# SCC input file

***SCC Winter School***
***CNR-IMAA, Potenza, Italy***
***4-6 December 2018***

# SCC input file

- All the SCC files (input, intermediate and output) are in Network Common Data Form (NetCDF)
- The NetCDF is a well known self-describing, machine-independent data format that support the creation, access, and sharing of array-oriented scientific data (http://www.unidata.ucar.edu/software/netcdf/)
- The NetCDF is a binary format that allows the definition of multi-dimensional variables of several types (integers, double, character, etc).
- For each variable it is possible to define one or more attributes where to specify variable properties like units, long name, description, etc.
- It is possible to define global attributes which are not related to a specific variable but to the whole file.

# NetCDF format

Typically a NetCDF file is composed by four different sections

1. **dimensions**
   contains all the dimensions used to define all the variables included in the NetCDF file

2. **variables**
   each variable is defined as a multi-dimensional array of a specific type and with the dimensions defined in the dimensions section

3. **global attributes**
   lists all the attributes referring to the whole file. As the variable the attributes (global or the one attached to a specific variable) can be of different types

4. **data**
   section containing the data values of each variable defined in variables section. Attribute values (both global or related to a specific variable) are not reported in data section but directly in variable or global attribute sections

# NetCDF format

## Available data types

| Name | Size | NetCDF-3* | NetCDF-4 |
|---|---|---|---|
| NC_BYTE | 8-bit signed integer | yes | yes |
| NC_UBYTE | 8-bit unsigned integer | yes | yes |
| NC_CHAR | 8-bit character byte | yes | yes |
| NC_SHORT | 16-bit signed integer | yes | yes |
| NC_USHORT | 16-bit unsigned integer | no | yes |
| NC_INT | 32-bit signed integer | yes | yes |
| NC_UINT | 32-bit unsigned integer | no | yes |
| NC_INT64 | 64-bit signed integer | no | yes |
| NC_UINT64 | 64-bit unsigned integer | no | yes |
| NC_FLOAT | 32-bit floating point | yes | yes |
| NC_DOUBLE | 64-bit floating point | yes | yes |
| NC_STRING | variable length character string | no | yes |

*does not support data compression

For each data type it is defined a corresponding fill value which should be used to identify missing values.

# NetCDF format

## Standard libraries and tools

- NetCDF-C
- NetCDF-Fortran
- NetCDF-C++
- NetCDF-Java

The libraries include also (command-line!) several conversion tools.

The most common used are:

- **ncdump**

  Generete a CDL (network Common data form Description Language) text representation of a NetCDF dataset
- **ncgen**

  Generate a NetCDF file out of CDL format
- **nccopy**

  utility to copy and optionally compress and chunk NetCDF data

# SCC Raw Input Files

To perform aerosol optical retrievals the SCC needs the raw lidar data as well as a certain number of input parameters to use in both pre-processing and optical processing stages.

The SCC gets these parameters looking at two different locations:

- Single Calculus Chain database (DB)
- Input files

The parameters can be found:

- in the input files only (those ones changing from measurement to measurement)
- in the DB only
- in both input files and DB

  when looking for a particular parameter the SCC checks first in the input file; if the required parameter is found the corresponding value DB in not taken into account

# SCC Raw Input Files

The SCC can handle four different types of input files

1. **Raw Lidar Data**
   lidar data as well as other parameters to be use in the SCC processing

2. **Sounding Data**
   the sounding data (from a correlative radiosounding or model).
   It is used by the SCC to compute the molecular density calculation

3. **Overlap**
   the measured overlap function

4. **Lidar Ratio**
   lidar ratio profile to use in elastic backscatter retrievals.

The Raw Lidar Data file is of course mandatory while the other are optional.

Detailed documentation on http://scc-documentation.readthedocs.io

# SCC Raw Input Files

## Mandatory fields

### dimensions:

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

### variables:

```
int channel_ID(channels) ;
string channel_string_ID(channels);
double Laser_Pointing_Angle(scan_angles) ;
double Background_Low(channels) ;
double Background_High(channels) ;
int Molecular_Calc ;
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" ;
```

# SCC Raw Input Files

## Filling the variables depending on channel dimensions

*channel_ID = 7, 5, 6, 8 ;*
or
*channel_string_ID='cc_1064','cc_532c','cc_532p', 'cc_607';*

The values reported by the variable *channel_ID* should match with the corresponding channel ID defined in the SCC DB.

Let's suppose the following associations are defined in the SCC DB:

| | | | | |
|---|---|---|---|---|
| 1064 | → | channel ID=7 | or | channel String ID=cc_1064 |
| 532 cross | → | channel ID=5 | or | channel String ID=cc_532c |
| 532 parallel | → | channel ID=6 | or | channel String ID=cc_532p |
| 607 | → | channel ID=8 | or | channel String ID=cc_607 |

For consistency **ALL** the variables depending on the dimension channel should be filled using the same channel order used for the variable *channel_ID*.

For example:

Raw_Lidar_Data[time][0][points] → raw time series of 1064 channel
Raw_Lidar_Data[time][1][points] → raw time series of 532 cross channel
Raw_Lidar_Data[time][2][points] → raw time series of 532 parallel channel
Raw_Lidar_Data[time][3][points] → raw time series of 1064 channel

# SCC Input File : available tools

1. **atmospheric-lidar**
   author: Iannis Binietoglou
   language: Python
   link: https://bitbucket.org/iannis_b/atmospheric-lidar
   description: **licel2scc**, licel2scc-depol (convert Licel binary files to the EARLINET's Single Calculus Chain NetCDF format)

2. **SCC netcdf checker**
   author: Iannis Binietoglou
   language: Python
   link: https://bitbucket.org/iannis_b/scc-netcdf-checker
   description: The aim of this script is to check if a netcdf file has the correct format to be used with the EARLINET's Single Calculus Chain

3. **SCC access**
   author: Iannis Binietoglou
   languange: Python
   link: https://bitbucket.org/iannis_b/scc-access
   description: tool for interacting with the Single Calculus Chain through the command line. Specifically, with the script you can upload a file to the SCC for processing, download the processed files and graphs, delete an existing measurement from the SCC