicklooks

use_log_scale: If set to True, a logarithmic scale will be used for the visualization of the z_axis (signal levels). It affects the selection of the default z_lims. Defaults to False.

Example: use_log_scale = True

use_range: If set to False, the y axis units of the quicklook will correspond to the height above the lidar in meters above sea level. If set to True, the y axis units corresponds to the range above the lidar. This variable determines the units of the following variables:

- **y_lims**
- **z_max_zone**
- **smoothing_range**

Defaults to: True

Example: use_range = False

x_lims: The x axis limits (lower and upper). Use two integers corresponding to the first and last temporal samples (not date!) that will be plotted in the quicklooks. Use 1 to start from the first sample. If values below 1 or above the total number of samples are used, they will be ignored

Example: x_lims = 100, 300

Assuming a Rayleigh Fit measurement with 1000 samples, the quicklook with start cover the zone between the 100th and the 300th.

x_tick: The x axis finest major tick in number of samples. Defaults to: automatic selection.

Example: x_tick = 2

t_lims: The x axis limits in time units. Use the following format hh:mm for both limits (not meant to be used for day-long quicklooks). Defaults to: automatic selection.

Example: *t_lims = 17:38, 21:56*

t_tick: The major tick for the time axis (same as x axis but with different units). Defaults to: automatic selection

Example: t_tick = 5

y_lims: The y axis limits (height/range) in km (lower and upper). Defaults to: 0., 14.

Example: y_lims = 0., 10.

y_tick: The y axis finest major tick in km. Defaults to: 1km

Example: y_tick = 0.5

z_lims: The colorscale limits of the normalized range-corrected signal normalized with the mean value in the **z_max_zone** region. This setting is useful if for example the normalization takes place in a very strong aerosol layer making all other layers difficult to discern. Defaults to:

- z_lims = 0., 1. when **use_log_scale** = False and
- z_lims = 1E-5, 1. when **use_log_scale** = True

*Example: z_lims = 0., 0.9 (assuming **use_log_scale** = False)*

z_max_zone: Provide a zone (min and max height/range) in km. The maximum range-corrected signal value encountered in the zone will be used for the normalization of the range-corrected signal for the quicklook. Particularly useful in order to avoid scaling the colors with a cloud. Defaults to: 0.1, 2.

Example: z_min_zone = 0.5, 1.5

z_min_zone: Provide the zone (min and max height/distance) in km. The minimum range-corrected signal value encountered in the zone will be used as lower z_axis limit in case the **z_lims** are not explicitly provided and **use_log_scale** is set to True. Defaults to 2., 10.

Example: z_min_zone = 5., 8.

smooth: If set to True, a sliding average smoothing filter will be applied on the signals across y axis for better visualization. The **smoothing_exponential**, **smoothing_exponential**, and **smoothing_window** will be ignored if **smooth** is set to False. Defaults to: False

Example: smooth = True

smoothing_range: Set the first and last height/range boundaries in km where smoothing should be applied. If they exceed the actual signal boundaries the actual boundaries will be used instead. Defaults to: 0.05, 14.

Example: smoothing_range = 0.02, 10.

smoothing_window: The smoothing window in the first and last bin of the smoothing region, in m. The widow progressively changes between from the first to the last value. Use a single value to apply a constant window. Defaults to: smoothing_window = 50., 500.

Example: smoothing_window = 100

smoothing_exponential: This variable is ignored if the upper and lower values of **smoothing_window** are the same. Choose one of:

- True: a smoothing window that exponentially increases with height/range will be applied.
- False: a smoothing window that exponentially increases with height/range will be applied

Defaults to: True.

Example: smoothing_exponential = False

dpi: The dots per inch (dpi) resolution of the exported figures. Defaults to: 300 dpi

Example: dpi = 100

color_reduction: If set to True, and provided that Imagemagick is installed, an adaptive color reduction will be applied to save space when exporting the figures. Defaults to True. A warning will be raised if Imagemagick is not installed.

Example: color_reduction = False

4.4 Section: rayleigh_fit

The following variables:

- **channels**
- **exclude_telescope_type**
- **exclude_channel_type**
- **exclude_acquisition_mode**
- **exclude_channel_subtype** (b, s, a, w, c)

where defined in section 5.2 and are also applicable here. The usage is the same with the only exception that exclusions take place in the visualization phase and only for the Rayleigh fit test. Respective default values are provided in parenthesis.

use_range: If set to False, the x axis units of the Rayleigh fit will correspond to the height in meters above the lidar. If set to True, the x axis units corresponds to the range above the lidar. This variable determines the units of the following variables:

- **normalization_region**
- **x_lims**
- **smoothing_range**

Defaults to: True

Example: use_range = False

The following variables:

- **smooth** (True)
- **smoothing_range** (0.250, 20.)
- **smoothing_window** (500.)
- **smoothing_exponential** (False)
- **dpi** (300)
- **color_reduction** (True)

where defined in section 5.3 and are also directly applicable here. Respective default values are provided in parenthesis.

use_lin_scale: If set to True, a linear scale will be used for the y axis (signal axis). If set to False a logarithmic scale will be applied instead. Defaults to False:

Example: use_lin_scale = True

auto_fit: If set to True an automatic identification of the molecular regions will be attempted. If the automatic procedure is successful, the **normalization_region** variable will be ignored. If the procedure is not successful or auto_ray is set to False, the manually-provided/default normalization will be used. Defaults to True

Example: auto_fit = False

normalization_region: The lower and upper limits of the region used for normalizing the signal in the Rayleigh fit. If use_range is called, the limits correspond to distance. If auto_ray is set to True and the automatic identification is successful for a specific channel, the normalization_region variable will be ignored even if provided. Defaults to: 8.5, 9.5

Example: normalization_region = 4., 6.

x_lims: Set the x axis (height/range) limits in km (lower and upper). Defaults to: 0., 20.

Example: x_lims = 0., 24.

x_tick: The x axis finest major tick in km. Defaults to: 2km

Example: x_tick = 1.

y_lims: The y axis (signal) limits (lower and upper) of the normalized rangecorrected signal in m^{-1} sr^{-1} (pseudo-units). It is recommended to skip this variable and use the automatic selection because channels in different wavelengths have different optimal limits. Defaults to: automatic selection

Example: y_lims = 5E-8, 5E-5

4.5 Section: telecover

The following variables:

- **channels**
- **exclude_telescope_type**
- **exclude_channel_type**
- **exclude_acquisition_mode**
- **exclude_channel_subtype**

where defined in section 5.2 and are also applicable here. The usage is the same with the only exception that exclusions take place in the visualization phase and only for the Rayleigh fit test

The following variables:

- **use_range** (True)
- **x_lims** (0., 2.4)
- **x_tick** (0.2)
- **y_lims** (automatic selection)
- **normalization region** (1.8, 2.2)
- **smooth** (True)
- **smoothing_exponential** (False)
- **smoothing_range** (0.50, 2.5)
- **smoothing_window** (100.)
- **dpi** (300)
- **color_reduction** (True)

where defined in section 5.4 and are also applicable here. The default values are not always the same and are provided in parenthesis. Note that:

- The **use_range**, **use_lin_scale**, **x_lims** and **x_tick** variables correspond to the x axis that is the same for all the 3 subplots of the telecover test.
- The smoothing options affect only the 2^{nd} and 4^{th} subplot of the telecover test (normalized signals in the near range and their respective deviations from the mean)

- The **y_lims** variable units should be the same with the units of the 2nd telecover subplot (normalized signals in the near range).

use_non_rangecor: If set to True, the non range corrected signals will be used for the left subplot of the telecover test. If set to False the range corrected signals will be used instead. Defaults to: False

Example: use_non_rangecor = True

use_last: While set to True an additional purple line will be added in all subplots of the telecover test. In the first 3 subplots the lines corresponds to the last sector (e.g. N2), while in the 4th subplot (deviations) it is the difference between the normalized signal of the last sector and the normalized mean signal of the correspondin sector e.g. N2 – N. Set to False to not visualize this line. Defaults to True.

Example: use_last = False

4.6 Section: polarization_calibration

The following variables:

- **use_range** (True)
- **dpi** (300)
- **color_reduction** (True)

where defined in section 5.4 and are also applicable here. The default values are not always the same and are provided in parenthesis.

The following variables:

- **smooth** (True)
- **smoothing_exponential** (False)
- **smoothing_range** (0.25, 15.)
- **smoothing_window** (500.)

operate in exactly the same way with the smoothing options defined in section 5.3. Smoothing is applied on the signals used for the ratios in the calibration subplot (left) of the polarization calibration figures and also on the ratios in the Rayleigh subplot (right) of the polarization calibration figures.

ch_r: Provide here channel names from the available ones with the reflected **channel_subtype** (e.g. 0355xpar). The number of reflected channels must be the same as the number of the respective transmitted channels provided by **ch_t** and by the GHK parameters. By default, all available reflected channels will be matched to all available transmitted channels that share the same telescope type, detection mode, and wavelength. WARNING! The field is mandatory if non-default GHK values are applied.

Example: ch_r = 0355xppr, 0532xcpr

ch_t: Provide here channel names from the available ones with the transmitted **channel_subtype** (e.g. 0355xcat). The number of reflected channels must be the same as the number of the respective transmitted channels provided by **ch_r** and by the GHK parameters. By default, all available reflected channels will be matched to all available transmitted channels that share the same telescope type, detection mode, and wavelength. WARNING! The field is mandatory if non-default GHK values are applied.

Example: ch_t = 0355xcpt, 0532xppt,

calibration_region: The lower and upper limits of the region used for Δ90 calibration. If use_range is called, the limits correspond to distance. Defaults to: 2., 4. km

Example: calibration_region = 3., 4.

rayleigh_region: The lower and upper limits of the region used for the comparison with the Rayleigh atmosphere. Defaults to: 8.5, 9.5

Example: rayleigh_region = 8.5, 9.5

x_lims_calibration: The x axis (height/range) limits in km (lower and upper) for the pol. calibration subplot. Defaults to: 0, 15

Example: x_lims_calibration = 0.1, 6.

x_lims_rayleigh: The x axis (height/range) limits in km (lower and upper) for the Rayleigh comparison subplot. Defaults to: 0, 15

Example: x_lims_rayleigh = 0.1, 8.

x_tick_calibration: The x axis finest major tick in km for the pol. calibration sunplot. Defaults to: 2.

Example: x_tick_calibration = 1.

x_tick_rayleigh: The x axis finest major tick in Km for the Rayleigh plot. Defaults to: 2.

Example: x_tick_rayleigh = 1.

y_lims_calibration: The y axis limits (lower and upper) of the gain ratios at +-45. Used for the pol. calibration subplot. Defaults to: automatic selection

Example: y_lims_calibration = 0, 0.1

y_lims_rayleigh: The y axis limits (lower and upper) of the volume depolarization ratio. Used for the Rayleigh subplot. Defaults to: automatic selection

y_lims_rayleigh = 0, 0.1

K: The K value for each channel pair. Defaults to: 1 for all channels

Example: K = 1.05, 1.

G_R: The G value for the reflected channel of the pair. Defaults to: 1 for all channels (assuming no correction for the emission and the receiver)

Example: G_R = 1., 1.

G_T: The G value for the transmitted channel of the pair. Defaults to: 1 for all channels (assuming no correction for the emission and the receiver)

Example: G_T = 1., 1.

H_R: The H value for the reflected channel of the pair. Defaults to:

- 1 for all co-polar (p) reflected channels
- -1 for all cross-polar (c) reflected channels

(assuming no correction for the emission and the receiver)

Example: H_R = 0.94, -0.98

H_T: The H value for the transmitted channel of the pair. Defaults to: 1 or -1 for all co-polar (p) and cross-polar (c) transmitted channels, respectively (no receiver optics + emitted pcb. state correction)

- 1 for all co-polar (p) transmitted channels
- -1 for all cross-polar (c) transmitted channels

(assuming no correction for the emission and the receiver)

Example: H_T = -0.94, 0.98

R_to_T_transmission_ratio: The transmission ratio of the R to T channels per pair (T_R/T_T). $T_R = 1$ and/or $T_T = 1$ if no filter was applied during calibration. Defaults to: 1 for all pairs

Example: R_to_T_transmission_ratio = 8., 0.15

5 The radiosonde files

This section includes some more details about the radiosonde files and how ATLAS is handling them. Please refer also to section 5.2 and specifically to the following variables:

- **rsonde_skip_header**
- **rsonde_skip_footer**
- **rsonde_delimiter**
- **rsonde_column_index**
- **rsonde_column_units**
- **rsonde_geodata**
- **rsonde_station_name**
- **rsonde_wmo_number**
- **rsonde_wban_number**

The file format is general ascii. The parser is flexible and can handle headers and footers, two different delimeters (comma and space), custom column order, and custom units.

<u>The Radiosonde folder</u>

This folder should contain the radiosonde files. At this stage only ascii files can be parsed. Note that more than one files can be provided here. ATLAS scans all files and selects the one that it is closest in time with the Rayleigh measurement. Depending on the application, it can be more convenient to use a specific radiosonde folder where all radiosonde data is included and provided it as a command line argument when running ATLAS. Note that the time difference between the Radiosonde and the lidar measurement **cannot be larger than 18 hours**.

<u>Filename Format</u>

The name of the radiosonde file must follow a specific format so that it is recognized by ATLAS. It must start with the date and end with .txt Whatever comes after the date and before the extension is not important for parsing but can be helpful for sorting purposes for the user:

yyyymmdd_hhmm<add_whatever_you_want_here>**.txt**

Example: 20230219_1200_Munich.txt

This radiosonde file corresponds to 19.02.2023 at 12:00 UTC.