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

Contents

1 Introduction **1**
- 1.1 Motivation for Separating Code and Configuration . . . 1
- 1.2 Rationale and other sources of information . . . 1

2 Data Processing **2**
- 2.1 Data Processing . . . 2
  - 2.1.1 Process data real time (main_process_data_rt.py) . . . 2
  - 2.1.2 Process data (main_process_data.py) . . . 3
  - 2.1.3 Process data (main_process_data_period.py) . . . 4
  - 2.1.4 Process trajectories (process_trajectory.py) . . . 5

3 Configuration **7**
- 3.1 Main Configuration File . . . 7
- 3.2 Location Configuration File . . . 8

4 Product Generation Configuration **11**
- 4.1 Basic Concept . . . 11
- 4.2 Input Volume Datatypes . . . 11
- 4.3 Product Configuration File . . . 13
- 4.4 The concept of processing level . . . 19

5 Datasets **20**
- 5.1 Product Configuration File . . . 20

6 Products **22**
- 6.1 VOL Products . . . 28

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 rad4alp 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 rad4alp 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>
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.
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:

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	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:
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).

Continued on next page

Table 3 – *Continued from previous page*

Name	Type	Description
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.

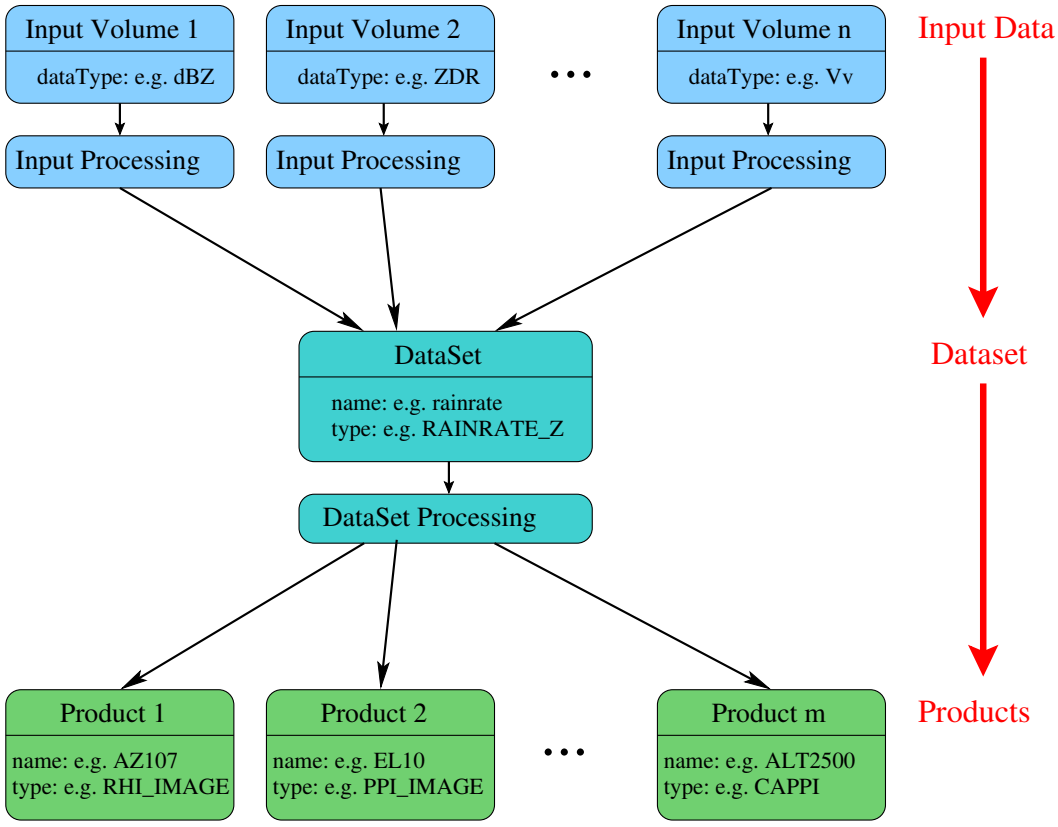

Figure 1: Basic concept for product generation from input (also called raw) data.

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 7.

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 **??**.

5.1 Product Configuration File

The product configuration files describes the products that are generated for the data processing. The fields are described in Table 7.

Table 7: List of dataset types with basic identification.

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

Continued on next page

Table 7 – *Continued from previous page*

Name	**Type**	**Reference**
PHIDP0_ESTIMATE	Estimation of Φ_{dp0}	
RHOHV_RAIN	ρ_{hv} in rain	
ZDR_PREC	Z_{DR} in precipitation	
ZDR_SNOW	Z_{DR} in snow	
SELFCONSISTENCY_KDP_PHIDP	Self consistency	
SELFCONSISTENCY_BIAS	Bias from consistency	
COSMO	COSMO data	
COSMO_LOOKUP	Cosmo lookup table	
COSMO_COORD	Cosmo coordinates	
HZT_LOOKUP	HZT lookup table	
TIME_AVG	Time averaging	
FLAG_TIME_AVG	Time average flag	
COLOCATED_GATES	Process colocated gates of radars	
INTERCOMP	Radars intercomparison	
INTERCOMP_TIME_AVG	Time average intercomparison	
MONITORING	Data monitoring	
GC_MONITORING	Data monitoring ??	
OCCURRENCE		
SUN_HITS	Sun hits in radar data	
TIMESERIES	Time series	
TRAJ	Trajectory	
TRAJ_ATPLANE	Trajectory at plane/object location	
TRAJ_ANTENNA_PATTERN	Trajectory at plane/object location, given antenna pattern	
TRAJ_LIGHTNING	Trajectory at lighning locations	

6 Products

For each dataset format several products can be generated. The following tables list the possible products for each dataset format.

Name	Description	Reference
PPI_IMAGE	PPI image of constant elevation	Section ??
PPI_MAP	PPI image on a map	**TODO: describe it**
PSEUDOPPI_IMAGE	**TODO: describe**	**TODO: describe it**
PSEUDOPPI_MAP	**TODO: describe**	**TODO: describe it**
RHI_IMAGE	RHI image of constant azimuth	Section ??
RHI_PROFILE	Averaged height profile	Section ??
PSEUDORHI_IMAGE	**TODO: describe**	**TODO: describe it**
CAPPI_IMAGE	Constant altitude PPI image	Section ??
PLOT_ALONG_COORD	**TODO: describe**	**TODO: describe it**
BSCOPE_IMAGE	**TODO: describe**	**TODO: describe it**
TIME_RANGE	**TODO: describe**	**TODO: describe it**
HISTOGRAM	**TODO: describe**	**TODO: describe it**
QUANTILES	**TODO: describe**	**TODO: describe it**
FIELD_COVERAGE	**TODO: describe**	**TODO: describe it**
CDF	**TODO: describe**	**TODO: describe it**
SAVEVOL	Save the generated dataset volume	Section ??
SAVEALL	**TODO: describe**	**TODO: describe it**
SAVESTATE	Save the time of the processed volume.	Section ??
	TODO: check -jgr- and -fvj- all the products below, what to do.	
WGS84_IMAGE	Image in WGS84 coordinates	Section ??
CH1903_IMAGE	Image in Swiss coordinates	Section ??
CAPPI_ASCII	XXX to be described	Section ??
PLOT_LINES	Plot values along a coordinate, holding the other two fixed.	Section ??
CDF_STAT	Cumulative distribution function	Section ??
NETCDF_CONV	Save data in netcdf file format	Section ??
MELTLAYER_IMAGE	Azimuth-Height graphic indicating the areas suspected to belong to the melting layer	Section ??
MELTLAYER_TS	Time series plot with the evolution of the melting layer.	Section ??
SAVE_DEM	Save a PPI in DEM format.	Section ??
RNGVSANG_IMAGE	Range versus angle image at a particular elevation or azimuth.	Section ??
SAVESLICE_PPI_ASCII	XXX to be described	Section ??
CONST_RANGE_IMAGE	Make a azimuth elevation plot at fixed range.	Section ??
CONTOUR_RANGE_IMAGE	make a contour plot (azimuth vs. elevation) at a fixed range gate	Section ??
CONTOUR_RANGE_IMAGE_3D	make a contour plot (azimuth vs. elevation) at a fixed range gate	Section ??
WRITE_BIN	Write bin values to a file.	Section ??

Name	Description	Reference
WRITE_SUN_HITS	**TODO: describe**	**TODO**
PLOT_SUN_HITS	**TODO: does it replace psunhits below?**	**TODO**
WRITE_SUN_RETRIEVAL	**TODO: describe**	**TODO**
PLOT_SUN_RETRIEVAL	**TODO: describe**	**TODO**
PLOT_SUN_RETRIEVAL_TS	**TODO: describe**	**TODO**
	TODO: check what to do with the IDL products below	
PSUNHITS_IMAGE	Creates a 2D plot where the x axis is the difference between azimuth position of the radar and the azimuth position of the sun, the y axis is the difference between elevation position of the radar and elevation position of the sun and color coded us the estimated sun hits power	Section **??**
PSUNRETRIEVAL_IMAGE	As above but with the retrieved sun power	Section **??**
ZDRSUNHITS_IMAGE	As above but with the Zdr of the sun hits	Section **??**
ZDRSUNRETRIEVAL_IMAGE	as above but with the retrieved Zdr of the sun	Section **??**
SUNRETRIEVAL_TS	Plot a time series showing the evolution of a sun retrieval parameter	Section **??**

Table 9: List of specific product types for dataset with SUN_HITS format. In addition to them, all the product types for dataset with VOL format can be applied.

Name	Description	Reference
	TODO: outdated? or replaced?	
POLAR_AZ_EL_IMAGE	A polar azimuth elevation plot	Section **??**
DISTANCE_VS_AZIMUTH_IMAGE	Make a 2D "data" vs azimuth plot	Section **??**
QUANTILE_STAT	Make "data" vs azimuth statistics	Section **??**

Table 10: List of all possible product types for dataset with RAY 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.	Section ??
PLOT_CUMULATIVE_POINT	Plots a time series with the accumulation in time of a variable at a particular point.	Section ??
COMPARE_POINT	Time series plot showing the evolution of a radar variable and a variable from another sensor placed at a particular point.	Section ??
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.	Section ??
COMPARE_TIME_AVG	**TODO**	**TODO**
PLOT_AND_WRITE	Make a plot of a timeseries. And write the timeseries to a file.	Section ??

Table 11: List of all possible product types for dataset with TIMESERIES format

Name	Description	Reference
	TODO: is it still of interest? is it replaced by QVP?	
TIME_ARRAY_IMAGE	Plot time-array plot (e.g. a time-height array).	Section ??
TIME_ARRAY_FILE	XXX to be described	Section ??

Table 12: List of all possible product types for dataset with TIMEARRAY format

Name	Description	Reference
	TODO: is it still of interest?	
TIME_ARRAY_IMAGE	Plot time-array plot (e.g. a time-height array).	Section ??
TIME_ARRAY_MAP	Plot time-array on a map	Section ??

Table 13: List of all possible product types for dataset with TIMEARRAY_ARRAY format

Name	Description	Reference
	TODO: now TRAJ and TRAJ_ONLY are merged in Pyrad, right?	
TRAJ_PLOT	Plot the range, elevation, azimuth of a plane trajectory.	Section ??
TRAJ_TEXT	Write the range, elevation, azimuth of a plane trajectory to a text file.	Section ??
TRAJ_ANTENNA	Plot the radar antenna movement.	Section ??
TRAJ_DX50_ANTENNA_HITS	Compare the DX50 antenna movement with a plane trajectory. Possible hits are listed.	Section ??

Table 14: List of all possible product types for dataset with TRAJ_ONLY format

Name	Description	Reference
	TODO: of interest?	
RHI_TRAJ	Overplot the trajectory over a RHI image with azimuth according to the planes azimuth angle.	Section **??**
PPI_TRAJ	Overplot the trajectory over a PPI image with elevation according to the planes elevation angle.	Section **??**

Table 15: List of all possible product types for dataset with TRAJ_VOL_COMPOSITE format

Name	Description	Reference
	TODO: of interest?	
SPECTRUM_IMAGE	Plot of a single spectrum	Section **??**
VERTICAL_SPECTRUM_IMAGE	XXX to be described	Section **??**
SINGLE_SPECTRUM_IMAGE	XXX to be described	Section **??**
SAVE_PSR_VOLUME	XXX to be described	Section **??**
SPEC_ALONG	Plot the spectrum along the range, along the azimuth or along an elevation	Section **??**
IQ_ALONG	Plot the IQ data along the range, along the azimuth or along an elevation	Section **??**

Table 16: List of all possible product types for dataset with PSR format

Name	Description	Reference
VOL_HISTOGRAM	**TODO: describe**	**TODO: describe**
PPI_HISTOGRAM	**TODO: describe**	**TODO: describe**
ANGULAR_DENSITY	**TODO: describe**	**TODO: describe**
VOL_TS	**TODO: describe**	**TODO: describe**
SAVEVOL	**TODO: describe**	**TODO: describe**
	TODO: check what to do with legacy products below	
WRITE_PHIDP0	Writes the information of the monitoring of the differential phase offset.	Section **??**
WRITE_ZDRBIAS	Writes the information of the monitoring of the Zdr value in moderate rain.	Section **??**
WRITE_RHOAV	Writes the information of the monitoring of the Rhohv in rain.	Section **??**
WRITE_ZHBIAS	Writes the information of the monitoring of the reflectivity bias.	Section **??**
PLOT_PHIDP0	Plots the differential phase offset as a function of azimuth for one elevation.	Section **??**
PLOT_ZDRBIAS	Plots the Zdr in moderate rain as a function of azimuth for one elevation.	Section **??**
PLOT_ZHBIAS	Plots the Zh bias as a function of azimuth for one elevation.	Section **??**
PLOT_PHIDP0_DAY_TS	Plots the instantaneous evolution of the differential phase offset.	Section **??**
PLOT_ZDRBIAS_DAY_TS	Plots the instantaneous evolution of the Zdr in moderate rain.	Section **??**
PLOT_RHOAV_DAY_TS	Plots the instantaneous evolution of the RhoHV in rain.	Section **??**
PLOT_ZHBIAS_DAY_TS	Plots the instantaneous evolution of the Zh bias.	Section **??**
PLOT_AND_WRITE_PHIDP0_TS	Plots the evolution on a daily basis of the phidp offset. Writes the same information in a file	Section **??**
PLOT_AND_WRITE_ZDRBIAS_TS	Plots the evolution on a daily basis of the Zdr in moderate rain. Writes the same information in a file	Section **??**
PLOT_AND_WRITE_RHOAV_TS	Plots the evolution on a daily basis of the RhoHV in rain. Writes the same information in a file	Section **??**
PLOT_AND_WRITE_ZHBIAS_TS	Plots the evolution on a daily basis of the Zh bias. Writes the same information in a file	Section **??**

Table 17: List of all possible product types for dataset with MONITORING format

Name	Description	Reference
	TODO: Are XVSC substituted/merged by INTERCOMP format?	
PLOT_INTERCOMP	scatter plot between the pairs of time-averaged reflectivity data at each (or all) elevations. Computation of various statistics	Section **??**
PLOT_AND_WRITE_INTERCOMP_TS	Computes various statistics and plots the daily evolution of the median bias	Section **??**

Table 18: List of all possible product types for dataset with XVSC format

Name	Description	Reference
PLOT_INTERCOMP	scatter plot between the pairs of polarimetric data at each (or all) elevations. Computation of various statistics	Section **??**

Table 19: List of all possible product types for dataset with INTERCOMP format

Name	Description	Reference
	TODO: Are those merged in VOL?	
WRITE_2DHIST	writes a 2D histogram in a file	Section **??**
PLOT_2DHIST	plots a 2D histogram	Section **??**
WRITE_HIST	writes a 1D histogram in a file	Section **??**
PLOT_HIST	plots a 1D histogram	Section **??**

Table 20: List of all possible product types for dataset with HISTOGRAM format

Name	Description	Reference
	TODO: what to do with them?	
PPI_RHI_IMAGE	Plot one PPI and up to 3 RHIs on the same image.	Section **??**.
CELL_3D_IMAGE	Plot in 3D composite volume data inside a bounding box centered along a given RHI direction	Section **??**.
RADAR_3D_IMAGE	Plot in 3D composite volume data over a given spatial domain	Section **??**

Table 21: List of all possible product types for COMPOSITE_VOLUME datasets.

TODO: in Pyrad there are also: COLOCATED_GATES_PRODUCTS, QVP_PRODUCTS, GRID_PRODUCTS, COSMO_COORD_PRODUCTS. Clarify how those relate with the IDL framework.

TODO: uncomment the description of the products below here, when defined which ones to keep

6.1 VOL Products