SCC NetCDF input files structure

version:3.2

March 9, 2021

The Single Calculus Chain (SCC) is composed by three different modules:

- pre-processing module (*ELPP*)
- optical processing module (*ELDA*)
- depolarization calibrator module (*ELDEC*)

To perfom aerosol optical retrievals the SCC needs not only the raw lidar data but also a certain number of parameters to use in both pre-processing and optical processing stages. The SCC gets these parameters looking at two different locations:

- Single Calculus Chain relational database (SCC_DB)
- Input files

There are some paramenters that can be found only in the input files (those ones changing from measurement to measurement), others that can be found only in the SCC_DB and other ones that can be found in both these locations. In the last case, if a particular parameter is needed, the SCC will search first in the input files and then in SCC_DB. If the parameter is found in the input files the SCC will keep it without looking into SCC_DB.

The input files have to be submitted to the SCC in NetCDF format. At present the SCC can handle four different types of input files:

1. *Raw Lidar Data*
2. *Sounding Data*
3. *Overlap*
4. *Lidar Ratio*

As already mentioned, the *Raw Lidar Data* file contains not only the raw lidar data but also other parameters to use to perform the pre-processing and optical processing. The *Sounding Data* file contains the data coming from a correlative radiosounding and it is used by the SCC for molecular density calculation. The *Overlap* file contains the measured overlap function. The *Lidar Ratio* file contains a lidar ratio profile to use in elastic backscatter retrievals. The *Raw Lidar Data* file is of course mandatory and the *Sounding Data*, *Overlap* and *Lidar Ratio* files are optional. If *Sounding Data* file is not submitted by the user, the molecular density will be calculated by the SCC using the "US Standard Atmosphere 1976". If the *Overlap* file is not submitted by the user, the SCC will get the full overlap height from SCC_DB and it will produce optical results starting from this height. If *Lidar Ratio* file is not submitted by the user, the SCC will consider a fixed value for lidar ratio got from SCC_DB.

The user can decide to submit all these files or any number of them (of course the file *Raw Lidar Data* is mandatory). For example the user can submit together with the *Raw Lidar Data* file only the *Sounding Data* file or only the *Overlap* file.

This document provides a detailed explanation about the structure of the NetCDF input files to use for SCC data submission. All Earlinet groups should read it carefully

because they have to produce such kind of input files if they want to use the SCC for their standard lidar retrievals. Every comments or suggestions regarding this document can be sent to Giuseppe D'Amico by e-mail at `damico@imaa.cnr.it`
This document is available for downloading at `http://scc-documentation.readthedocs.io/en/lates`

In table 1 is reported a list of dimensions, variables and global attributes that can be used in the NetCDF *Raw Lidar Data* input file. For each of them it is indicated:

- The name. For the multidimensional variables also the corresponding dimensions are reported
- A description explaining the meaning
- The type
- If it is mandatory or optional

As already mentioned, the SCC can get some parameters looking first in the *Raw Lidar Data* input file and then into SCC_DB. This means that to use the parameters stored in SCC_DB the optional variables or optional global attributes must not appear within *Raw Lidar Data* file. This is the suggested and recommended way to use the SCC. Please include optional parameters in the *Raw Lidar Data* only as an exception.

In table 2, 3 and 4 are reported all the information about the structure of *Sounding Data*, *Overlap* and *Lidar Ratio* input files respectively.

1 Example

Let's now consider an example of *Raw Lidar Data* input file. Suppose we want to generate NetCDF input file corresponding to a measurement with the following properties:

Start Date	30^{th} January 2009
Start Time UT	00:00:01
Stop Time UT	00:05:01
Station Name	Dummy station
Earlinet call-sign	cc
Pointing angle	5 degrees with respect to the zenith

Moreover suppose that this measurement is composed by the following lidar channels:

1. 1064 lidar channel

Emission wavelength=1064nm	Detection wavelength=1064nm
Time resolution=30s	Number of laser shots=1500
Number of bins=3000	Detection mode=analog
Range resolution=7.5m	Polarization state=total

2. 532 cross lidar channel

Emission wavelength=532nm	Detection wavelength=532nm
Time resolution=60s	Number of laser shots=3000
Number of bins=5000	Detection mode=photoncounting
Range resolution=15m	Polarization state=cross (transmitted)

3. 532 parallel lidar channel

Emission wavelength=532nm	Detection wavelength=532nm
Time resolution=60s	Number of laser shots=3000
Number of bins=5000	Detection mode=photoncounting
Range resolution=15m	Polarization state=parallel (reflected)

4. 607 N_2 vibrational Raman channel

Emission wavelength=532nm	Detection wavelength=607nm
Time resolution=60s	Number of laser shots=3000
Number of bins=5000	Detection mode=photoncounting
Range resolution=15m	

Finally let's assume we have also performed dark measurements before the lidar measurements from the 23:50:01 UT up to 23:53:01 UT of 29th January 2009.

1.1 Dimensions

Looking at table 1 we have to fix the following dimensions:

```
points
channels
time
nb_of_time_scales
scan_angles
time_bck
```

The dimension `time` is unlimited so we don't have to fix it.
We have 4 lidar channels so:

```
channels=4
```

Regarding the dimension `points` we have only one channel with a number of vertical bins equal to 3000 (the 1064nm) and all other channels with 5000 vertical bins. In cases like this the dimension `points` has to be fixed to the maximum number of vertical bins so:

```
points=5000
```

Moreover only one channel (1064nm) is acquired with a time resolution of 30 seconds, all the other channels have a time resolution of 60 seconds. This means that we have to define two different time scales. We have to set:

```
nb_of_time_scales=2
```

The measurement is performed only at one scan angle (5 degrees with respect to the zenith) so:

```
scan_angles=1
```

We have 3 minutes of dark measurements and two different time scales one with 60 seconds time resolution and the other one with 30 seconds time resolution. So we will have 3 different dark profiles for the channels acquired with the first time scale and 6 for the lidar channels acquired with the second time scale. We have to fix the dimension `time_bck` as the maximum between these values:

```
time_bck=6
```

1.2 Variables

In this section it will be explained how to fill all the possible variables either mandatory or optional of *Raw Lidar Data* input file.

- `Raw_Data_Start_Time(time, nb_of_time_scales)`
  This 2 dimensional mandatory array has to contain the acquisition start time (in seconds from the time given by the global attribute `RawData_Start_Time_UT`) of each lidar profile. In this example we have two different time scales: one is characterized by steps of 30 seconds (the 1064nm is acquired with this time scale) the other by steps of 60 seconds (532cross, 532parallel and 607nm). Moreover the measurement start time is 00:00:01 UT and the measurement stop time is 00:05:01 UT. In this case we have to define:

  ```
  Raw_Data_Start_Time =
    0, 0,
    60, 30,
    120, 60,
    180, 90,
    240, 120,
    _, 150,
    _, 180,
    _, 210,
    _, 240,
    _, 270 ;
  ```

  The order used to fill this array defines the correspondence between the different time scales and the time scale index. In this example we have a time scale index of 0 for the time scale with steps of 60 seconds and a time scale index of 1 for the other one.

- `Raw_Data_Stop_Time(time, nb_of_time_scales)`
  The same as previous item but for the data acquisition stop time. Following a similar procedure we have to define:

  ```
  Raw_Data_Stop_Time =
    60, 30,
    120, 60,
    180, 90,
    240, 120,
    300, 150,
    _, 180,
    _, 210,
    _, 240,
    _, 270,
    _, 300 ;
  ```

- `Raw_Lidar_Data(time, channels, points)`
  This 3 dimensional mandatory array has to be filled with the time-series of raw lidar data. The photoncounting profiles have to submitted in counts (so as integers)

while the analog ones in mV. The order the user chooses to fill this array defines the correspondence between channel index and lidar data.
For example if we fill this array in such way that:
`Raw_Lidar_Data(time,0,points)` → is the time-series of 1064 nm
`Raw_Lidar_Data(time,1,points)` → is the time-series of 532 cross
`Raw_Lidar_Data(time,2,points)` → is the time-series of 532 parallel
`Raw_Lidar_Data(time,3,points)` → is the time-series of 607 nm
from now on the channel index 0 is associated to the 1064 channel, 1 to the 532 cross, 2 to the 532 parallel and 3 to the 607nm.

- `Raw_Bck_Start_Time(time_bck, nb_of_time_scales)`
  This 2 dimensional optional array has to contain the acquisition start time (in seconds from the time given by the global attribute `RawBck_Start_Time_UT`) of each dark measurements profile. Following the same procedure used for the variable `Raw_Data_Start_Time` we have to define:

```
Raw_Bck_Start_Time =
  0, 0,
  60, 30,
  120, 60,
  _, 90,
  _, 120,
  _, 150;
```

- `Raw_Bck_Stop_Time(time_bck, nb_of_time_scales)`
  The same as previous item but for the dark acquisition stop time. Following a similar procedure we have to define:

```
Raw_Bck_Stop_Time =
  60, 30,
  120, 60,
  180, 90,
  _, 120,
  _, 150,
  _, 180 ;
```

- `Background_Profile(time_bck, channels, points)`
  This 3 dimensional optional array has to be filled with the time-series of the dark measurements data. The photoncounting profiles have to submitted in counts (so as integers) while the analog ones in mV. The user has to fill this array following the same order used in filling the array `Raw_Lidar_Data`:
  `Background_Profile(time_bck,0,points)` → dark time-series at 1064 nm
  `Background_Profile(time_bck,1,points)` → dark time-series at 532 cross
  `Background_Profile(time_bck,2,points)` → dark time-series at 532 parallel
  `Background_Profile(time_bck,3,points)` → dark time-series at 607 nm

- `channel_ID(channels)`
  This mandatory array provides the link between the channel index within the *Raw Lidar Data* input file and the channel ID in SCC_DB. To fill this variable the user has to know which channel IDs in SCC_DB correspond to his lidar channels. For

this purpose the SCC, in its final version will provide to the user a special tool to get these channel IDs through a Web interface. At the moment this interface is not yet available and these channel IDs will be communicated directly to the user by the NA5 people.
Anyway to continue the example let's suppose that the four lidar channels taken into account are mapped into SCC_DB with the following channel IDs:

1064 nm	→ channel ID=7
532 cross	→ channel ID=5
532 parallel	→ channel ID=6
607 nm	→ channel ID=8

In this case we have to define:

```
channel_ID = 7, 5, 6, 8 ;
```

- `id_timescale(channels)`
  This mandatory array is introduced to determine which time scale is used for the acquisition of each lidar channel. In particular this array defines the link between the channel index and the time scale index. In our example we have two different time scales. Filling the arrays `Raw_Data_Start_Time` and `Raw_Data_Stop_Time` we have defined a time scale index of 0 for the time scale with steps of 60 seconds and a time scale index of 1 for the other one with steps of 30 seconds. In this way this array has to be set as:

```
id_timescale = 1, 0, 0, 0 ;
```

- `Laser_Pointing_Angle(scan_angles)`
  This mandatory array contains all the scan angles used in the measurement. In our example we have only one scan angle of 5 degrees with respect to the zenith, so we have to define:

```
Laser_Pointing_Angle = 5 ;
```

- `Laser_Pointing_Angle_of_Profiles(time, nb_of_time_scales)`
  This mandatory array is introduced to determine which scan angle is used for the acquisition of each lidar profile. In particular this array defines the link between the time and time scales indexes and the scan angle index. In our example we have a single scan angle that has to correspond to the scan angle index 0. So this array has to be defined as:

```
Laser_Pointing_Angle_of_Profiles =
  0, 0,
  0, 0,
  0, 0,
  0, 0,
  0, 0,
  _, 0,
  _, 0,
  _, 0,
  _, 0,
  _, 0 ;
```

- `Laser_Shots(time, channels)`
  This mandatory array stores the laser shots accumulated at each time for each channel. In our example the number of laser shots accumulated is 1500 for the 1064nm channels and 3000 for all the other channels. Moreover the laser shots do not change with the time. So we have to define this array as:

```
 Laser_Shots =
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, _, _, _,
  1500, _, _, _,
  1500, _, _, _,
  1500, _, _, _,
  1500, _, _, _ ;
```

- `Emitted_Wavelength(channels)`
  This optional array defines the link between the channel index and the emission wavelength for each lidar channel. The wavelength has to be expressed in nm. This information can be also taken from SCC_DB. In our example we have:

```
Emitted_Wavelength = 1064, 532, 532, 532 ;
```

- `Detected_Wavelength(channels)`
  This optional array defines the link between the channel index and the detected wavelength for each lidar channel. Here detected wavelength means the value of center of interferential filter expressed in nm. This information can be also taken from SCC_DB. In our example we have:

```
Detected_Wavelength = 1064, 532, 532, 607 ;
```

- `Raw_Data_Range_Resolution(channels)`
  This optional array defines the link between the channel index and the raw range resolution for each channel. If the scan angle is different from zero this quantity is different from the vertical resolution. More precisely if α is the scan angle used and Δz is the range resolution the vertical resolution is calculated as $\Delta z' = \Delta z \cos\alpha$. This array has to be filled with Δz and not with $\Delta z'$. The unit is meters. This information can be also taken from SCC_DB. In our example we have:

```
Raw_Data_Range_Resolution = 7.5, 15.0, 15.0, 15.0 ;
```

- `Scattering_Mechanism(channels)`
  This optional array defines the scattering mechanism involved in each lidar channel. In particular the following values are adopted:

0 → Total elastic backscatter
1 → N_2 vibrational Raman backscatter
2 → Cross polarization elastic backscatter
3 → Parallel polarization elastic backscatter
4 → H_2O vibrational Raman backscatter
5 → Rotational Raman low quantum number
6 → Rotational Raman high quantum number

This information can be also taken from SCC_DB. In our example we have:

```
Scattering_Mechanism = 0, 2, 3, 1 ;
```

- `Signal_Type(channels)`
  This optional array defines the type of signal involved in each lidar channel. In particular the following values are adopted:
  0 → Total elastic
  1 → Total elastic near range
  2 → Total elastic far range
  3 → N_2 vibrational Raman
  4 → N_2 vibrational Raman near range
  5 → N_2 vibrational Raman far range
  6 → Elastic polarization reflected
  7 → Elastic polarization transmitted
  8 → Rotational Raman line close to elastic line
  9 → Rotational Raman line far from elastic line
  10 → Elastic polarization reflected near range
  11 → Elastic polarization reflected far range
  12 → Elastic polarization transmitted near range
  13 → Elastic polarization transmitted far range
  14 → H_2O vibrational Raman backscatter
  15 → Rotational Raman line far from elastic line near range
  16 → Rotational Raman line far from elastic line far range
  17 → Rotational Raman line close to elastic line near range
  18 → Rotational Raman line close to elastic line far range
  19 → H_2O vibrational Raman backscatter near range
  20 → H_2O vibrational Raman backscatter far range
  21 → Total elastic ultra near range
  22 → +45 rotated elastic polarization transmitted
  23 → +45 rotated elastic polarization reflected
  24 → -45 rotated elastic polarization transmitted
  25 → -45 rotated elastic polarization reflected
  26 → +45 rotated elastic polarization transmitted near range
  27 → +45 rotated elastic polarization transmitted far range
  28 → +45 rotated elastic polarization reflected near range
  29 → +45 rotated elastic polarization reflected far range
  30 → -45 rotated elastic polarization transmitted near range
  31 → -45 rotated elastic polarization transmitted far range
  32 → -45 rotated elastic polarization reflected near range
  33 → -45 rotated elastic polarization reflected far range

  This information can be also taken from SCC_DB. In our example we have:

```
Signal_Type = 0, 7, 6, 3 ;
```

- `Acquisition_Mode(channels)`
  This optional array defines the acquisition mode (analog or photoncounting) involved in each lidar channel. In particular a value of 0 means analog mode and 1 photoncounting mode. This information can be also taken from SCC_DB. In our example we have:

```
Acquisition_Mode = 0, 1, 1, 1 ;
```

- `Laser_Repetition_Rate(channels)`
  This optional array defines the repetition rate in Hz used to acquire each lidar channel. This information can be also taken from SCC_DB. In our example we are supposing we have only one laser with a repetition rate of 50 Hz so we have to set:

```
Laser_Repetition_Rate = 50, 50, 50, 50 ;
```

- `Dead_Time(channels)`
  This optional array defines the dead time in ns associated to each lidar channel. The SCC will use the values given by this array to correct the photoncounting signals for dead time. Of course for analog signals no dead time correction will be applied (for analog channels the corresponding dead time values have to be set to undefined value). This information can be also taken from SCC_DB. In our example the 1064 nm channel is acquired in analog mode so the corresponding dead time value has to be undefined. If we suppose a dead time of 10 ns for all other channels we have to set:

```
Dead_Time = _, 10, 10, 10 ;
```

- `Dead_Time_Corr_Type(channels)`
  This optional array defines which kind of dead time correction has to be applied on each photoncounting channel. The SCC will correct the data supposing a not-paralyzable channel if a value of 0 is found while a paralyzable channel is supposed if a value of 1 is found. Of course for analog signals no dead time correction will be applied and so the corresponding values have to be set to undefined value. This information can be also taken from SCC_DB. In our example the 1064 nm channel is acquired in analog mode so the corresponding has to be undefined. If we want to consider all the photoncounting signals as not-paralyzable ones: we have to set:

```
Dead_Time_Corr_Type = _, 0, 0, 0 ;
```

- `Trigger_Delay(channels)`
  This optional array defines the delay (in ns) of the middle of the first rangebin with respect to the output laser pulse for each lidar channel. The SCC will use the values given by this array to correct for trigger delay. This information can be also taken from SCC_DB. Let's suppose that in our example all the photoncounting channels are not affected by this delay and only the analog channel at 1064nm is acquired with a delay of 50ns. In this case we have to set:

```
Trigger_Delay = 50, 0, 0, 0 ;
```

- `Background_Mode(channels)`
  This optional array defines how the atmospheric background has to be subtracted from the lidar channel. Two options are available for the calculation of atmospheric background:

  1. Average in the far field of lidar channel. In this case the value of this variable has to be 1
  2. Average within pre-trigger bins. In this case the value of this variable has to be 0

  This information can be also taken from SCC_DB. Let's suppose in our example we use the pre-trigger for the 1064nm channel and the far field for all other channels. In this case we have to set:

  ```
  Background_Mode = 0, 1, 1, 1 ;
  ```

- `Background_Low(channels)`
  This mandatory array defines the minimum altitude (in meters) to consider in calculating the atmospheric background for each channel. In case pre-trigger mode is used the corresponding value has to be set to the rangebin to be used as lower limit (within pre-trigger region) for background calculation. In our example, if we want to calculate the background between 30000 and 50000 meters for all photoncounting channels and we want to use the first 500 pre-trigger bins for the background calculation for the 1064nm channel we have to set:

  ```
  Background_Low= 0, 30000, 30000, 30000 ;
  ```

- `Background_High(channels)`
  This mandatory array defines the maximum altitude (in meters) to consider in calculating the atmospheric background for each channel. In case pre-trigger mode is used the corresponding value has to be set to the rangebin to be used as upper limit (within pre-trigger region) for background calculation. In our example, if we want to calculate the background between 30000 and 50000 meters for all photoncounting channels and we want to use the first 500 pre-trigger bins for the background calculation for the 1064nm channel we have to set:

  ```
  Background_High = 500, 50000, 50000, 50000 ;
  ```

- `Molecular_Calc`
  This mandatory variable defines the way used by SCC to calculate the molecular density profile. At the moment two options are available:

  1. US Standard Atmosphere 1976. In this case the value of this variable has to be 0
  2. Radiosounding. In this case the value of this variable has to be 1

  If we decide to use the option 1. we have to provide also the measured pressure and temperature at lidar station level. Indeed if we decide to use the option 2. a radiosounding file has to be submitted separately in NetCDF format (the structure of this file is summarized in table 2). Let's suppose we want to use the option 1. so:

```
Molecular_Calc = 0 ;
```

- `Pressure_at_Lidar_Station`
  Because we have chosen the US Standard Atmosphere for calculation of the molecular density profile we have to give the pressure in hPa at lidar station level:

  ```
  Pressure_at_Lidar_Station = 1010 ;
  ```

- `Temperature_at_Lidar_Station`
  Because we have chosen the US Standard Atmosphere for calculation of the molecular density profile we have to give the temperature in C at lidar station level:

  ```
  Temperature_at_Lidar_Station = 19.8 ;
  ```

- `LR_Input(channels)`
  This array is required only for lidar channels for which elastic backscatter retrieval has to be performed. It defines the lidar ratio to be used within this retrieval. Two options are available:

  1. The user can submit a lidar ratio profile. In this case the value of this variable has to be 0.
  2. A fixed value of lidar ratio can be used. In this case the value of this variable has to be 1.

  If we decide to use the option 1. a lidar ratio file has to be submitted separately in NetCDF format (the structure of this file is summarized in table 4). If we decide to use the option 2. the fixed value of lidar ratio will be taken from SCC_DB. In our example we have to give a value of this array only for the 1064nm lidar channel because for the 532nm we will be able to retrieve a Raman backscatter coefficient. In case we want to use the fixed value stored in SCC_DB we have to set:

  ```
  LR_Input = 1,_,_,_ ;
  ```

- `DAQ_Range(channels)`
  This array is required only if one or more lidar signals are acquired in analog mode. It gives the analog scale in mV used to acquire the analog signals. In our example we have only the 1064nm channel acquired in analog mode. If we have used a 100mV analog scale to acquire this channel we have to set:

  ```
  DAQ_Range = 100,_,_,_ ;
  ```

1.3 Global attributes

- `Measurement_ID`
  This mandatory global attribute defines the measurement ID corresponding to the actual lidar measurement. It is a string composed by 12 characters. The first 8 characters give the start date of measurement in the format YYYYMMDD. The next 2 characters give the Earlinet call-sign of the station. The last 2 characters are used to distinguish between different time-series within the same date. In our example we have to set:

```
Measurement_ID= "20090130cc00" ;
```

- `RawData_Start_Date`
  This mandatory global attribute defines the start date of lidar measurements in the format YYYYMMDD. In our case we have:

  ```
  RawData_Start_Date = "20090130" ;
  ```

- `RawData_Start_Time_UT`
  This mandatory global attribute defines the UT start time of lidar measurements in the format HHMMSS. In our case we have:

  ```
  RawData_Start_Time_UT = "000001" ;
  ```

- `RawData_Stop_Time_UT`
  This mandatory global attribute defines the UT stop time of lidar measurements in the format HHMMSS. In our case we have:

  ```
  RawData_Stop_Time_UT = "000501" ;
  ```

- `RawBck_Start_Date`
  This optional global attribute defines the start date of dark measurements in the format YYYYMMDD. In our case we have:

  ```
  RawBck_Start_Date = "20090129" ;
  ```

- `RawBck_Start_Time_UT`
  This optional global attribute defines the UT start time of dark measurements in the format HHMMSS. In our case we have:

  ```
  RawBck_Start_Time_UT = "235001" ;
  ```

- `RawBck_Stop_Time_UT`
  This optional global attribute defines the UT stop time of dark measurements in the format HHMMSS. In our case we have:

  ```
  RawBck_Stop_Time_UT = "235301" ;
  ```

1.4 Example of file (CDL format)

To summarize we have the following NetCDF *Raw Lidar Data* file (in CDL format):

```
dimensions:
        points = 5000 ;
        channels = 4 ;
        time = UNLIMITED ; // (10 currently)
        nb_of_time_scales = 2 ;
        scan_angles = 1 ;
        time_bck = 6 ;
```

```
variables:
        int channel_ID(channels) ;
        int Laser_Repetition_Rate(channels) ;
        double Laser_Pointing_Angle(scan_angles) ;
        int Signal_Type(channels);
        double Emitted_Wavelength(channels) ;
        double Detected_Wavelength(channels) ;
        double Raw_Data_Range_Resolution(channels) ;
        int Background_Mode(channels) ;
        double Background_Low(channels) ;
        double Background_High(channels) ;
        int Molecular_Calc ;
        double Pressure_at_Lidar_Station ;
        double Temperature_at_Lidar_Station ;
        int id_timescale(channels) ;
        double Dead_Time(channels) ;
        int Dead_Time_Corr_Type(channels) ;
        int Acquisition_Mode(channels) ;
        double Trigger_Delay(channels) ;
        int LR_Input(channels) ;
        int Laser_Pointing_Angle_of_Profiles(time, nb_of_time_scales) ;
        int Raw_Data_Start_Time(time, nb_of_time_scales) ;
        int Raw_Data_Stop_Time(time, nb_of_time_scales) ;
        int Raw_Bck_Start_Time(time_bck, nb_of_time_scales) ;
        int Raw_Bck_Stop_Time(time_bck, nb_of_time_scales) ;
        int Laser_Shots(time, channels) ;
        double Raw_Lidar_Data(time, channels, points) ;
        double Background_Profile(time_bck, channels, points) ;
        double DAQ_Range(channels) ;

// global attributes:
                :Measurement_ID = "20090130cc00" ;
                :RawData_Start_Date = "20090130" ;
                :RawData_Start_Time_UT = "000001" ;
                :RawData_Stop_Time_UT = "000501" ;
                :RawBck_Start_Date = "20090129" ;
                :RawBck_Start_Time_UT = "235001" ;
                :RawBck_Stop_Time_UT = "235301" ;

data:

 channel_ID = 7, 5, 6, 8 ;

 Laser_Repetition_Rate = 50, 50, 50, 50 ;

 Laser_Pointing_Angle = 5 ;

 Signal_Type = 0, 7, 6, 3 ;
```

```
Emitted_Wavelength = 1064, 532, 532, 532 ;

Detected_Wavelength = 1064, 532, 532, 607 ;

Raw_Data_Range_Resolution = 7.5, 15, 15, 15 ;

Background_Mode = 0, 1, 1, 1 ;

Background_Low = 0, 30000, 30000, 30000 ;

Background_High = 500, 50000, 50000, 50000 ;

Molecular_Calc = 0 ;

Pressure_at_Lidar_Station = 1010 ;

Temperature_at_Lidar_Station = 19.8 ;

id_timescale = 1, 0, 0, 0 ;

Dead_Time = _, 10, 10, 10 ;

Dead_Time_Corr_Type = _, 0, 0, 0 ;

Acquisition_Mode = 0, 1, 1, 1 ;

Trigger_Delay = 50, 0, 0, 0 ;

LR_Input = 1,_,_,_ ;

DAQ_Range = 100,_,_,_ ;

Laser_Pointing_Angle_of_Profiles =
 0, 0,
 0, 0,
 0, 0,
 0, 0,
 0, 0,
 _, 0,
 _, 0,
 _, 0,
 _, 0,
 _, 0 ;


Raw_Data_Start_Time =
 0, 0,
 60, 30,
 120, 60,
```

```
  180, 90,
  240, 120,
  _, 150,
  _, 180,
  _, 210,
  _, 240,
  _, 270 ;

 Raw_Data_Stop_Time =
  60, 30,
  120, 60,
  180, 90,
  240, 120,
  300, 150,
  _, 180,
  _, 210,
  _, 240,
  _, 270,
  _, 300 ;


 Raw_Bck_Start_Time =
  0, 0,
  60, 30,
  120, 60,
  _, 90,
  _, 120,
  _, 150;


Raw_Bck_Stop_Time =
  60, 30,
  120, 60,
  180, 90,
  _, 120,
  _, 150,
  _, 180 ;


 Laser_Shots =
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, 3000, 3000, 3000,
  1500, _, _, _,
  1500, _, _, _,
  1500, _, _, _,
  1500, _, _, _,
```

```
  1500, _, _, _ ;


 Raw_Lidar_Data = ...

 Background_Profile = ...
```

The name of the input file should have the following format:

`Measurement_ID.nc`

so in the example the filename should be 20090130cc00.nc.

Please keep in mind that in case you submit a file like the previous one all the parameters present in it will be used by the SCC even if you have different values for the same parameters within the SCC_DB. If you want to use the values already stored in SCC_DB (this should be the usual way to use SCC) the *Raw Lidar Data* input file has to be modified as follows:

```
dimensions:
        points = 5000 ;
        channels = 4 ;
        time = UNLIMITED ; // (10 currently)
        nb_of_time_scales = 2 ;
        scan_angles = 1 ;
        time_bck = 6 ;
variables:
        int channel_ID(channels) ;
        double Laser_Pointing_Angle(scan_angles) ;
        double Background_Low(channels) ;
        double Background_High(channels) ;
        int Molecular_Calc ;
        double Pressure_at_Lidar_Station ;
        double Temperature_at_Lidar_Station ;
        int id_timescale(channels) ;
        int Laser_Pointing_Angle_of_Profiles(time, nb_of_time_scales) ;
        int Raw_Data_Start_Time(time, nb_of_time_scales) ;
        int Raw_Data_Stop_Time(time, nb_of_time_scales) ;
        int Raw_Bck_Start_Time(time_bck, nb_of_time_scales) ;
        int Raw_Bck_Stop_Time(time_bck, nb_of_time_scales) ;
        int LR_Input(channels) ;
        int Laser_Shots(time, channels) ;
        double Raw_Lidar_Data(time, channels, points) ;
        double Background_Profile(time_bck, channels, points) ;
        double DAQ_Range(channels) ;

// global attributes:
                :Measurement_ID = "20090130cc00" ;
                :RawData_Start_Date = "20090130" ;
                :RawData_Start_Time_UT = "000001" ;
```

```
                :RawData_Stop_Time_UT = "000501" ;
                :RawBck_Start_Date = "20090129" ;
                :RawBck_Start_Time_UT = "235001" ;
                :RawBck_Stop_Time_UT = "235301" ;

data:

 channel_ID = 7, 5, 6, 8 ;

 Laser_Pointing_Angle = 5 ;

 Background_Low = 0, 30000, 30000, 30000 ;

 Background_High = 500, 50000, 50000, 50000 ;

 Molecular_Calc = 0 ;

 Pressure_at_Lidar_Station = 1010 ;

 Temperature_at_Lidar_Station = 19.8 ;

 id_timescale = 1, 0, 0, 0 ;

 LR_Input = 1,_,_,_ ;

 DAQ_Range = 100,_,_,_ ;

 Laser_Pointing_Angle_of_Profiles =
  0, 0,
  0, 0,
  0, 0,
  0, 0,
  0, 0,
  _, 0,
  _, 0,
  _, 0,
  _, 0,
  _, 0 ;


 Raw_Data_Start_Time =
  0, 0,
  60, 30,
  120, 60,
  180, 90,
  240, 120,
  _, 150,
  _, 180,
  _, 210,
```

```
 _, 240,
 _, 270 ;

Raw_Data_Stop_Time =
 60, 30,
 120, 60,
 180, 90,
 240, 120,
 300, 150,
 _, 180,
 _, 210,
 _, 240,
 _, 270,
 _, 300 ;


Raw_Bck_Start_Time =
 0, 0,
 60, 30,
 120, 60,
 _, 90,
 _, 120,
 _, 150;


Raw_Bck_Stop_Time =
 60, 30,
 120, 60,
 180, 90,
 _, 120,
 _, 150,
 _, 180 ;


Laser_Shots =
 1500, 3000, 3000, 3000,
 1500, 3000, 3000, 3000,
 1500, 3000, 3000, 3000,
 1500, 3000, 3000, 3000,
 1500, 3000, 3000, 3000,
 1500, _, _, _,
 1500, _, _, _,
 1500, _, _, _,
 1500, _, _, _,
 1500, _, _, _ ;


Raw_Lidar_Data = ...
```

```
Background_Profile = ...
```

This example file contains the minimum collection of mandatory information that has to be found within the *Raw Lidar Data* input file. If it is really necessary, the user can decide to add to these mandatory parameters any number of additional parameters considered in the previous example.

Finally, suppose we want to make the following changes with respect to the previous example:

1. use a sounding file for molecular density calculation instead of "US Standar Atmosphere 1976"
2. supply a lidar ratio profile to use in elastic backscatter retrieval instead of a fixed value
3. provide a overlap function for overlap correction

In this case we have to generate the following NetCDF additional files:

- `rs_20090130cc00.nc`
  The name of *Sounding Data* file has to be computed as follows:
  `"rs_"+Measurement_ID`
  The structure of this file is summarized in table 2.
- `ov_20090130cc00.nc`
  The name of *Overlap* file has to be computed as follows:
  `"ov_"+Measurement_ID`
  The structure of this file is summarized in table 3.
- `lr_20090130cc00.nc`
  The name of *Lidar Ratio* file has to be computed as follows:
  `"lr_"+Measurement_ID`
  The structure of this file is summarized in table 4.

Moreover we need to apply the following changes to the *Raw Lidar Data* input file:

1. Change the value of the variable `Molecular_Calc` as follows:

   ```
   Molecular_Calc = 1 ;
   ```

   Of course the variables `Pressure_at_Lidar_Station` and `Temperature_at_Lidar_Station` are not necessary anymore.

2. Change the values of the array `LR_Input` as follows:

   ```
   LR_Input = 0,_,_,_ ;
   ```

3. Add the global attribute `Sounding_File_Name`

   ```
   Sounding_File_Name = "rs_20090130cc00.nc" ;
   ```

4. Add the global attribute `LR_File_Name`

   ```
   LR_File_Name = "lr_20090130cc00.nc" ;
   ```

5. Add the global attribute `Overlap_File_Name`

   ```
   Overlap_File_Name = "ov_20090130cc00.nc" ;
   ```

Table 1: NetCDF Raw Lidar Data file structure

Description	Type	
Dimensions		
points		
Number of vertical bins of lidar profiles. In case different channels correspond to different numbers of vertical bins this dimension has to be set to the maximum number of vertical bins	-	Mandatory
channels		
Number of lidar channels	-	Mandatory
nb_of_time_scales		
Number of different time scales included in lidar data. If all channels are acquired with the same time scale this dimension has to be set to 1.	-	Mandatory
time		
Number of profiles included in the time-series	UNLIMITED	Mandatory
time_bck		
Number of dark measurement profiles	-	Optional
scan_angles		
Number of scan angles used during the measurement	-	Mandatory
Variables		
channel_ID(channels)		
Channel ID in SCC relational database.	`int`	Mandatory
channel_string_ID(channels)		
Channel string ID in SCC relational database.	`string`	Optional
Laser_Repetition_Rate(channels)		
Laser repetition rate in Hz for each channel	`int`	Optional
Laser_Pointing_Angle(scan_angles)		
Laser pointing angle(s) with respect to the zenith expressed in degrees	`double`	Mandatory
Scattering_Mechanism(channels)		
Defines the scattering mechanism involved in each lidar channel. 0-Elastic, 1-Raman N_2, 2-Cross Polarization, 3-Parallel Polarization, 4-Raman H_2O, 5-RRl, 6-RRh	`int`	Optional
Signal_Type(channels)		
Defines the signal type involved in each lidar channel. All the possible values are given in the text	`int`	Optional
Emitted_Wavelength(channels)		
Emitted wavelengths in nm for each channel	`double`	Optional
Detected_Wavelength(channels)		
Detected wavelengths in nm. These are the center of your interferential filter for each channel.	`double`	Optional
Raw_Data_Range_Resolution(channels)		
Raw data range resolution of lidar profile in m for each channel	`double`	Optional

continued on next page

continued from previous page

Description	**Type**	
Background_Mode(channels)		
Defines the way to use for atmospherical background subtraction for each channel. 0-Pre-trigger, 1-Far field	`int`	Optional
Background_Low(channels)		
Minimum altitudes for atmospherical background calculation in meters for each channel. If pre-trigger is used as background subtraction mode for a particular channel, the corresponding value of this variable has to be set to the rangebin to be used as lower limit (within pre-trigger region) for background calculation.	`double`	Mandatory
Background_High(channels)		
Maximum altitude for atmospherical background calculation in meters for each channel. If pre-trigger is used as background subtraction mode for a particular channel, the corresponding value of this variable has to be set to the rangebin to be used as upper limit (within pre-trigger region) for background calculation. If the variable First_Signal_Rangebin is not given the first valid lidar rangebin will be the next after Background_High one.	`double`	Mandatory
Molecular_Calc		
Defines the way to calculate molecular numerical density. 0-US Standard Atmosphere 1976, 1-External radiosounding, 2-GFS Model. This last option is not yet implemented in the SCC.	`int`	Mandatory
id_timescale(channels)		
This array determines which time scale is used for the acquisition of each channel. In particular this array defines the link between the channel index and the time scale index. In case a single time scale is used, all values of this array have to be set to 0.	`int`	Mandatory
Dead_Time_Corr_Type(channels)		
This array defines the type of dead time correction that has to be applied of photoncounting lidar data. Please use a value of 0 for a not-paralyzable channel and 1 for a paralyzable one.	`int`	Optional
Dead_Time(channels)		
Value of dead time in ns for each channel	`double`	Optional
Acquisition_Mode(channels)		
Defines the acquisition mode used for each channel. 0-Analog, 1-Photoncounting	`int`	Optional
Trigger_Delay(channels)		
The delay in ns between the laser pulse output and the middle of the first rangebin for each channel.	`double`	Optional

continued on next page

continued from previous page

Description	Type	
Laser_Pointing_Angle_of_Profiles(time,nb_of_time_scales)		
The array determines which scan angle is used for the acquisition of each lidar profile. In particular this array defines the link between the time and timescale indexes and the scan angle index.	`int`	Mandatory
Raw_Data_Start_Time(time, nb_of_time_scales)		
Start time of each raw lidar profile expressed in seconds from the RawData_Start_Time_UT	`int`	Mandatory
Raw_Data_Stop_Time(time, nb_of_time_scales)		
Stop time of each raw lidar profile expressed in seconds from the RawData_Start_Time_UT	`int`	Mandatory
Laser_Shots(time, channels)		
Number of laser shots accumulated for each channel at any time	`int`	Mandatory
Raw_Lidar_Data(time, channels, points)		
Raw lidar data. For photoncounting channels the counts have to be used and for analog channels the signal in mV.	`double`	Mandatory
Pol_Calib_Range_Min(channels)		
Mimimum of altitude range to use for the linear polarization calibration	`double`	Mandatory for depolarization calibration product
Pol_Calib_Range_Max(channels)		
Maximum of altitude range to use for the linear polarization calibration	`double`	Mandatory for linear polarization calibration product
LR_Input(channels)		
Lidar ratio to be used within the elastic-only backscatter retrieval. Two options are available: 0 for lidar ratio profile (taken from an external file submitted by the user), 1 for fixed value (taken from SCC_DB)	`int`	Mandatory if elastic backscatter retrievals have to be done
DAQ_Range(channels)		
Analog scale used to acquire analog signals in mV	`double`	Mandatory if analog signals are present
Pressure_at_Lidar_Station		
Pressure measured at lidar station level in hPa.	`double`	Mandatory if Molecular_Calc=0
Temperature_at_Lidar_Station		
Temperature measured at lidar station level in C.	`double`	Mandatory if Molecular_Calc=0
Background_Profile(time_bck,channels,points)		
Dark measurements profiles. These profiles will be subtracted from the lidar profiles.	`double`	Optional
Raw_Bck_Start_Time(time_bck, nb_of_time_scales)		

continued on next page

continued from previous page

Description	Type	
Start time of each dark measurement profile expressed in seconds from the RawBck_Start_Time_UT	int	Mandatory if Background_Profile is given
Raw_Bck_Stop_Time(time_bck, nb_of_time_scales)		
Stop time of each dark measurement profile expressed in seconds from the RawBck_Start_Time_UT	int	Mandatory if Background_Profile is given
Error_On_Raw_Lidar_Data(time, channels, points)		
This array has to be used only by lidar systems able to provide the errors on each single raw analog lidar profile. This array has to be filled only in correspondence of analog channels leaving all other values as undefined (for the photoncounting channels the SCC will calculate the errors as the square root of the counts.	double	Optional
First_Signal_Rangebin(channels)		
Rangebin at which lidar profile begins starting from 0. If it is not given the first valid rangebin will be the one after the Background_High in case pre-trigger is used as background subtraction mode, 0 if far field is used.	int	Optional
cloud_mask_channel_idx		
Index of the channel the provided cloud mask refers to (in terms of channel position in the file starting with 0, i.e. channel dimension index)	int	Optional
cloud_mask(time, points)		
Manual cloud mask defined as bitmask (3 bits): bit0→unknown_cloud; bit1→water_cloud; bit2→cirrus. Cloud-free region should have all the bits unset. Valid range: 0-7. Undefined values is NC_FILL_BYTE (-127).	byte	Mandatory if cloud_mask_channel_idx has been defined
Global Attributes		
Measurement_ID		
Measurement identifier defining your measurement. The value of this global attribute has to match with the Measurement_ID given in SCC database for the same measurements.	text	Mandatory
RawData_Start_Date		
The start date of measurement in format YYYYMMDD. The value of this attribute has to match with the start date given in SCC database	text	Mandatory
RawData_Start_Time_UT		
Start Time of measurement (UT) in format HHMMSS	text	Mandatory
RawData_Stop_Time_UT		
Stop Time of measurement (UT) in format HHMMSS	text	Mandatory
RawBck_Start_Date		
The start date of the dark measurement in format YYYYMMDD	text	Mandatory if Background_Profile is given

continued on next page

continued from previous page

Description	**Type**	
RawBck_Start_Time_UT		
Start Time of dark measurement (UT) in format HHMMSS	`text`	Mandatory if Background_Profile is given
RawBck_Stop_Time_UT		
Stop Time of measurement (UT) in format HHMMSS	`text`	Mandatory if Background_Profile is given
Sounding_File_Name		
Name of NetCDF sounding file to use in molecular density calculation.	`text`	Mandatory if Molecular_Calc=1
LR_File_Name		
Name of NetCDF file containing the lidar ratio profile to use within elastic backscatter retrievals.	`text`	Mandatory if at least one value of LR_Input is zero
Overlap_File_Name		
Name of NetCDF overlap file	`text`	Optional
Location		
Location where the lidar system is running	`text`	Optional
System		
Lidar system name	`text`	Optional
Latitude_degrees_north		
Latitude where the lidar system is running	`double`	Optional
Longitude_degrees_east		
Longitude where the lidar system is running	`double`	Optional
Altitude_meter_asl		
Altitude above see level	`double`	Optional

Table 2: NetCDF Sounding Data file structure

Description	Type	
Dimensions		
points		
Number of vertical bins of sounding profiles.	-	Mandatory
Variables		
Altitude(points)		
Altitude above sounding station in m. The vertical resolution can be different from the resolution of lidar profile. In this case the molecular density will be interpolated at same resolution of lidar profile.	`double`	Mandatory
Temperature(points)		
Sounding temperature profile in °C	`double`	Mandatory
Pressure(points)		
Sounding pressure profile in hPa	`double`	Mandatory
RelativeHumidity(points)		
Sounding relative humidity profile (%)	`double`	Optional
Global Attributes		
Latitude_degrees_north		
Latitude of sounding station	`double`	Mandatory
Longitude_degrees_east		
Longitude of sounding station	`double`	Mandatory
Altitude_meter_asl		
Altitude above sea level of sounding station	`double`	Mandatory
Location		
Location name of sounding station	`text`	Optional
Sounding_Station_Name		
Sounding station name	`text`	Optional
WMO_Station_Number		
WMO station number	`text`	Optional
WBAN_Station_Number		
WBAN station number	`text`	Optional
Sounding_Start_Date		
Sounding start date in format YYYYMMDD	`text`	Mandatory
Sounding_Start_Time_UT		
Sounding start (synoptic) time UT in format HHMMSS	`text`	Mandatory
Sounding_Stop_Time_UT		
Sounding stop time UT in format HHMMSS	`text`	Optional

Table 3: NetCDF Overlap file structure

Description	Type	
Dimensions		
points		
Number of vertical bins of overlap function. In case different channels have different numbers of bins this dimension has to be set to the maximum number of vertical bins.	-	Mandatory
channels		
Number of lidar channels for which the overlap function has been measured	-	Mandatory
Variables		
Altitude(points)		
Hight above lidar station profile in m. The vertical resolution can be different from the resolution of lidar profile. In this case the overlap profile will be interpolated at same resolution of lidar profile	`double`	Mandatory
Overlap_Function(channels,points)		
Correction factor for overlap.	`double`	Mandatory
channel_ID(channels)		
Channel ID in SCC relational database for which overlap function has been measured. In the final version of SCC it will be provided to the user a tool to get the channel IDs for his lidar system via Web. At the moment these IDs will be comunicated directly to the user.	`int`	Mandatory
Global Attributes		
Lidar_Station_Name		
Earlinet call-sign for Lidar Station	`text`	Mandatory
Overlap_Measurement_Date		
The date in which the overlap function has been measured in format YYYYMMDD	`text`	Mandatory

Table 4: NetCDF Lidar Ratio file structure

Description	Type	
Dimensions		
points		
Number of vertical bins of lidar ratio profiles. In case different channels have different number of bins this dimension has to be set to the maximum number of vertical bins	-	Mandatory
products		
Number of lidar products for which the lidar ratio profile is given	-	Mandatory
Variables		
Altitude(points)		
Hight above lidar station profile in m. The vertical resolution can be different from the resolution of lidar profile. In this case the lidar ratio profile will be interpolated at same resolution of lidar profile	`double`	Mandatory
Lidar_Ratio(products,points)		
Lidar ratio profile	`double`	Mandatory
Lidar_Ratio_Error(products,points)		
Lidar ratio error profile	`double`	Optional
product_ID(products)		
Product ID in SCC relational database for which the lidar profile has been given. In the final version of SCC it will be provided to the user a tool to get the product IDs for his lidar system via Web. At the moment these IDs will be comunicated directly to the user.	`int`	Mandatory
Global Attributes		
Lidar_Station_Name		
Earlinet call-sign for Lidar Station	`text`	Mandatory