Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Meteorologie und Klimatologie MeteoSchweiz

Pyrad

Data Processing Cookbook

Version: 0.4.0
Date: October, 18, 2017

Document History

Responsible People

Creation/Edition	J. Grazioli
Revision	
Approval	
Further information	Document editors

Versionenkontrolle

Version	**Edited by**	**Date**	**Activity**
0.2.0	J. Grazioli	2017-08-23	Creation begun
0.3.0	J. Grazioli	2017-10-18	check products section
0.4.0	J. Grazioli, J. Figueras i Ventura	2017-11-28	Review and structural modification
0.5.0	D. Wolfensberger	2021-01-12	Added section on GECSX

Contents

1 Introduction

1.1 Motivation for Separating Code and Configuration

- A future "final" version of the code should not be touched.
- Clear change and versioning control of the code.
- No local file paths and personal settings in the code.
- Developing and test can be done using local config files.
- Parameters are not hard coded and can be changed easily.
- High flexibility. All settings are made by changing only 1–3 config files.

1.2 Rationale and other sources of information

The processing is based on the PyRad framework. Usually pyrad is cloned in the home directories of each users in the appropriate servers. As an example, for zueub222 and user jgr:

```
jgr@zueub222:pyrad$ pwd
/home/lom/users/jgr/pyrad
```

The purpose of this document is to allow users to process data by manipulating only configuration files. The focus is therefore here on the processing output and not on PyRad itself. For an overview of the functionalities of PyRad, its installation, and development, please refer to the documentation available at:

```
pyrad/doc/
```

that includes the following main documents:

- *pyart-mch_library_reference_dev*:
- *pyart-mch_library_reference_users*:

2 Data Processing

2.1 Data Processing

The information about this section is complementary to Sec. 2 of the document:

```
pyrad/doc/pyrad_user_manual.docx
```

The data processing can be started from the linux shell, after activation of the proper conda environment. Depending on the server, the appropriate environment may be "root" (i.e. for zueub222):

```
jgr@zueub222:~$ source activate root
(root) jgr@zueub222:~$
```

or it can be "pyrad" (i.e. for CSCS and cirrus servers). The python scripts used to process the radar data can be called from the directory:

```
pyrad/src/pyrad_proc/scripts/
```

The scripts that are useful for this document are the following processing and realtime scripts:

- main_process_data.py
- main_process_data_rt.py
- main_process_data_period.py
- process_trajectory.py (obsolete, other functions should be used)

they can all be called from the linux shell.

2.1.1 Process data real time (main_process_data_rt.py)

This script is designed to process data in real time. It can operate in two different ways:

- The script is "listening" on some data folders and immediately process new data se they appear. The script therefore remains active all the time.
- The script is periodically restarted by cronjob.

Verbatim, from the help page of the script:

```
This program performs real time processing of the data

To run the processing framework type:
    python main_process_data.py [config_files]
     --starttime [process_start_time] --endtime [process_end_time]
     --cfgpath [cfgpath] --proc_period [proc_period]

If startime or endtime are specified the program will start processing at the
specified time and end at the specified time. Otherwise the program ends when
the user interrupts it.
cfgpath is an optional argument with default: '$HOME/pyrad/config/processing/'
proc_period is the time that has to pass before attempting to restart the
processing in s
if proc_finish is not none it indicates the time the program is allowed to ran
```

```
berfore forcing it to end


Example:
    python main_process_data.py 'paradiso_fvj_vol.txt' 'paradiso_fvj_rhi.txt'
    --starttime '20140523000000' --endtime '20140523001000'
    --cfgpath '$HOME/pyrad/config/processing/' --proc_period 60 --proc_finish 120


usage: main_process_data_rt.py [-h] [--starttime STARTTIME]
                               [--endtime ENDTIME] [--cfgpath CFGPATH]
                               [--proc_period PROC_PERIOD]
                               [--proc_finish PROC_FINISH]
                               cfgfiles [cfgfiles ...]

Entry to Pyrad processing framework

positional arguments:
  cfgfiles              name of main configuration file

optional arguments:
  -h, --help            show this help message and exit
  --starttime STARTTIME
                        starting time of the data to be processed. Format
                        YYYYMMDDhhmmss
  --endtime ENDTIME     end time of the data to be processed. Format
                        YYYYMMDDhhmmss
  --cfgpath CFGPATH     configuration file path
  --proc_period PROC_PERIOD
                        Period between processing rounds (s)
  --proc_finish PROC_FINISH
                        Processing time allowed before shutdown (s)
```

2.1.2 Process data (main_process_data.py)

Standard data processing (i.e., usually non real time) is performed by this script. Verbatim from the help page:

```
This program processes and post-processes data over a time span

To run the processing framework type:
    python main_process_data.py [config_file] --starttime [process_start_time] --endtime
     [process_end_time] --postproc_cfgfile [postproc_config_file] --cfgpath [cfgpath]

If startime and endtime are not specified the program determines them from
the trajectory file or the last processed volume.
postproc_cfgfile is an optional argument with default: None
cfgpath is an optional argument with default: '$HOME/pyrad/config/processing/'

Example:
    python main_process_data.py 'paradiso_fvj_vol.txt' --starttime '20140523000000'
```

```
    --endtime '20140523001000' --postproc_cfgfile 'paradiso_fvj_vol_postproc.txt' --cfgpath
     '$HOME/pyrad/config/processing/'


usage: main_process_data.py [-h] [--starttime STARTTIME] [--endtime ENDTIME]
                            [--postproc_cfgfile POSTPROC_CFGFILE]
                            [--cfgpath CFGPATH] [-i INFOSTR] [-t TRAJFILE]
                            proc_cfgfile

Entry to Pyrad processing framework

positional arguments:
  proc_cfgfile          name of main configuration file

optional arguments:
  -h, --help            show this help message and exit
  --starttime STARTTIME
                        starting time of the data to be processed. Format
                        YYYYMMDDhhmmss
  --endtime ENDTIME     end time of the data to be processed. Format
                        YYYYMMDDhhmmss
  --postproc_cfgfile POSTPROC_CFGFILE
                        name of main post-processing configuration file
  --cfgpath CFGPATH     configuration file path
  -i INFOSTR, --infostr INFOSTR
                        Information string about the actual data processing
                        (e.g. 'RUN57'). This string is added to the filenames
                        of the product files.
  -t TRAJFILE, --trajfile TRAJFILE
                        Definition file of plane trajectory. Configuration of
                        scan sector, products, ...
```

2.1.3 Process data (main_process_data_period.py)

This script is used in post-processing to process data over long periods of time, usually several days. It can, for example:

- Process several individual days.
- Process several days portions (e.g. several days, all from 08 to 10)

According to its help entry:

```
This program does the daily processing and post-processing over a period of time.

To run the processing framework type:
    python main_process_data.py [config_file] [process_start_date] [process_end_date]
    --starttime [process_start_time] --endtime [process_end_time] --postproc_cfgfile
     [postproc_config_file] --cfgpath [cfgpath]

starttime is an optional argument with default: '000000'
endtime is an optional argument with default: '235959'
```

```
postproc_cfgfile is an optional argument with default: None
cfgpath is an optional argument with default: '$HOME/pyrad/config/processing/'

Example:
    python main_process_data.py 'paradiso_fvj_vol.txt' '20140523' '20140525'
    --starttime '000000' --endtime '001000' --postproc_cfgfile 'mals_emm_vol_postproc.txt'
    --cfgpath '$HOME/pyrad/config/processing/'


usage: main_process_data_period.py [-h] [--starttime STARTTIME]
                                   [--endtime ENDTIME]
                                   [--postproc_cfgfile POSTPROC_CFGFILE]
                                   [--cfgpath CFGPATH]
                                   proc_cfgfile startdate enddate

Entry to Pyrad processing framework

positional arguments:
  proc_cfgfile          name of main configuration file
  startdate             starting date of the data to be processed. Format
                        YYYYMMDD
  enddate               end date of the data to be processed. Format YYYYMMDD

optional arguments:
  -h, --help            show this help message and exit
  --starttime STARTTIME
                        starting date of the data to be processed. Format
                        hhmmss
  --endtime ENDTIME     end date of the data to be processed. Format hhmmss
  --postproc_cfgfile POSTPROC_CFGFILE
                        name of main post-processing configuration file
  --cfgpath CFGPATH     configuration file path
```

This script creates the product once all the data has been processed.

2.1.4 Process trajectories (process_trajectory.py)

This script is used in post-processing to process trajectory data. The usage of this script is:

```
usage: process_trajectory.py [-h] [-c CFGFILE]
                             [--preproc_cfgfile PREPROC_CFGFILE] [-i INFOSTR]
                             trajfile [starttime] [endtime]

Create PYRAD products using a plane trajectory

positional arguments:
  trajfile              Definition file of plane trajectory. Configuration of
                        scan sector, products, ...
  starttime             Starting time of the data to be processed. Format:
                        YYYYMMDDhhmm[ss]. If not given, the time of the first
                        sample is used.
  endtime               End time of the data to be processed. Format:
```

```
                        YYYYMMDDhhmm[ss]. If not given, the time of the last
                        sample is used.

optional arguments:
  -h, --help            show this help message and exit
  -c CFGFILE, --cfgfile CFGFILE
                        Main configuration file. Defines the
  --preproc_cfgfile PREPROC_CFGFILE
                        name of main pre-processing configuration file
  -i INFOSTR, --infostr INFOSTR
                        Information string about the actual data processing
                        (e.g. 'RUN57'). This string is added to the filenames
                        of the product files.

Example:
  process_trajectory.py -c $HOME/pyrad/config/processing/mals_emm_rw22_traj.txt
     --preproc_cfgfile $HOME/pyrad/config/processing/mals_emm_rw22_traj_preproc.txt
     -i TS011 /data/mals_plane_traj/EMM/gnv_20161026_ts011_seat_emmen_flt01_ADS.txt
```

3 Configuration

The configuration of the data processing is divided into three files. The *main configuration file* (see Section 3.1), the *location configuration file* (see Section 3.2) describing the location of the weather radar and the used scans. The *product configuration file* describes the datasets and products. As this is bit more complicated it is described in its own Section 4.

The configuration files are located in *malsgit/config_pyrad/processing/*.

3.1 Main Configuration File

Tha main configuration file is used to define the global settings, notably the paths to the different sources of data. The parameters of the main configuration file are described in Table 2

Table 2: Configuration parameters of the main configuration file

Name	Type	Description
name	STRING	Name of the data processing. This name is used in the path of the saved products in the following manner: <saveimgbasepath>/<name>/<YYYY-MM-DD>/<datasetname>/<prodname>/ <outputname>
datapath	STRING	Base directory of the rainbow raw data. This field must have a trailing '/'. The raw data files of a scan can be found using the following file path: <datapath>/<scanname>/<YYYY-MM-DD>/<YYYYMMDDHHMMSS00datatype>.<ext>
configpath	STRING	Base directory of the configuration files. This directory contains clutter maps, filter coefficients, antenna pattern, and the data processing configuration files.
cosmopath	STRING	Base directory of the COSMO data files.
dempath	STRING	Base directory of the Digital Elevation Model (DEM) files. Basically to load the radar visibility (Optional)
smnpath	STRING	Base directory of the SwissMetNet stations data. Used in the comparison between radar data and rain gauges (Optional)
disdropath	STRING	Base directory of the disdrometer data. Used in the comparison between radar data and disdrometers (Optional)
solarfluxpath	STRING	Base directory of the solar flux data. Used to plot the calibration bias based on sun monitoring (Optional)
locationConfigFile	STRING	File name (with full path) of the location configuration file. Described in Section 3.2.
productConfigFile	STRING	File name (with full path) of the product configuration file. Described in Section 4.
lastStateFile	STRING	File name (with full path) of the file containing the time of the last processed scan. Used in particular for real time processing.
imgformat	STRING/STRARR	File format(s) of the images. The following formats are supported: eps, png and jpg. If *saveimg* is set to 0, this field is not used.

Continued on next page

Table 2 – *Continued from previous page*

Name	Type	Description
saveimgbasepath	STRING	Base directory for the images to save. The directory structure looks as follows: <saveimgbasepath>/<name>/<YYYY-MM-DD>/<datasetname>/<prodname>/ <outputname> If *saveimg* is set to 0, this field is not used.
loadbasepath	STRING	OPTIONAL. Base path of saved data. By default, this field is set to *saveimgbasepath*.
loadname	STRING	OPTIONAL. Name of the saved data processing. Used for saved volume loading. By default, this field is set to *name*.

3.2 Location Configuration File

The location configuration files describes some parameters that are depending on the specific location of a radar (type of scans we want to measure, radar name, etc). The location of the weather radar (its position) itself, is instead usually read from the radar metadata directly and it is not necessarily defined in this file. The fields are described in Table 3.

Table 3: Configuration parameters of the location configuration file

Name	Type	Description
RadarName	STRING	Short version name of a C-band radar (i.e. A, D, L, P) or DX50, MXPol for the X-band radars.
RadarRes	STRING	rad4alp radar resolution (H or L). Only necessary if rad4alp data is processed
RadarBeamwidth	FLOAT	Radar antenna beam width [Deg]
AntennaGaindB	FLOAT	antenna gain [dB]
ScanList	STRARR	A list with the scans used for this data processing. Note that the first scan in this list is used as master scan. The master scan must be the first (temporal) scan of the corresponding rainbow task. In case of composite volumes the master scan is usually a PPI and the following are RHIs. If the radar processed is MCH C-band the scan list consists of the radar elevation (i.e. from 001 to 020). All scan names must have a trailing '/' except if rad4alp data is processed
ScanPeriod	FLOAT	Repetition period of each scan in minutes.
Azimtol	FLOAT	Tolerance in azimuth for irregular data. (0.5 is a good value)
clutterMap	STRING	Clutter map of the data processing. The clutter map is located at <configpath>/clutter/<clutterMap>
AntennaGain	FLOAT	Radar antenna gain. Not used for X-band MCH data.
radarconsth(v)	FLOAT	Radar constant h (v). Not mandatory.
mflossh(v)	Matched filter losses h (v). Not mandatory.	

Continued on next page

Table 3 – *Continued from previous page*

Name	Type	Description
attg	FLOAT	Gas attenuation coeafficient (**units?** (1 way attenuation)
CosmoRunFreq	INT	Frequency of a COSMO model run in hours.
CosmoForecasted	INT	Hours forecasted by the COSMO model.
rmax	FLOAT	For C-band data, the maximum range in [m] to be considered. USeful for speed considerations.
elmax	FLOAT	Maximum elevation [°] to consider
ppiImageConfig	STRUCT	Structure defining the PPI image generating. The following 6 fields are described below:
rhiImageConfig	STRUCT	Structure defining the RHI image generating. The following 6 fields are described below:
xsize	INT	Number of horizontal pixels of the picture (without frame).
ysize	INT	Number of vertical pixels of the picture (without frame).
xmin	FLOAT	Distance of the left image boundary to the radar in km.
xmax	FLOAT	Distance of the right image boundary to the radar in km.
ymin	FLOAT	Distance of the lower image boundary to the radar in km.
ymax	FLOAT	Distance of the upper image boundary to the radar in km.
ppiMapImageConfig	STRUCT	Structure defining the PPI image overlayed on a map. The following 9 fields are described below:
rngRing	FLOAT	Distance between range rings (0 means no range ring) [km].
xsize	FLOAT	Image size (inches) [ich].
ysize	FLOAT	Image size [ich].
lonmin	FLOAT	Minimum WGS84 longitude [°].
lonmax	FLOAT	Maximum WGS84 longitude [°].
latmin	FLOAT	Minimum WGS84 latitude [°].
latmax	FLOAT	Maximum WGS84 latitude [°].
mapres	STRING	Map resolution. Accepted strings are: “10m”, “50m”, “110m”
maps	STRARR	String array of possible maps to overplot. Accepted entries include: relief, countries, provinces, urban_areas, roads, railroads, coastline, lakes, lakes_europe, rivers, rivers_europe
rvsazImageConfig	STRUCT	Structure defining the range versus azimuth image. The following 4 fields are described below:
xmin	FLOAT	Min angle on horizontal axis [Deg].
xmax	FLOAT	Max angle on horizontal axis [Deg].
ymin	FLOAT	Min range on vertical axis [km].
ymax	FLOAT	Max range on vertical axis [km].
rvselImageConfig	STRUCT	Structure defining the range versus elevation image. It contains the same 4 fields as rvsazImageConfig.
sunhitsImageConfig	STRUCT	Structure defining the sun hits image. The following 6 fields are described below:

Continued on next page

Table 3 – *Continued from previous page*

Name	Type	Description
xsize	INT	Number of horizontal pixels of the picture (without frame).
ysize	INT	Number of vertical pixels of the picture (without frame).
xmin	FLOAT	Minimum azimuth angle difference (between sun and radar).
xmax	FLOAT	Maximum azimuth angle difference (between sun and radar).
ymin	FLOAT	Minimum elevation angle difference (between sun and radar).
ymax	FLOAT	Maximum azimuth angle difference (between sun and radar).
azPatternFile	STRING	Name of the azimuth pattern file of the antenna. This file and path must be <config-path>/antenna/<azPatternFile>
elPatternFile	STRING	Name of the elevation pattern file of the antenna. This file and path must be <config-path>/antenna/<elPatternFile>
fixed_angle	FLOAT	Fixed angle of a PAR antenna in degrees. For the PAR azimuth antenna this is the elevation angle. For the elevation antenna is is the azimuth angle.

4 Product Generation Configuration

This section describes the product configuration.

4.1 Basic Concept

The concept is based on three stages: 1. input or raw data volume, 2. datasets and 3. products.

The center point of the data processing is the dataset. A dataset can be generated from one or more than one input data volumes (e.g. a rainrate dataset uses 5 different raw data volume to be generated).

There are several different formats of a dataset. For example, a dataset can be a volume, a trajectory, a volume composite or a time series. Section 5 summarizes the possible datasets.

From a dataset the products are generated. What products can be generated from a dataset depends on the type (volume, volume composite, trajectory or time series) of the dataset.

Figure 1 shows a schema of an example of the basic concept of the product generation. From different input data a dataset is generated and from this dataset m products are created.

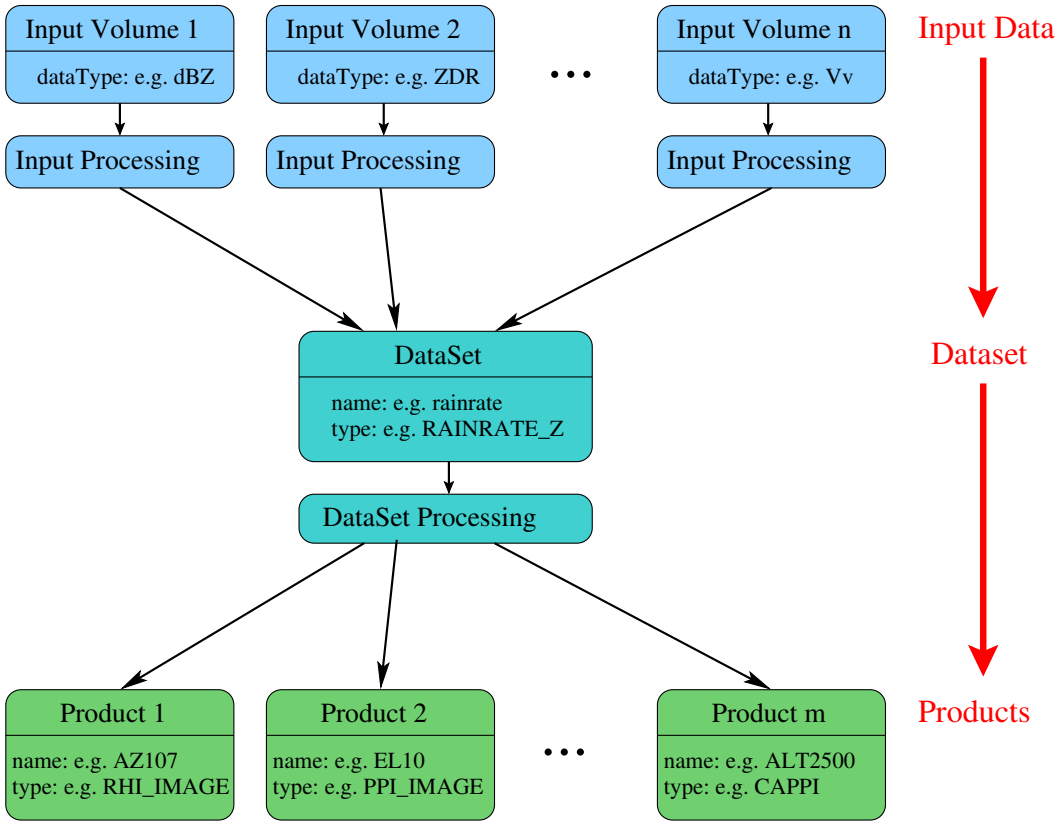

Figure 1: Basic concept for product generation from input (also called raw) data.

The dataset types are described in Section 5 and the product types in Section 6

4.2 Input Volume Datatypes

There are different group of input volume datatypes that can be read as input data for a dataset. These groups are summarized in Table 4. The possible volume datatypes for each datatype group are listed in Table 5.

The specification of the datatype for a dataset is done by first writing the datatype group, followed by a ':' and then the name of the datatype. If no ':' is given, it is assumed that the

datatype belongs to the group *RAW*. For a datatype from the group *SAVED*, the dataset and product name must be specified. This information is separated by ','. Note that *NETCDF* datatypes cannot be mixed with other datatypes.

Caveat: please do not mistake the input volume datatype with the dataset type (entry "type" of the dataset structure)

```
<datasetname> STRUCT 2
   type     STRING <datasettype>
   datatype STRARR
      RAW:dBZ
      PSR:NhDBM
      COSMO:TEMP
      SAVED:RR_h,<olddataset>,<oldproduct>
```

Group	Description
RAW	Raw data generated by the DX50 (or other rainbow format). Stored in rainbow data format.
CFRADIAL	A dataset generated by this dataprocessing procedure. As additional information the dataset and the product name of the previous dataset must be specified.
COSMO	Data created by COSMO. Converted to polar radar coordinates and stored in rainbow file format.
DEM	Digital Elevation Model data (DEM). Basically visibility in rainbow file format
NETCDF	**NOT THERE YET!**
RAD4ALP	rad4alp data.
RAD4ALPDEM	rad4alp visibility data.
RAD4ALPCOSMO	a binary file with COSMO data for rad4alp processing.
PROC	indicates that the dataset is the result of the preprocessing of raw data. (i.e. it will be created on the fly).

Table 4: List of input volume datatype groups

Group	Datatypes
RAW	dBZ, dBZv, dBuZ, dBuZv, V, Vv, Vu, Vvu, W, Wv, Wu, Wvu, KDP, uKDP, uKDPu, PhiDP, uPhiDP, uPhiDPu, RhoHV, RhoHVu, uRhoHV, L, ZDR, ZDRu, SQI, SQIv, SQIu, SQIvu, SNRh, SNRv, CDR
IQ	Same as PSR plus WhADU, WvADU, WhDBADU, WvDBADU, WhDBM, WvDBM, WhDBZ, WvDBZ, IhCPX, IvCPX, IhRAW, IvRAW, IhADU, IvADU, IhDBADU, IvDBADU, IhDBM, IvDBM, IhDBZ, IvDBZ, IhDEG, IvDEG
CFRADIAL	(Some examples, not complete) dV, dVv, dVu, dVvu, RR_Zh, RR_Ah, RR_Kdp, Att, SAN, TRAJ, HEIGHT, WP, WPDIFF, WPRELDIFF, dtfilter, RAINEXT, RCS
COSMO	ISO0, TEMP, H_ISO0
DEM	VIS
NETCDF	**NICE TO HAVE, NOT THERE YET**
RAD4ALP	dBZ, ZDR, RhoHV, uPhiDP, V, W, SNRh, SNRv, L, CDR.
RAD4ALPDEM	VIS.
RAD4ALPCOSMO	ISO0, TEMP
PROC	dBZc, dBZvc, ZDRc, PhiDPc, KDPc, RhoHVc, Ah, Adp.

Table 5: List of possible datatypes

4.3 Product Configuration File

The product configuration files describes the products that are generated for the data processing. The fields are described in Table 6.

Table 6: Configuration parameters of the product configuration file

Name	Type	Description
dataSetList	STRARR	A list of the datasets that are generated for the data processing. There must be a structure in the product configuration file defining the dataset and its product for each dataset in this list. The list of datasets may include the processing level. **TODO: add link to the definition of processing level**
<datasetname>	STRUCT	A structure defining a dataset. The <datasetname> must be a member of the *dataSetList* list. The structure defines the type of the dataset, its parameters and the products that are applied to this dataset. The <datasetname> can be freely chosen. Just make sure the spelling is the same in the dataSetList. The <datasetname> is used for the path to store the products: <saveimgbasepath>/<name>/<YYYY-MM-DD>/<datasetname>/<prodname>/ <outputname>. The mandatory fields of this structure are described below. The fields depending on the dataset type are described in Section 5.
type	STRING	Type of the dataset. The tables in Section 5 list all possible dataset types.

Continued on next page

Table 6 – *Continued from previous page*

Name	Type	Description
datatype	STRARR	Raw (or input) datatype. The dataset is generated using rainbow raw files of this datatypes or the dataset is generated using multiple raw files with different datatypes.
IGNORE_-MISSING_VOLS	INT	OPTIONAL. If set, the function processing the dataset is called if not all input data volumes could be selected. For example this could be used for the sanity check. In such a case only the checks are done with the available input volumes. By default, this option is off. **TODO: can be removed?**
DSSAVENAME	STRING	OPTIONAL. Usually the product files are stored under the name of the dataset. If this parameter is saved, the files are stored under this name instead of the dataset name.
INPUT-PROCESSING	STRUCT	OPTIONAL. Input processing of one or more input volumes. See section **??** for details. **TODO: can be removed?**
DATASET-PROCESSING	STRUCT	OPTIONAL. Dataset processing of a volume dataset. See section **??** for details. **TODO: can be removed?**
products	STRUCT	This structure contains a list of products. Each product is a structure named as <prodname>. For the product description see Section 6.
<prodname>	STRUCT	Structure defining a product of the dataset. The <prodname> can be freely chosen. The <prodname> is used to store the output products: <saveimgbasepath>/<name>/<YYYY-MM-DD>/<datasetname>/<prodname>/ <outputname>. The mandatory fields of this structure are described below. The fields depending on the product type are described in Section 6.
type	STRING	Type of the product. Section 6 describes all possible product types.

Example: A simple product config file is listed below:

```
#
# Product generation configuration
#

# List of datasets to generate.
# The detailed specification of each dataset is given below.
dataSetList STRARR 12
   l0:TEMP
   l0:reflectivity
   l0:ZDR
   l0:RhoHV
   l0:echoID
   l1:echoFilter
   l3:echoFilter_Ah
   l2:outlierFilter
   l2:Att_ZPhi
```

```
   l3:hydroclass
   l4:rainrate
   l3:wind


# ============================================================================================
#               COSMO data
# ============================================================================================
TEMP STRUCT 6
    type STRING COSMO_LOOKUP
    datatype STRARR 1
        dBZ
    cosmo_type STRING TEMP
    regular_grid INT 0
    lookup_table INT 1
    MAKE_GLOBAL INT 1


# ============================================================================================
#                 raw data processing
# ============================================================================================
reflectivity STRUCT 3
   type     STRING RAW
   datatype STRING dBZ
   products STRUCT 4
      EL03_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 0
         voltype STRING dBZ
      EL04_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 1
         voltype STRING dBZ
      EL05_7 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 2
         voltype STRING dBZ
      SAVESTATE STRUCT 2
         type STRING SAVESTATE
         voltype STRING dBZ

ZDR STRUCT 3
   type     STRING RAW
   datatype STRING ZDR
   products STRUCT 3
      EL03_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 0
         voltype STRING ZDR
      EL04_0 STRUCT 3
         type  STRING PPI_IMAGE
```

```
         anglenr INT 1
         voltype STRING ZDR
      EL05_7 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 2
         voltype STRING ZDR

RhoHV STRUCT 3
   type     STRING RAW
   datatype STRING RhoHV
   products STRUCT 3
      EL03_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 0
         voltype STRING RhoHV
      EL04_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 1
         voltype STRING RhoHV
      EL05_7 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 2
         voltype STRING RhoHV


# ============================================================================================
#                 echo identification
# ============================================================================================
echoID STRUCT 3
    type STRING SAN
    datatype STRARR 4
        dBZ
        ZDR
        uPhiDP
        RhoHV
    MAKE_GLOBAL INT 1


# ============================================================================================
#                 clutter and noise suppression
# ============================================================================================
# echo type 3 : precip, 2 : clutter, 1 : noise
echoFilter STRUCT 4
    type STRING ECHO_FILTER
    datatype STRARR 8
        PROC:echoID
        dBZ
        ZDR
        RhoHV
        PhiDP
        KDP
```

```
V
W
    echo_type INT 3
    MAKE_GLOBAL INT 1

echoFilter_Ah STRUCT 4
    type STRING ECHO_FILTER
    datatype STRARR 2
        PROC:echoID
        PROC:Ah
    echo_type INT 3
    MAKE_GLOBAL INT 1


# ================================================================================================
#                 outlier filter
# ================================================================================================
outlierFilter STRUCT 8
    type STRING OUTLIER_FILTER
    datatype STRARR 1
        PROC:Vc
    threshold FLOAT 10.
    nb INT 2
    nb_min INT 3
    percentile_min FLOAT 5.
    percentile_max float 95.
    MAKE_GLOBAL INT 1


# ================================================================================================
#                 Attenuation
# ================================================================================================
Att_ZPhi STRUCT 5
    type STRING ATTENUATION
    datatype STRARR 4
        PROC:dBZc
        PROC:ZDRc
        PROC:PhiDPc
        PROC:TEMP
    ATT_METHOD STRING ZPhi
    fzl FLOAT 2000.
    MAKE_GLOBAL INT 1


# ================================================================================================
#                 hydrometeor classification products
# ================================================================================================
hydroclass STRUCT 5
    type STRING HYDROCLASS
    datatype STRARR 5
        PROC:dBZc
```

```
        PROC:ZDRc
        PROC:RhoHVc
        PROC:KDPc
        PROC:TEMP
    HYDRO_METHOD STRING SEMISUPERVISED
    RADARCENTROIDS STRING DX50
    MAKE_GLOBAL INT 1


# ================================================================================================
#               rainfall rate
# ================================================================================================
rainrate STRUCT 5
    type STRING RAINRATE
    datatype STRARR 3
        PROC:dBZc
        PROC:Ahc
        PROC:hydro
    RR_METHOD STRING hydro
    MAKE_GLOBAL INT 1
    products STRUCT 3
      EL03_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 0
         voltype STRING RR
      EL04_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 1
         voltype STRING RR
      EL05_7 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 2
         voltype STRING RR


# ================================================================================================
#               wind velocity
# ================================================================================================
wind STRUCT 5
    type STRING WIND_VEL
    datatype STRARR 1
        PROC:Vc
    vert_proj INT 0
    MAKE_GLOBAL INT 1
    products STRUCT 3
      EL03_0 STRUCT 3
         type  STRING PPI_IMAGE
         anglenr INT 0
         voltype STRING wind_vel_h_az
      EL04_0 STRUCT 3
         type  STRING PPI_IMAGE
```

```
   anglenr INT 1
   voltype STRING wind_vel_h_az
EL05_7 STRUCT 3
   type  STRING PPI_IMAGE
   anglenr INT 2
   voltype STRING wind_vel_h_az
```

The example product configuration files defines twelve datasets: let us take the example of *l0:reflectivity* and *l4: rainrate*.

The *reflectivity* dataset is a *RAW* dataset generated using the raw datatype *dBZ*. Four products are generated for this dataset: the products *EL03_ 0*, *EL04_ 0*, *EL05_ 7* which are of type *PPI_ IMAGE* (the first, second and third PPIs in a volume scan, as given in field *anglenr*). The images are stored in <saveimgbasepath>/<name>/<YYYY-MM-DD>/reflectivity/EL0X_ X/. The fourth product saves the volume in the path given by *loadbasepath* fo the main configuration file.

The *rainrate* dataset is a *RAINRATE* dataset generated using the processed datatypes *dBZc*, *Ahc*, *hydro*, by means of the *HYDRO* retrieval method. In analogous way with respect to the *reflectivity* dataset, PPI images are generated as products.

4.4 The concept of processing level

Processing level(s) are defined in the product configuration file, in the initial definition of the dataset list (i.e. $l0$, $l1$, $l2\ldots$). The processing level defines the order in which the datasets will be processed. This is particularly useful when subsequent processing levels need data from previous datasets. In this case, the order matters, and the option "MAKE_GLOBAL" should be used.

5 Datasets

A thorough description of the available datasets can be found in Sec. 2 of **pyrad_library_reference_user**. The dataset type is defined by the "type" entry in the dataset block of the product configuration file. Dataset types can be found in Table 7.

5.1 Dataset type and format

The field "type", for each dataset listed in the product configuration, takes the values as in Table 7. Depending on the type, the proper pyrad/pyart processing routine is applied. Each dataset type has a dataset "format" (more types may correspond to the same format), that will be used to generate the appropriate products.

Table 7: List of dataset types with basic identification.

Type	(Format) Explanation	Reference
RAW	(VOL) Process raw data	
GRID	(GRID) Grid data	
QVP	(QVP) Quasi-Vertical-Profile	
TIME_HEIGHT	(QVP) Time-height time series	
CDF	(VOL) Cumulative Density Function	
NCVOL	(VOL) Volume in NetCDF format	
PWR	(VOL) Signal power	
SNR	(VOL) Signal-to-noise ratio	
RHOHV_CORRECTION	(VOL) Noise correction ρ_{HV}	
BIAS_CORRECTION	(VOL) Bias correction	
L	(VOL)	
CDR	(VOL)	
SAN	(VOL) Echo identification/sanity	
CLT_TO_SAN	(VOL) Clutter to echo classification	
ECHO_FILTER	(VOL) Filter on echo classification	
SNR_FILTER	(VOL) Filter on SNR	
VIS_FILTER	(VOL) Filter on visibility	
OUTLIER_FILTER	(VOL) Filter outliers	
PHIDP0_CORRECTION	(VOL) Correct on starting Φ_{dp}	
PHIDP_SMOOTH_1w	(VOL) Single window wmoothing of Φ_{dp}	
PHIDP_SMOOTH_2w	(VOL) Double window wmoothing of Φ_{dp}	
PHIDP_KDP_VULPIANI	(VOL) Φ_{dp} and K_{dp} estimation by Vulpiani et al.	
PHIDP_KDP_KALMAN	(VOL) Φ_{dp} and K_{dp} estimation by Schneebeli et al.	
PHIDP_KDP_MAESAKA	(VOL) Φ_{dp} and K_{dp} estimation by Maesaka et al.	
PHIDP_KDP_LP	(VOL) Φ_{dp} and K_{dp} estimation by ??	
KDP_LEASTSQUARE_1W	(VOL) K_{dp} estimation, single window least square	
KDP_LEASTSQUARE_2W	(VOL) K_{dp} estimation, double window least square	
ATTENUATION	(VOL) Radar attenuation	
RAINRATE	(VOL) Rainrate estimation	

Continued on next page

Table 7 – *Continued from previous page*

Name	Type	Reference
WIND_VEL	(VOL) Wind velocity (radial) estimation	
WINDSHEAR	(VOL) Wind shear (spectral width)	
HYDROCLASS	(VOL) Hydrometeor identification	
ML_DETECTION	(VOL) Melting layer detection	
PHIDP0_ESTIMATE	(VOL) Estimation of Φ_{dp0}	
RHOHV_RAIN	(VOL) ρ_{hv} in rain	
ZDR_PREC	(VOL) Z_{DR} in precipitation	
ZDR_SNOW	(VOL) Z_{DR} in snow	
SELFCONSISTENCY_KDP_PHIDP	(VOL) Self consistency	
SELFCONSISTENCY_BIAS	(VOL) Bias from consistency	
COSMO	(VOL) COSMO data	
COSMO_LOOKUP	(VOL) Cosmo lookup table	
COSMO_COORD	(COSMO_COORD) Cosmo coordinates	
HZT_LOOKUP	(VOL) HZT lookup table	
HZT	(VOL)	
TIME_AVG	(TIMEAVG) Time averaging	
FLAG_TIME_AVG	(TIMEAVG) Time average flag	
COLOCATED_GATES	(COLOCATED_GATES) Process colocated gates of radars	
INTERCOMP	(INTERCOMP) Radars intercomparison	
INTERCOMP_TIME_AVG	(INTERCOMP) Time average intercomparison	
MONITORING	(MONITORING) Data monitoring	
GC_MONITORING	(MONITORING) Data monitoring ??	
OCCURRENCE	(OCCURRENCE)	
OCCURRENCE_PERIOD	(OCCURRENCE)	
SUN_HITS	(SUN_HITS) Sun hits in radar data	
POINT_MEASUREMENT	(TIMESERIES)	
TIMESERIES	(TIMESERIES) Time series	
TRAJ	(TRAJ_ONLY) Trajectory	
TRAJ_ATPLANE	(TIMESERIES) Trajectory at plane/object location	
TRAJ_ANTENNA_PATTERN	(TIMESERIES) Trajectory at plane/object location, given antenna pattern	
TRAJ_LIGHTNING	(TIMESERIES) Trajectory at lighning locations	

5.2 Datasets and configuration parameters

TODO TODO

6 Products

For each dataset format several products can be generated, as seen in the product generation file. The list of products for each dataset are passed as a string array, and each product is defined as a structure. Different product "type" can be selected, determining the subsequent process. Not all product types are available for each data type. The following tables and sections list the possible products for each dataset format.

6.1 VOL products

Listed here are the products for VOL dataset formats.

Name	Description	Reference
PPI_IMAGE	PPI image of constant elevation	Sec: 6.1.1
PPI_MAP	PPI image on a map	Sec: 6.1.2
PSEUDOPPI_IMAGE	Reconstructed PPI from non-PPI data	Sec: 6.1.3
PSEUDOPPI_MAP	PSEUDO-PPI on a map	Sec: 6.1.4
RHI_IMAGE	RHI image of constant azimuth	Sec: 6.1.4
RHI_PROFILE	Averaged height profile	Sec: 6.1.6
PSEUDORHI_IMAGE		Sec: 6.1.5
CAPPI_IMAGE	Constant altitude PPI image	Sec: 6.1.7
PLOT_ALONG_COORD	Plot along coordinates	Sec: 6.1.8
BSCOPE_IMAGE	B-scope type image (ray versus range)	Sec: 6.1.9
TIME_RANGE	Time vs range display	Sec: 6.1.10
HISTOGRAM	Histogram of the variable	Sec: 6.1.11
QUANTILES	Quantiles of some variables	Sec: 6.1.12
FIELD_COVERAGE		Sec: 6.1.13
CDF	Cumulative density function	Sec: 6.1.14
SAVEVOL	Save the generated dataset volume	Sec: 6.1.15
SAVEALL	**TODO: describe**	Sec: 6.1.17
SAVESTATE	Save the time of the processed volume.	Sec: 6.1.16

Table 8: List of all possible product types for dataset VOL format

6.1.1 PPI_IMAGE

PPI image of constant elevation. The parameters of a *PPI_IMAGE* product are listed in Table 11.

Parameter	Type	Description
TYPE	STRING	Must be *PPI_IMAGE*.
anglenr	INT	Elevation number (starting with 0) of the PPI image to generate.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)

Table 9: Parameters of a PPI_IMAGE product.

The dimension and boundaries of the PPI image are set in the location configuration file (Sec. 3.2).

6.1.2 PPI_MAP

PPI image of constant elevation on a map The parameters of a *PPI_MAP* product are the same as listed in Table 11. The dimension and boundaries of the PPI map are set in the location configuration file (Sec. 3.2).

6.1.3 PSEUDOPPI_IMAGE

Pseudo-PPI is a PPI reconstruction from volumic data (they can be also a series of RHIs), not from genuine PPIs. It is displayed as a PPI. The parameters of a PSEUDOPPI_IMAGE can be found in Table 10.

Parameter	Type	Description
TYPE	STRING	Must be *PSEUDOPPI_IMAGE*.
angle	FLOAT	Elevation angle, in degrees of the pseudo-PPI
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
EleTol	FLOAT	Elevation tolerance on the given angle, in degrees.

Table 10: Parameters of a PSEUDOPPI_IMAGE product.

6.1.4 PSEUDOPPI_MAP

As for Sec. 6.1.3, but displayed on a map.

RHI_IMAGE

RHI image of constant azimuth. The parameters of a *RHI_IMAGE* product are listed in Table **??**.

Parameter	Type	Description
TYPE	STRING	Must be *RHI_IMAGE*.
anglenr	INT	Azimuth number (starting with 0) of the RHI image to generate.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)

Table 11: Parameters of a RHI_IMAGE product.

The dimension and boundaries of the RHI image are set in the location configuration file (Sec. 3.2).

6.1.5 PSEUDORHI_IMAGE

Pseudo-RHI is a PPI reconstruction from volumic data, not from genuine RHIs. It is displayed as a PPI. The parameters of a PSEUDORHI_IMAGE can be found in Table 12.

Parameter	Type	Description
TYPE	STRING	Must be *PSEUDORHI_IMAGE*.
angle	FLOAT	Azimuth angle, in degrees of the pseudo-RHI
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
EleTol	FLOAT	Azimuth tolerance on the given angle, in degrees.

Table 12: Parameters of a PSEUDORHI_IMAGE product.

6.1.6 RHI_PROFILE

Averaged height profile. The dataset is limited by the *rangeStart* and *rangeStop* parameters. For each radar gate the vertical height is calculated, the heights are divided into height layers. For each height layer the median, the 15% and 75% quantiles are calculated.

The product is a figure data value vs. altitude with median and the two quantile curves. Also a text file with the height profile is generated. The height resolution in the text file is according the parameter *heightResolution*. In the image file the height resolution is always 100 m. The height is given in meters above sea level.

The parameters of a *RHI_PROFILE* product are listed in Table 13.

Parameter	Type	Description
TYPE	STRING	Must be *RHI_PROFILE*.
anglenr	INT	Azimuth number (starting with 0) of the RHI image to generate.
rangeStart	FLOAT	Minimal distance to the radar for the gates to be used for the RHI_PROFILE processing. Given in meters.
rangeStop	FLOAT	Maximal distance to the radar for the gates to be used for the RHI_PROFILE processing. Given in meters.
heightResolution	FLOAT	Resolution of the height layers for the processing. Is only applied to the text output file. The profile image height resolution is fixed to 100 m.
heightMax	FLOAT	Maximum height to consider, in [m].
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)

Table 13: Parameters of a RHI_PROFILE product.

6.1.7 CAPPI_IMAGE

The parameters of a *CAPPI_IMAGE* product are listed in Table 14.

Parameter	Type	Description
TYPE	STRING	Must be *CAPPI_IMAGE*.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
altitude	FLOAT	Altitude of the CAPPI. In meters above see level.

Table 14: Parameters of a CAPPI product.

6.1.8 PLOT_ALONG_COORD

Plotting along a given coordinate. The parameters of a PLOT_ALONG_COORD product are listed in Table 15.

Parameter	Type	Description
TYPE	STRING	Must be *PLOT_ALONG_COORD*.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
mode	STRING	*ALONG_RNG*, or *ALONG_AZI*, or *ALONG_ELE*. It sets the radar dimension along which to plot.
fix_azimuths	FLTARR	All fixed azuth angles (same length of a given second dimension)
fix_elevations	FLTARR	All fixed elevation angles.
fix_ranges	FLTARR	All fixed ranges in meters.
AngTol	FLOAT	Tolerance on the angles, in degrees.
value_start	FLOAT	Starting value of the extracted profile. Units depend on selected "mode"
value_end	FLOAT	Ending value of the extracted profile. Units depend on selected "mode"

Table 15: Parameters of a PLOT_ALONG_COORD product.

If for, example we are interested in extracting data along range (i.e. a radial), we will set the mode as *ALONG_RNG*, and we will provide the given couples of fixed elevations and azimuths (we will not provide fixed ranges).

6.1.9 BSCOPE_IMAGE

The parameters of a *BSCOPE_IMAGE* product are listed in Table 16. A B-scope is the typical 2D radar display given by ray vs range.

Parameter	Type	Description
TYPE	STRING	Must be *BSCOPE_IMAGE*.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
anglenr	INT	Azimuth number (starting with 0) or elevation number (starting with 0) to generate.

Table 16: Parameters of a radar BSCOPE product.

6.1.10 TIME_RANGE

Time vs range radar image from volume data. The parameters are given in Table 17. Somehow similar to a BSCOPE.

Parameter	Type	Description
TYPE	STRING	Must be *TIME_RANGE*.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
anglenr	INT	Azimuth number (starting with 0) or elevation number (starting with 0) to generate.

Table 17: Parameters of a radar TIME_RANGE product.

6.1.11 HISTOGRAM

Histogram of VOL data. The HISTOGRAM product accept the following parameters:

Parameter	Type	Description
TYPE	STRING	Must be *TIME_RANGE*.
voltype	STRING	A valid datatype string (i.e., KDP, KDPc, TYPE, dBZ...)
anglenr	INT	Azimuth number (starting with 0) or elevation number (starting with 0) to generate.

Table 18: Parameters of a radar HISTOGRAM product.

6.1.12 QUANTILES

Generate quantiles of volume data, for a given variable type. The parameters of a QUANTILE product can be found in Table 19

Parameter	Type	Description
TYPE	STRING	Must be *QUANTILES*.
voltype	STRING	Type of volume (i.e., KDP, KDPc, TYPE, dBZ...)
quantiles	FLTARR	List of quantiles to compute. Example: 50 (50%, median).

Table 19: Parameters of a radar QUANTILES product.

6.1.13 FIELD_COVERAGE

Field coverage (for eaxample rain extension) of a given volume type. For its configuration, please see Table 19

Parameter	Type	Description
TYPE	STRING	Must be *FIELD_COVERAGE*.
voltype	STRING	Type of volume (i.e., KDP, KDPc, TYPE, dBZ...)
threshold	FLOAT	To be given. Threshold on the chosen variable to discriminate its field.
nvalid_min	FLOAT	Default: 5. Minimum number of valid points.
ele_res	FLOAT	Default: 1. Elevation resolution to be used, in degrees.
azi_res	FLOAT	Default: 2. Azimuth resolution to be used, in degrees.
ele_min	FLOAT	Default: 0. Minimum elevation angle to use, in degrees.
ele_max	FLOAT	Default: 30.Maximum elevation angle to use, in degrees.
ele_step	FLOAT	Default: 5. Elevation step to use, in degrees.
ele_sec_start	FLOAT	degrees
ele_sec_stop	FLOAT	degrees
quantiles	FLTARR	Quantiles to be considered.

Table 20: Parameters of a radar FIELD_COVERAGE product.

6.1.14 CDF

Cumulative density function, whose parameters are listed in Table 21.

Parameter	Type	Description
TYPE	STRING	Must be *CDF*.
voltype	STRING	Type of volume (i.e., KDP, KDPc, TYPE, dBZ...)
sector	STRUCT	If givem a structure used to limit the domain. It includes the fields: *rmin, rmax, azmin,azmax, elmin, elmax, hmin, hmax*. Also only some of those can be given.
vismin	FLOAT	
absolute	INT	Boolean.
quantiles	FLTARR	List of quantiles to be used
use_nans	INT	Boolean. Default False
nan_value	FLOAT	Default is 0. Value for NaN data
filterclt	INT	Boolean. Default is 0 (false). Filter clutter or not.
filterprec	INTARR	???

Table 21: Parameters of a radar CDF product.

6.1.15 SAVEVOL

Save the generated dataset volume in NetCDF cfradial. The *TRAJ_SAVED* datasets read the saved files and combine them with trajectory data. The parameters of a *SAVEVOL* product are listed in Table 22.

Parameter	Type	Description
TYPE	STRING	Must be *SAVEVOL*.

Table 22: Parameters of a SAVEVOL product.

6.1.16 SAVESTATE

Save the time of the processed volume. This is used for the realtime processing. The time is written to the file defined by the config parameter *lastStateFile*.

Use this product only for one dataset and only for the realtime processing. Because the time is written the same file independently of the dataset. Thus, the previous time can be easily overwritten if not paying attenuation.

The parameters of a *SAVESTATE* product are listed in Table 23.

Parameter	Type	Description
TYPE	STRING	Must be *SAVESTATE*.

Table 23: Parameters of a SAVESTATE product.

6.1.17 SAVEALL

SAVEALL is similar to SAVEVOL.

6.2 Timeseries products

Listed here are the products for TIMESERIES dataset format.

Name	Description	Reference
PLOT_AND_WRITE_POINT	Plots a time series with the evolution of a variable at a particular point. Writes the same information in a file.	
PLOT_CUMULATIVE_POINT	Plots a time series with the accumulation in time of a variable at a particular point.	
COMPARE_POINT	Time series plot showing the evolution of a radar variable and a variable from another sensor placed at a particular point.	
COMPARE_CUMULATIVE_POINT	Time series plot showing the accumulated value of a radar variable and a variable from another sensor placed at a particular point.	
COMPARE_TIME_AVG	**TODO**	**TODO**
PLOT_AND_WRITE	Make a plot of a timeseries. And write the timeseries to a file.	
PLOT_HIST	Plot a histogram of the timeseries.	

Table 24: List of all possible product types for dataset with TIMESERIES format

6.3 Grid products

Listed here are the products for grid data type.

Name	Description	Reference
SURFACE_IMAGE		
LATITUDE_SLICE		
LONGITUDE_SLICE		
CROSS_SECTION		
SAVEVOL	Save the volume product	

Table 25: List of all possible product types for dataset with GRID format

6.4 TIMEAVG products

TIMEAVG dataset products are the same as VOL products.

6.5 SUN_HITS products

SUN_HITS dataset format products are listed here:

Name	Description	Reference
WRITE_SUN_HITS	Write sun hits data	
PLOT_SUN_HITS	Plot sun hits data	
WRITE_SUN_RETRIEVAL	Write sun retrieval data	
PLOT_SUN_RETRIEVAL	Plot sun retrieval data	
PLOT_SUN_RETRIEVAL_TS	Plot sun retrieval time series	

Table 26: List of all the possible product types for datasets of SUN_HITS format

6.6 MONITORING products

MONITORING dataset products are listed here:

Name	Description	Reference
VOL_HISTOGRAM	Compute volumic data histogram.	
PPI_HISTOGRAM	Compute PPI data histogram.	
ANGULAR_DENSITY	Density plot along angles	
VOL_TS		
SAVEVOL	Save the volume data in NetCDF format.	

Table 27: List of all the possible product types for datasets of MONITORING format

6.7 OCCURRENCE products

OCCURRENCE dataset products are listed here:

Name	Description	Reference
WRITE_EXCESS_GATES		

Table 28: List of all the possible product types for datasets of OCCURRENCE format

6.8 INTERCOMP products

INTERCOMP dataset products are listed here:

Name	Description	Reference
WRITE_INTERCOMP	Write intercomparison between radars	
WRITE_INTERCOMP_TIME_AVG	Write time-averaged intercomparison	
PLOT_AND_WRITE_INTERCOMP_TS	Plot and write the timeseries of intercomparison	

Table 29: List of all the possible product types of INTERCOMP format

6.9 TRAJ_ONLY products

TRAJ_ONLY (trajectory only, without radar data interpolated) products are listed here:

Name	Description	Reference
TRAJ_PLOT	Plot of a given trajectory, in radar coordinates	
TRAJ_TEXT	Text file of a given trajectory	

Table 30: List of all the possible product types of TRAJ_ONLY format

6.10 COLOCATED_GATES

COLOCATED_GATES dataset products are listed here:

Name	Description	Reference
WRITE_COLOCATED_GATES	Write colocated gates products.	

Table 31: List of all the possible product types of COLOCATED_GATES format

6.11 COSMO_COORD products

COSMO_COORD dataset products are listed here:

Name	Description	Reference
SAVEVOL	Save data in NetCDF format	

Table 32: List of all the possible product types of COSMO_COORD format

6.12 QVP products

QVP (quasi-vertical-profiles) products are a part of VOL products, although generated over several volumes. The datset format is although QVP instead of VOL.

7 GECSX interface

Pyrad provides a wrapper to the GECSX algorithm. GECSX is a radar visibility estimation method originally designed by (**?**). GECSX allows to estimate the radar visibility as well as the ground clutter radar signal with a digital elevation model. GECSX can be run using the *main_process_gecsx* script.

This is an extraction of the script docstring.

```
    ================================================
Pyrad: The MeteoSwiss Radar Processing framework
================================================
To run the processing framework type:
    python main_process_data.py \
[config_file] --starttime [process_start_time] --endtime [process_end_time] \
--cfgpath [cfgpath] --gatherplots

cfgpath is an optional argument with default: \
'$HOME/pyrad/config/processing/'

if gatherplots is set to 1, all generated figures will be copied into a
new directory called "ALL_FIGURES" located in the output folder. This is
convenient since GECSX can generate many figures and they are placed by Pyrad
in separate folders.

There are two ways to use this program:
    1. By providing it with a set of valid radar scans, in this case the
    "datapath" entry in the main config file must be defined and a valid radar
    scan must be located in '<datapath>/<scanname>/<YYYY-MM-DD>/
    <YYYYMMDDHHMMSS00datatype>.<ext>', <scanname> can be any name, and datatype
    must correspond to the datatype entry for the GECSX dataset in the prod
    config file. You also need to provide starttime and endtime that include
    the timestamp of the radar scan
    2. Without radar data, by providing the following entries in the prod
    file for the GECSX dataset (you can choose any value, these are examples)
     rmax         FLOAT 50000.    # [m] maximum range
     azmin        FLOAT 0.        # [deg] minimum azimuth angle
     azmax        FLOAT 360.      # [deg] maximum azimuth angle
     anglestep    FLOAT 1.       # [deg] azimuth angle step
     range_resolution FLOAT 50. # [deg] range resolution
     antenna_elevations FLTARR 2 # deg
            0.7
            3.0
    as well as the following entries in the loc file (again choose any value)
    RadarPosition STRUCT 3
        latitude FLOAT 46.842473
        longitude FLOAT 6.918370
        altitude FLOAT 449.5

See the two examples pay_main_DX50.txt and pay_main_norad.txt in
$HOME/pyrad/config/gecsx/

Example:
```

```
    python main_process_gecsx.py pay_main_norad.txt
--cfgpath $HOME/pyrad/config/gecsx/ --gatherplots 1

    python main_process_gecsx.py pay_main_DX50.txt --starttime \
'20160101000000' --endtime '20170101001000' --cfgpath $HOME/pyrad/config/gecsx/
--gatherplots 1
```

There are two ways to use GECSX:

- With real radar data, by providing the path to proper radar data in the configuration files. Note that at the moment only one timestep and one radar will be used by the GECSX routine (the first one). If you want to process several radars I would suggest to create a new configuration file for each radar. To see an example of GECSX processing with real radar data check the files *pay_main_DX50.txt*, *pay_loc_DX50.txt* and *pay_prod_DX50.txt* in the folder *$HOME/pyrad/config/gecsx/*.

  An example of use in this case would be :

  ```
  python $HOME/pyrad/src/pyrad_proc/scripts/main_process_gecsx pay_main_DX50.txt
  --starttime 20160101000000 --endtime 20170101001000
  --cfgpath $HOME/pyrad/config/gecsx/ --gatherplots 1
  ```

- Without radar data by entering the properties of a virtual radar in the loc and prod files. To see an example of GECSX processing with a virtual radar check the files *pay_main_norad.txt*, *pay_loc_norad.txt* and *pay_prod_norad.txt* in the folder *$HOME/pyrad/config/gecsx/*.

  An example of use in this case would be :

  ```
  python $HOME/pyrad/src/pyrad_proc/scripts/main_process_gecsx pay_main_norad.txt
  --cfgpath $HOME/pyrad/config/gecsx/ --gatherplots 1
  ```

  Note the –gatherplots option, which copies all generate figures into a new directory located in the root directory of the GECSX outputs. This can be helpful since GECSX produces many products, which are all stored in separate folders by Pyrad.

The following Pyrad configuration parameters are particularly relevant for GECSX. Optional parameters are shown in italic.

7.1 GECSX parameters in the main configuration file

Table 33: GECSX parameters of the main configuration file

Name	Type	Description
dempath	STRING	Path where the DEM data is stored
datapath	STRING	Path where the radar data is stored, if not provided or empty, a virtual radar object will be created using the parameters given in the loc and prod files.

7.2 In the local configuration file

Table 34: GECSX parameters of the local configuration file

Name	Type	Description
ScanList	STRING	Subfolder within <datapath> where the radar files are stored, if not provided or empty, a virtual radar object will be created using the parameters given in the loc and prod files.
frequency	FLOAT	Radar transmit frequency in GHz
radar_beam_width_h	FLOAT	Radar half power beam width in degrees
pulse_width	FLOAT	Radar pulsewidth in seconds
txpwrh	FLOAT	Transmit power in W
AntennaGainH	FLOAT	Antenna gain along beam axis in dB
mflossh	FLOAT	matching filter losses in dB
lrxh	FLOAT	receiver losses from antenna feed to reference point in dB
RadarPosition	STRUCT	Structure containing the radar position in WGS84, its 3 keys are defined below. If not provided will use the position defined in the radar scan (as long as radar data is effectively provided).
latitude	FLOAT	Radar latitude in degrees
longitude	FLOAT	Radar longitude in degrees
altitude	FLOAT	Radar altitude in meters
attg	FLOAT	Gas attenuation coefficient in dB/km 1-way attenuation), if not provided will default to 0.012 *AzimTol*
FLOAT	Azimuth convolution window for moving antenna, if not provided, no convolution will be performed	
mosotti_factor	FLOAT	Clausius-Mosotti factor K, will default to 0.9644, as for water under standard temperature.
refcorr	FLOAT	Effect of atmospheric refraction to the earth bending radius, will default to 4/3.

7.3 In the product configuration file

Name	Type	Description
type	STRING	Must be GECSX, so that Pyrad understands that this is a GECSX product

datatype	STRING	Is used to find radar data. Indeed, the radar data must be stored under the name <datapath>/<ScanList>/<%Y-%m-%d/%Y%m%d%H%M%s00<datatype>.extension, where extension must be readable by Pyrad. If no file can be found a virtual radar object will be created using the parameters given in the loc and prod files.
demfile	STRING	Name of the DEM file which must be stored in the <dempath> folder. Currently geotiff, IDRISI files and ASCII DEMs are supported.
demproj	STRING	Either a EPSG code, a OGC WKT or Proj4 string, which define the projection of the DEM. It is required if projection is not available in demfile metadata. Check the webpage https://spatialreference.org/ref/epsg/ for a list.
az_discretization	FLOAT	Numerical step in azimuth in Cartesian visibility estimation in degrees. The lower the better but it will increase the computation time. If not provided a step of 0.2° will be used.
range_discretization	FLOAT	numerical step in range in meters in the Cartesian visibility estimation. If not provided a step of 100 m will be used.
do_range_weighting	INT	if set to 1, weight the backscattering cross sections of each cell according to their range offset using the radar pulse length. If not provided will be set to 1.
raster_oversampling	INT	The raster resolution of the DEM should be smaller than the range resolution of the radar (defined by the pulse length). If this is not the case, this keyword can be set to increase the raster resolution. If set to 0, no oversampling is done, if set to 1, oversampling is done, the factor N is automatically calculated such that 2*dx/N < pulse length, if set to 2 or larger, oversampling is done with this value as N. If not provided, will be set to 1 (automatic oversampling).
verbose	INT	if set to 1, info about the current processing state will be displayed, will be activated by default.
clip	INT	if set to 1, the input DEM will be clipped to the domain covered by the radar. This decreases a lot the computation speed and is generally recommended. If not provided, will be activated by default.
sigma0_method	STRING	Which method to use to compute $sigma_0$, i.e. the factor that relates backscattering area to backscattering cross-section of the ground clutter. Can be either "Gabella" (**?**) or "Delrieu" (**?**). If not provided, "Gabella" will be used as default.
rmax	FLOAT	maximum range of the virtual radar. Is required if no radar data is provided.
azmin	FLOAT	minimum azimuth angle of the virtual radar. Is required if no radar data is provided.
azmax	FLOAT	maximum azimuth angle of the virtual radar. Is required if no radar data is provided.

anglestep	FLOAT 2.	azimuth angle step of the virtual radar. Is required if no radar data is provided.
range_resolution	FLOAT	range resolution of the virtual radar. Is required if no radar data is provided.
antenna_elevations	FLTARR	list of elevations operated by the virtual radar. The radar visibility and clutter signal will be computed separately for all elevations. It is required if no radar data is provided. If radar data is provided and this parameter is defined in the config file, GECSX will only compute the elevation angles defined in this parameters, as long as they are available in the radar file (this can be used to process only a subset of the elevation angles operated by a real radar).

7.4 GECSX outputs

The GECSX produces the following outputs, which can all be used to generate Pyrad products.

Name	**Coords**	**Description**
terrain_altitude	CART	Altitude of the terrain, as obtained from the DEM [m]
bent_terrain_altitude	CART	Altitude of the terrain, corrected for atmospheric refraction [m]
terrain_slope	CART	Terrain slope [deg]
terrain_aspect	CART	Terrain orientation (to the north) [deg]
elevation_angle	CART	Radar elevation angle at topography level [deg]
visibility	CART	Visibility of every Cartesian pixel in the DEM map. 0 if not visible, 1 if visible.
min_vis_elevation	CART	Minimum visible elevation angle [deg]
min_vis_altitude	CART	Minimum visible altitude [m]
min_vis_altitude	CART	Minimum visible altitude [m]
sigma_0	CART	Ratio between clutter rcs and effective backscattering area [-]
incident_angle_field	CART	Angle of incidence of the radar beam with the topography [deg]
effective_area	CART	Effective backscattering area [m^2]
rcs_clutter	POL	Backscattering cross-section of the ground clutter [m^2], at every considered elevation angle
dBm_clutter	POL	Power signal of the ground clutter [dBm], at every considered elevation angle
dBm_clutter	POL	Reflectivity of the ground clutter [dBZ], at every considered elevation angle
visibility_polar	POLAR	Visibility at every polar gate [%], at every considered elevation angle

CART indicates Cartesian fields which have the dimensions of the (possibly clipped) DEM, POL indicates polar fields which have the dimensions of the either real or virtual radar provided as input to GECSX. The typical Pyrad graph product type for CART outputs is SURFACE$_IMAGE, and for POL outputs, PPI_IMAGE. The product type SAVEALL_VOL can be used to save all P$