# Use of generate\_days\_netcdf\_lidar.py

This code generates netCDF files compatible with CF1.6-AMF1.0 from the Chilbolton Atmospheric Observatory Halo Doppler Streamline Pro lidar and the Vaisala CL51 ceilometer. Data from both instruments are recorded using a Windows DAQ computer in a text format unique to that instrument. From each DAQ computer the data are mirrored to the central /data server at Chilbolton.

There are 4 modules to the code:

generate\_days\_netcdf\_lidar.py which in turn calls:

halo\_python.py

module\_data\_object\_python3.py

lidar\_plot.py

There are other functions which are included within the halo\_python and lidar\_plot modules, which are described at the end of this document. Module\_data\_object\_python3.py, written by David Hooper, is described in a separate document.

The codes run under Python version 2 or 3, but depend on the import of some external libraries such as PyArt which are available at Chilbolton as part of the conda chil\_3\_8 installation. The “#!” first line of the code looks for that installation so the code has been developed using version 3.8.1.

These Python scripts can all be installed in 1 directory, or different directories. If they are in different directories, the command

export PYTHONPATH=/*directory\_where\_code\_stored*/

must be entered at the command line for each directory used.

The user can remove the need to enter this command before running the code by including it in their .bash\_profile file.

## Quick start guide to producing netCDF files from met. data

1. Open a Unix or Linux command window (a run-time environment such as Cygwin can be used if preferred, or a terminal window such as PuTTY).
2. Enter the command **conda activate chil\_3\_8**
3. Connect to the directory where the generate\_days\_netcdf\_lidar.py code is located (unless it is in your $PATH command, it which can you can run it from any directory).
4. Enter **./generate\_days\_netcdf\_lidar.py -s *yyyymmdd\_start* -e *yyyymmdd\_end* -c *calibration factor* –l *lidar* *option* -r *as-recorded or toolbox-cleaned option (Halo only)*.**

*yyyymmdd\_start* = first date for which you want to produce files

*yyyymmdd\_end* = last date for which you want to produce files

*lidar option* = which instrument to process (1 for CL51 ceilometer, 2 for Halo)

*calibration factor* =factor to multiply as-recorded backscattering data to apply calibration, normally determined from strato-cumulus calibration

*as-recorded or toolbox-cleaned option* = determines whether a data file is written using just the as-recorded .hpl files from the Halo lidar (option 1) or also using the cleaned .nc files produced using the FMI Matlab toolbox for removing noise from the raw data (option 2). These data are archived in directories under /data/lidar-doppler-halo/netcdf/calibrated/ and the process used to produce them is described in a separate document… This term is not needed for the CL51 ceilometer.

Default values

## Raw data locations

|  |  |
| --- | --- |
| **Instrument, data type** | **Location** |
| CL51 ceilometer, all data \*.DAT file | /data/vaisala\_ceilometer/mirror\_cl51sky\_local\_ctview |
| Halo lidar, as-recorded vertically-pointing co-polar and VAD \*.hpl files | /data/lidar-doppler-halo/mirror\_lidar-doppler-halo2\_data/Proc/yyyy/yyyymm/yyyymmdd |
| Halo lidar, as-recorded vertically-pointing cross-polar \*.hpl files | /data/lidar-doppler-halo/mirror\_lidar-doppler-halo2\_data/Proc/cross/yyyy/yyyymm/yyyymmdd |
| Halo lidar co-polar \*.nc files after FMI toolbox cleaning of data (1 file per day) | /data/lidar-doppler-halo/netcdf/calibrated/halo-doppler-lidar-118-co/yyyy |
| Halo lidar VAD \*.nc files after FMI toolbox cleaning of data (1 file for each VAD scan during day) | /data/lidar-doppler-halo/netcdf/calibrated/halo-doppler-lidar-118-vad75/yyyy/yyyymmdd |

## How the netCDF production codes work

Diagram?

1. The user input of start date, end date and options to generate\_days\_netcdf\_lidar.py determines which part of the code to call and in turn which raw file is read, as shown in the quick start guide above. This determines the function called by the code.
2. The relevant daily file(s) are opened and read. The formats are described in the relevant instrument manuals, except for the Halo cleaned .nc files produced by the FMI Matlab toolbox. These are described by the documentation for the toolbox and in any case are self-describing.
3. For both instruments, the files include some diagnostics and housekeeping data, more thoroughly for the CL51. These are read and written to .nc files, as well as the profile data. The CL51 files include attenuated aerosol backscattering coefficient, cloud base height from the basic instrument algorithm and cloud coverage and a second measure of cloud base height from the optional sky condition algorithm. The Halo lidar files include attenuated aerosol backscattering coefficient, radial velocity and (signal:noise ratio + 1).
4. The date is passed from generate\_days\_netcdf\_lidar.py to halo\_python.py as a numeric date. If you are using Python 2.x this date is referenced to 01 January 0000. If you are using Python 3.x it’s referred to 01 January 1970. A check is made of which is used based on the magnitude of the numeric date (referred to as nday in the code). A short function, generate\_netcdf\_common(nday), returns the date in various formats for use elsewhere in the code or output .nc file.
5. The directory or directories where raw files are located are determined from the date and a list of files to read is collected. For the Halo lidar one directory contains co-polar fixed vertical and VAD measurements and another contains cross-polar measurements (these are all fixed vertical measurements). The co-polar fixed, cross-polar fixed and co-polar VAD files are read to separate arrays. They each have separate timestamps as the measurements are made sequentially. For the ceilometer there is normally 1 file per day (unless data acquisition was stopped and restarted) and they are not separated into different directories, for example by year.
6. For the Halo lidar there is a process in place to check whether any points for that day were included in the data files for the previous day. This can happen occasionally. When it does, the code writes an extra text data file in /data/amof-netCDF/ncas-lidar-dop-3 when that previous day was processed, highlighting that this has happened and containing only the data from that single timeslot. This effect appears extremely rare for the CL51 ceilometer, so if a profile after midnight is included in a file, it is discarded.
7. For the Halo lidar there is another check, performed by the function halo\_check\_monotonic, which checks whether there is a time in the series of times read from the raw data files where the time decreases. For fixed scans (co- or cross-polar) this would normally be at the end of the day and only affect the last data point. This time and data are then saved for reading when the next day is processed, using the process described in step 6. For VAD scans however, if a scan starts very soon before midnight, several points may be timestamped with a time from the next day. These will be included in the VAD .nc file for the current day with a time of the recorded time + 86400 seconds (i.e. 1 day) in order to keep a complete VAD scan together.
8. For both instruments the length of a range gate (10 m for the CL51, 30 m for the Halo lidar) is hardwired into the code. Along with the number of range gates, which is read from the raw file, these are used to define the altitude variable for the CL51 and the range variable for the Halo lidar. The range/altitude are named and treated differently for the two instruments because the CL51 measures in a fixed direction whereas the Halo lidar can scan. The latter therefore requires more versatility in how the range/altitude is defined. For the CL51, the altitude is defined as the height of the instrument with respect to the WGS84 geoid, rather than using sea level or the instrument location as the reference. The height of the instrument with respect to the geoid is 131 m, so this height is added to the altitude with respect to the location. For the Halo lidar, the range is defined with respect to the instrument location.
9. The quality of each data point is defined, using a series of criteria. This is described in the “Assignment of quality flags” section.
10. For the Halo lidar, if the option “-r 2” is selected, a cleaned .nc file is opened for the co-polar and VAD measurements. No cleaned file is produced by the FMI toolbox for cross-polar files. Checks are performed on the data from these files, particularly to check that the times and number of profiles match and quality flags are assigned in a similar way to those in step 9, although there is an extra flag compared to the as-recorded data (see “Halo Doppler lidar attenuated aerosol backscattering coefficient and radial velocity quality flags” section). Checks on the timestamps between the as-recorded and cleaned files can sometimes reveal a discrepancy, as the criteria for which timestamps are included in the files are sometimes slightly different between the NCAS standard and the FMI toolbox files. This is often just the difference between 1 extra profile in either file and the profiles can still be matched. If the timing is more ambiguous, the entire cleaned file in NCAS format is marked with a Fillvalue. A log is kept in a text file in /data/amof-netCDF/ncas-lidar-dop-3 of cases where there is a discrepancy in the number of timestamps.
11. For the Halo lidar the profiles are expanded from 2 dimensions (time and height) to 3 (time, height, number of angles) using the function my\_broadcast. This meets the requirements of the NCAS standard, which allows for versatility in the range of elevation and azimuth angles included in a scan. The number of angles is only 1 for fixed profiles but for the VAD scans the default is 24 azimuths and 1 elevation. The number of angles is therefore 24. As the elevation remains fixed at 15° for all elements of the VAD in our application, the effect is merely to duplicate the profiles 24 times along that axis. This makes the VAD .nc files rather large but meets the requirements of the data standard.
12. Data files are written using the module\_data\_object\_python3.py module and YAML files also contained in the /home/chilbolton\_software/python/ncas\_python/halo directory. The term “standard” or “advanced” is included to indicate the level of processing in the output .nc files. Standard indicates that the data have not been modified since reading from the raw files. Advanced indicates that a calibration factor has been applied and, for the Halo lidar, the cleaned FMI-toolbox data are included in the file rather than the as-recorded data. The YAML files are as follows:

|  |  |  |
| --- | --- | --- |
| **Data type** | **YAML file** | **netCDF filename format** |
|  |  | ncas-ceilometer-5\_cao\_*yyyymmdd*\_... |
| CL51 ceilometer standard | ncas-ceilometer-5-template\_rawbs.yaml | aerosol-backscatter\_standard\_v1.0.nc |
| CL51 ceilometer advanced | ncas-ceilometer-5-template\_calbs.yaml | aerosol-backscatter\_advanced\_v1.0.nc |
| CL51 ceilometer cloud base height | ncas-ceilometer-5-template\_cbh.yaml | cloud-base\_v1.0.nc |
| CL51 ceilometer cloud coverage and cloud base height | ncas-ceilometer-5-template\_cov.yaml | cloud-coverage\_v1.0.nc |
|  |  | ncas-lidar-dop-3\_cao\_*yyyymmdd*\_... |
| Halo lidar co-polar standard | ncas-lidar-dop-3-co-lv1\_template.yaml | aerosol-backscatter-radial-winds\_fixed\_co\_standard\_v1.0.nc |
| Halo lidar cross-polar standard | ncas-lidar-dop-3-cr-lv1\_template.yaml | aerosol-backscatter-radial-winds\_fixed\_cr\_standard\_v1.0.nc |
| Halo lidar VAD standard | ncas-lidar-dop-3-co-lv1-vad\_template.yaml | aerosol-backscatter-radial-winds\_vad\_standard\_v1.0.nc |
| Halo lidar co-polar advanced | ncas-lidar-dop-3-co-lv2\_template.yaml | aerosol-backscatter-radial-winds\_fixed\_co\_advanced\_v1.0.nc |
| Halo lidar VAD advanced | ncas-lidar-dop-3-co-lv2-vad\_template.yaml | aerosol-backscatter-radial-winds\_vad\_advanced\_v1.0.nc |

1. A plot is produced of the data using the lidar\_plotting module in lidar\_plot.py.

## Assignment of quality flags

There are some differences between how quality flags are assigned for profile measurements between the CL51 ceilometer and the Halo Doppler lidar, so the processes will be described separately.

### CL51 ceilometer attenuated aerosol backscattering coefficient quality flag

The same quality flags are used for the standard and advanced files, since the only difference between the 2 datasets is the application of a calibration factor to the advanced file profiles.

1. We want to assess the profile background level without a range2 correction applied. This correction is applied within the DAQ software if using the mode where range2 scaling is applied, as we do. It is set by sending the “set message profile noise h2 on” command to the CL51. Divide backscatter data by h2 variable to remove range2 correction, producing the braw\_nor2 variable. Problem for profiles with no increase in noise with h?
2. Work out a set of range bins in 100 bin blocks from the top of the profile and evaluate the variance in those bins, bgvar.
3. For each profile, check:
4. if the first cloud base is above zero (i.e. a 1st cloud base was recorded) and that there are less than 4 cloud bases and that all recorded cloud bases are below 3000 m, or
5. The number of cloud bases equals 4, indicating a vertical visibility measurement, or
6. The measured backscatter sum is > 300.
7. If the above condition is satisfied, evaluate and save for that profile the maximum variance in array braw\_nor2 from 100 m above the highest cloud. Use that value to replace the value worked out in step 2.
8. Evaluate the signal to noise ratio (SNR) for each point in braw\_nor2 as the value of braw\_nor2/(bgvar)1/2.
9. Loop through each height bin in each profile and compare the SNR with a threshold value hard-wired into the code as snr\_threshold. If SNR is above the threshold the qc\_flag is assigned as 1 if the profile was recorded at high gain (the normal mode) or 2 if the profile was recorded at low gain (used when there is high backscatter from fog, low cloud or rain). The gain setting is indicated in the housekeeping data.
10. Also if SNR is above snr\_threshold, a check is made for speckle. This is defined as a point above snr\_threshold where at least 3 of the 4 adjacent points are below snr\_threshold. If the point is identified as speckle, bs\_flag is set to 6.
11. If SNR is less than snr\_threshold, bs\_flag is set to 3 for a high gain profile or 4 for a low gain profile.
12. If the as-recorded backscattering coefficient is less than zero, bs\_flag is set to 5. This is usually due to instrument noise.
13. If the recorded backscattering coefficient was missing due to an instrument warning or alarm, bs\_flag is set to 7.

### CL51 ceilometer loud base height and coverage quality flags

For the cloud base height file, the qc\_flag is used to write the number of cloud bases or the visibility flag, as they are defined in the instrument manual. The qc\_flag value is actually (number\_of\_cloud\_bases) + 1, so that qc\_flag = 1 implies no cloud bases detected. A qc\_flag of 7 indicates that there was an error in the recording of that profile and/or the derivation of cloud base height(s).

If the sky condition algorithm has insufficient data to report a cloud coverage value, for example when data acquisition has started very recently, the cloud coverage qc\_flag is set to 2. If the sky condition algorithm was not operating, the cloud coverage qc\_flag is set to 3.

Halo Doppler lidar attenuated aerosol backscattering coefficient and radial velocity quality flags

The process of finding a background variance cannot be so easily applied to the Halo lidar as to the ceilometer, as the maximum range is 9.6 km rather than 15 km. This means that there is too high a likelihood of cloud contamination at the top of the profile to be able to reliably calculate the variance of the background signal.

Different flags are set for backscattering coefficient and radial velocity. No quality flag is currently written for (SNR+1) as it is not clear what flags should be applied.

Separate fractional errors for backscattering coefficient and radial velocity are reported in the cleaned netCDF file produced by the FMI toolbox. These allow for further quality flags to be produced for the “advanced” files, based on those data. The FMI files also report that that the SNR data have the same fractional uncertainty same as backscattering data, but as yet this has not been applied to the SNR+1 data.

1. If the backscattering coefficient is less than zero qc\_flag is set to 3 in as-recorded “standard” files. The as-recorded data are superimposed on a background which becomes increasingly negative with height, an artefact of how data are recorded by the instrument. This filter is not applied for FMI toolbox “advanced” files as that background is not present.
2. If the backscattering coefficient is less than 1.0 E-7 or greater than 1000 this is considered outside a feasible measurement range and qc\_flag is set to 2, over-riding any values set in the previous step.
3. If there is a step of more than 5 m/s in velocity between adjacent 30 m bins the velocity qc\_flag is set to 3.
4. If the as-recorded velocity is outside the range -19 m/s to +19 m/s this is beyond what can be measured by the system and the velocity qc\_flag is set to 2, over-riding any values set in the previous step.
5. For cleaned files, if the fractional error is greater than 1.0, qc\_flag is set to 3, 4 respectively for backscattering coefficient and radial velocity. This may sound like a high uncertainty threshold, but signals from tenuous clouds can be noisy but we don’t want to over-filter them.
6. For cleaned files, the same speckle reduction algorithm as is described in step 6 for the ceilometer is applied, except that the search used is for at least 3 of the 4 adjacent points having a fractional error above the uncertainty threshold while the current point does not. If this speckle condition is met, the qc\_flag is set to 4, 5 respectively for backscattering coefficient and radial velocity.
7. If cleaned profiles could not be matched to corresponding as-recorded ones, the qc\_flag is set to 5,6 respectively for backscattering coefficient and radial velocity.

## Plotting lidar images – functions in the lidar\_plot module

### Time-height profile measurements

The function lidar\_plotting in the lidar\_plot.py code is used to plot profile measurements from both lidars. The numeric day, the lidar type (1 for CL51 ceilometer, 2 for Halo Doppler lidar) and the data type (1 for standard, 2 for advanced) are passed to it. The code is called from the end of the above halo\_python.py code or can be run independently if the netCDF files for the day have already been produced. In the latter case, it is run using the generate\_days\_netcdf\_lidar\_graphs.py wrapper.

The input information is used to generate the path and filename of the netCDF file to read and also those for output .png files. The latter are sorted into directories by year, month and the variable that is plotted. The names of the variables to read from a ceilometer or lidar file and their respective quality flags are hard-wired into the code.

The PyART module, produced by ARM, is imported at the start of the code and is used to provide some functions and colormaps.

The plots are produced using the Matplotlib pcolormesh command. The backscattering coefficient and (SNR+1) plots used the HomeyerRainbow colormap, which is designed to produce plots which are easier to interpret by people who are colour-blind. This colormap is less helpful for the radial velocity plots where a range of values centred on zero is to be displayed. The Matplotlib coolwarm colormap is used for these plots. The colormaps are hard-wired into the code but can be changed easily. The ranges of the colormaps are also hard-wired. These could be moved to the command line if desired.

For each variable, a pair of plots is produced, with and without the quality flag applied. Where the quality flag is applied, all points with a quality flag greater than 1 are excluded from the plot. The Halo lidar backscattering coefficient qc\_flag is used for (SNR+1) plots as a separate flag is not written for that variable. For radial velocity plots, the qc\_flag for backscattering coefficient is added to that for radial velocity, then a value of 1 is subtracted from the sum. This has the effect of combining the flags to provide more stringent masking of the velocity data. Attributes from the netCDF files are used to label the plots.

### Cloud base height and cloud coverage measurements

The cloud-base height and cloud coverage plots are time-series plots showing several datasets on the same axes, produced using the lidar\_plotting\_time\_series function. The Matplotlib “plot” function is used to provide the plots. The names of files to read and write are generated in a similar way to those for time-height plots.

Cloud coverage is reported on a scale of 1 – 8 oktas for each cloud base that is detected by the sky condition algorithm. It is displayed for each layer on a time series plot, together with the cloud base height for that layer as derived by the cloud coverage algorithm.

## generate\_netcdf\_common function

The numeric version of the day is passed to this function. It returns the date in various formats which are used later in the code.

## halo\_check\_monotonic function

This looks for indices in the Halo lidar time variable where the value is not monotonic and returns the index at which a negative time increment is spotted. It appends details of the file where the negative time increment is found to /data/amof-netCDF/ncas-lidar-dop-3/halo\_errors.txt so that a log is kept of when this feature occurs. It also writes the profiles after the negative time increment to /data/amof-netCDF/ncas-lidar-dop-3/halo\_midnight\_*yyyymmdd*\_*n\_mono*\_v2.txt where *n\_mono* is 0 for a fixed co-polar file, 1 for a fixed cross-polar file and 2 for VAD files.

## my\_broadcast function

This function broadcasts (expands) Halo lidar data arrays from 2 to 3 dimensions, where the 3rd dimension is equal to the number of azimuth angles recorded (currently 1 azimuth for a fixed scan, 24 for a VAD).

## Production of webpages to display quicklooks

Scripts using Webmagick are used to produce a directory structure of quicklooks of all plots. These are described in a separate document.

## Future possible addition for Halo lidar data

Some housekeeping data for the Halo lidar are recorded separately to the lidar data files. They are stored on the lidar DAQ PC in C:\Lidar\Data\Proc\*yyyy*\*yyyymm* as a file named system\_parameters\_118\_*yyyymm*.dat. They are not currently recorded on G: but this could be changed. It would be helpful to include these housekeeping data in the netCDF files.