Quality control of radar data for NWP assimilation using PRORAD and CONRAD

Morten Salomonsen
morten.salomonsen@met.no

Martin S. Grønsleth
martin.s.gronsleth@met.no

Christoffer A. Elo
christoffer.elo@met.no

Trygve Aspenes
trygve.aspenes@met.no

May 22, 2012

Abstract

This document gives a short overview of the ProRad system, and its capabilities regarding filtering of weather radar data for NWP assimilation. ConRad is also described, a tool for converting radar data between various formats. ProRad and ConRad can work together to allow vendor specific formats to be filtered by ProRad, and converted to the format required by HARMONIE. The core part of ConRad, and the quality control framework of ProRad, have been developed as part of a partially externally financed project with the Norwegian hydro power industry (NFR project ES439901/193048, NFR - The Research Council of Norway, RCN). ConRad is released as open source software, under GPLv2. ProRad will soon be available as open source software under the same licence. A fully functional data chain from raw radar data, via quality control and format conversion, to updates of the model state is reported. This system is flexible, and well suited for implementation of further QC algorithms, and resilient with respect to file format changes.

1 PRORAD

1.1 What is PRORAD

The mission statement formulated at the start of the ProRad project was

> *Construct a flexible, scalable, event driven and distributed system for processing weather radar data, capable of handling all necessary processing of 2D and 3D datasets.*

which still is the guideline for the continued development of the system. Development of the ProRad system started in the autumn of 2005 at met.no. The first operational version of ProRad was installed at met.no in June 2008. The QC algorithms developed as part of the joint 3 year long project with the hydro power sector was declared pre-operational November 2011, and operational January 2012. ProRad is currently used for all processing, and visualisation of radar data at met.no. Most of the ProRad software and libraries are written in C/C++.

To achieve the goal of the mission statement, ProRad is not built as a single, monolithic piece of software. The ProRad system consist of a collection of small, specialised programs, each responsible for a well defined step in the processing of the radar data. These specialised programs are called modules. Synchronisation of the processing of radar data between these modules is done by passing

messages between the modules. This will be describe in more detail later. A collection of modules to perform a set of processing task is what we call a *node*, or *ProRad node*.

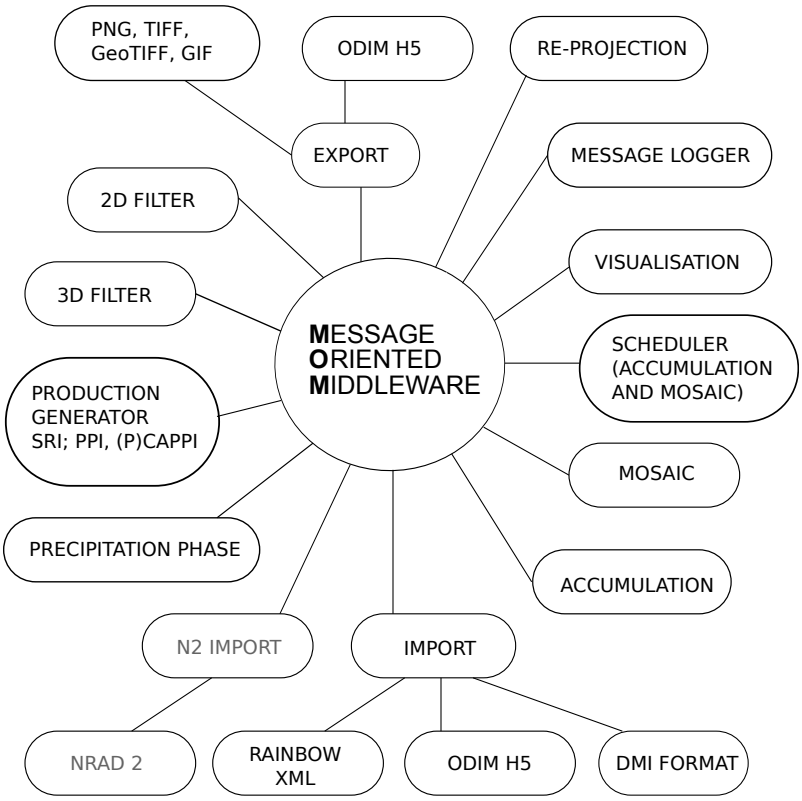


Figure 1: Diagram of ProRad

The diagram illustrates met.no's operational ProRad node, in its current configuration. A more detailed description of the diagram can be found in appendix B.

1.2 Designed from the start to handle QC

Filtering of the radar data was part of the ProRad design process from an early stage. As a consequence flags to indicate whether a pixel, or data point, contains non meteorological echoes or not, are an integral part of the internal ProRad dataset objects, and the information model used by ProRad to store the datasets.

We have chosen to implement the quality information as a set of flags rather than a single index, or set of indices. We have, to mention a few, included flags for various types of clutter, such as sea or ground clutter, and include flags to indicate blocked sectors and to what extent a sector is blocked. The argument for using a set of flags rather than indices is that the requirements may vary depending on the end use of the data. NWP assimilation requirements differ from those of visual inspection of the data. The flags enables the end user, or provider of the data, to calculate a quality index depending on the intended use of the data.

1.3 The design philosophy

The goal of the design for ProRad was to parallelise processing of data as much as possible. One way of achieving this is to use threads within a single, monolithic program to perform various steps of the processing. Another approach is to divide the various processing steps between several programs. Both approaches require some form of synchronisation of the processing steps, but the latter opens up the possibility of distributed processing of the data. The latter will also make it easier to create a system that is flexible and easy to extend or modify. We have chosen the latter approach for ProRad.

1.3.1 An asynchronous network of producers and consumers

Processing of radar data can be broken down into smaller steps, or set of task. The following example lists a set of steps needed to make a mosaic, also referred to as a composite, of PCAPPI images for example

- collect or import the radar data
- re-project the data to the map-projection and scale of the mosaic
- collect datasets for the mosaic area, and the nominal time of the mosaic
- produce the mosaic when all required datasets are available, or the cutoff time for the production has been reached

and all these steps may be handled by separate programs, all independent of each other, with information passed between them to synchronise the processing. One way of synchronisation is to pass messages between the programs through a dedicated program which collects and distributes the messages. This dedicated program is know as a Message Oriented Middleware, or MOM. What we have now is what is known as an *asynchronous network of producers and consumers*.

1.3.2 Synchronisation of the processing

The messages passed between the various programs, or modules that make up the ProRad system, contains a set of meta data which uniquely identifies a dataset. When a module has completed its processing, it will send a message to the MOM, and the MOM will distribute the message as a multicast message to all the other modules connected to the MOM.

The modules receiving the messages, and who find a match in their configuration file, will process the dataset, and send a new message with meta data describing the resulting dataset back to the MOM, and so forth[1]!

1.3.3 Flexibility

By sending messages through the MOM, none of the modules need to know about the existence of any other modules than the MOM. This is an important point, because this means modules can be added or removed without having to restart or reconfigure the remaining modules. The only visible change to the other modules will be that new messages appear from the added modules, or messages will be

[1] To reduce the load on the system it is important that all modules remain passive if no new messages are available. There are several ways of achieving this, but this is beyond the scope of this document. All ProRad modules remains passive if no data are available for processing.

missing from the removed modules. Should a module stop, all you need to do is restart that particular module.

Removing and/or adding modules will have an implication on the overall processing performed by the collection of modules connected to a MOM, *i.e.* the ProRad node. This means that processing, or functionality, may be added without disrupting the processing the ProRad node is already performing. The processing performed depends on the configuration of the individual modules, and the type of modules the node is running.

1.3.4 Event based systems vs scheduler based systems

The processing at a certain step in an *event based* processing chain will be triggered by output from the previous processing step, regardless of the processing time required by the previous steps in the processing chain. The event that will trigger the entire chain of events is the ingestion of a dataset. This may be done directly from the radar control software; at met.no this is triggered by the Selex Gematronik software package Rainbow, and executed as soon as the desired dataset is available.

As mentioned earlier, we have built ProRad as an asynchronous network of producers and consumers, and synchronisation of processing among these producers and consumers is done by passing messages *i.e.* an *event based system*. Such system are well suited for streams of real-time data where arrival time of datasets cannot be determined with a high level of accuracy.

With a *scheduler based* system the processing will be run at fixed times, and at given intervals. The system will use a time trigger to initiate the processing, regardless of whether datasets are available for processing or not. Such systems are well suited for datasets that arrive at predefined intervals, but may require more logic to handle situations where data arrive at unscheduled intervals.

1.4 File format considerations

ODIM H5 is an HDF5 based exchange format, and the definition of its content, data and meta data, will always be the subject of plenary discussions and decisions within a community. The ODIM model will be a least common denominator solution, and may not cater to the needs of individual users, even if there are possibilities to store locally defined data and meta data.

We switched from a HDF5 based format to an XML based format a couple of years back to enable us to store the meta data and data we needed for the processing of 2D and 3D data. The meta data include flags to indicate *e.g.* whether a datapoint may be contaminated by clutter or not.

ProRad is capable of reading and writing polar volume data in the ODIM H5 format. This means we are be able to read polar volumes stored in the ODIM H5 format, and use the ProRad QC algorithms on the dataset. The ProRad XML files can then be converted to the format required by the chosen NWP model for assimilation, taking into consideration the added QC information in the process[2]. ConRad, described in section 3, is a tool capable of doing such a conversion.

1.4.1 QC flags

In addition to the dataset and meta data, a set of flags are stored along with the dataset. The ProRad dataset object stores these flags, and the flags are calculated for each pixel of the dataset. The currently defined flags are (version 1.0 of the flags definition)

[2]The HARMONIE model for example, requires a BUFR format. The BUFR format specification is written by Meteo-France, and the format is referred to as MF-BUFR.

is_nodata (2D datasets) this flag indicates if the pixel is outside the scanned area, value 1 or 0 *i.e.* True or False

is_lowele (2D datasets) this flag indicates if the pixel is from the lowest scan elevation, value 1 or 0 *i.e.* True or False

is_highele (2D datasets) this flag indicates if the pixel is from the highest scan elevation, value 1 or 0 *i.e.* True or False

is_blocked this flag indicates if the pixel is in a blocked sector, value 1 or 0 *i.e.* True or False

block_percent this is a 7 bit flag indicating the percent of geometric blocking of the beam in the blocked sector, and is calculated from the point of the obstacle, value 0 - 100

is_seaclutter this flag indicates if the pixel has been identified as sea clutter, value 1 or 0 *i.e.* True or False

is_groundclutter this flag indicates if the pixel has been identified as ground clutter, value 1 or 0 *i.e.* True or False

is_otherclutter this flag indicates if the pixel has been identified as other clutter, value 1 or 0 *i.e.* True or False

clutter_probability this is a 7 bit flag indicating the probability of the pixel being clutter, value 0 - 100

classification this is a two bit flag used to indicate what phase the precipitation would have at ground level, and the possible values are: 0 " not classified", 1 "rain", 2 "sleet", or 3 "snow"

2 Quality control

Most of the non-meteorological targets are eliminated in the radar signal processor with the aid of Doppler velocity. However, some residual clutter still exists in the dataset, which need to be correctly identified for further usage in the numerical weather prediction models.

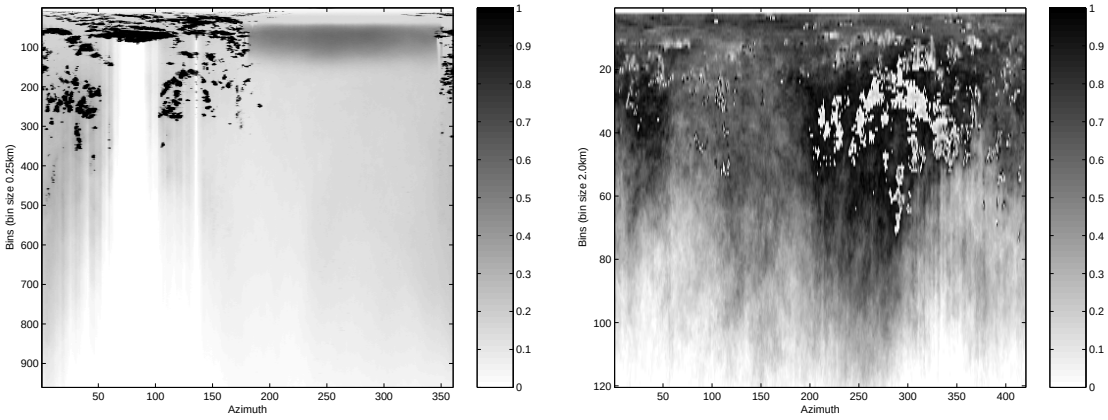
The datasets consisting of polar volume data are processed with a set of algorithms according to a given XML configuration. Each radar site is described in the configuration with elevations and parameters so that it is flexible and easy to optimize the algorithms with respect to local conditions, *e.g.* the Bømlo radar which is contaminated from both strong sea and ground clutter and therefore needs a more careful parameter tuning as opposed to the Hurum radar which is not located along the coast.

The algorithms are only dependent on the Doppler and the non-Doppler filtered radar reflectivity values, together with a mask that indicate if the reflectivity value is over sea or not.

2.1 Ground clutter

Residual ground clutter from the signal processor is easily detectable visually by looping through individual radar products and look for stationary targets. This can also be done with an algorithm that gathers sufficient data to achieve an approximation to a probability function of an event occurring. For the Norwegian radars we approximate this function by using seasonal data (spring, summer, fall and winter) from the previous year. The method loops through all the seasonal data and counts every

radar reflectivity value above a given threshold as an occurrence and stores the result as *frequency of occurrence* in a binary file for all elevations. Flagging the ground clutter is then done by looking up the correct frequency of occurrence file, and a threshold in the XML configuration to create a mask that indicates if the radar reflectivity is contaminated by ground clutter or not.



(a) Probability function for the norwegian Bømlo radar. The function is generated from 15 min non-Doppler filtered polar volume data for the summer period.

(b) Probability function for the swedish Kiruna radar.The function is generated for 3 hours Doppler filtered polar volumes from the test period.

Figure 2: The difference between using Doppler and non-Doppler filtered data. The swedish data is also not sufficient due to reduced available data.

2.2 Sea clutter

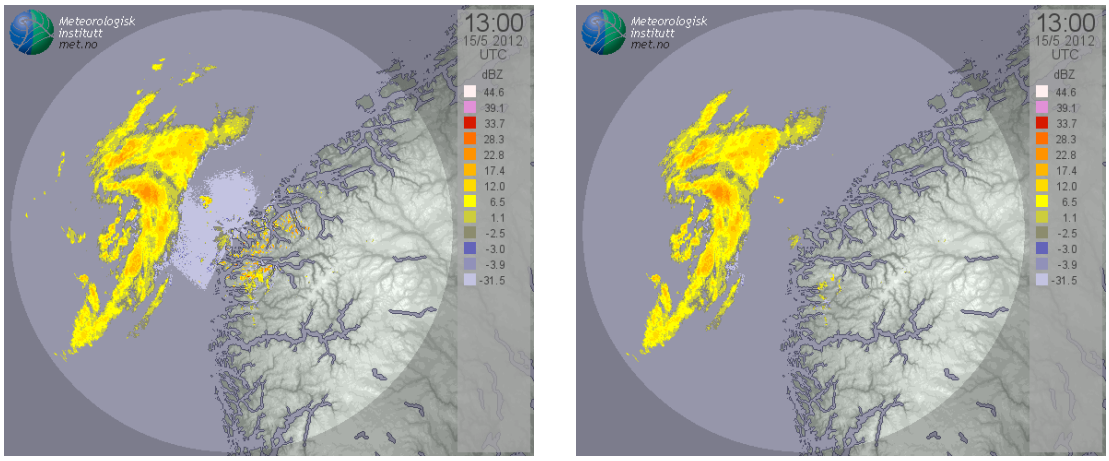
Identifying sea clutter is based on solving a boundary value problem by propagating a front from multiple initial small reflectivity values (<-10 dBZ) near the radar to the boundary of the sea clutter regions. The boundary of the sea clutter can be described as the arrival time function of a closed curved evolved from the the initial values. This function is dependent on the radar reflectivity values as the speed function which determines where the front moves fast or slow. The sea clutter regions are finally flagged by thresholding the time arrival function.

We have chosen to use the Fast Marching Method[9] to numerically solve the boundary value problem. This method is easily extended to polar and spherically coordinates for both 2D and volume radar data. The figure 3a shows sea clutter and precipitation near the Stad radar, the image to the right shows the same data but with the flagged pixels removed.

2.3 Other clutter

Ship traffic in the North sea is often corrupting the radar data as small peaks with high reflectivity, see figure 4a. The ships have a small extent in range and are often larger in the azimuth extent direction since the ships are affecting the sidelobes as well. An automated method has been developed to recognize these patterns and flag it as clutter. The method is based on looping through all the reflectivity values that are over sea and clustering regions with high reflectivity values (>35 dBZ). Within the clustered regions the method calculates the area and pixel count, azimuth and range extent, maximum and background reflectivity value and use these values to identify the regions as clutter or precipitation.

The figure 4a shows the first elevation zoomed-in from a polar volume dataset. All the reflectivity seen in the image is spurious noise with high reflectivity that comes from ships and some sea clutter

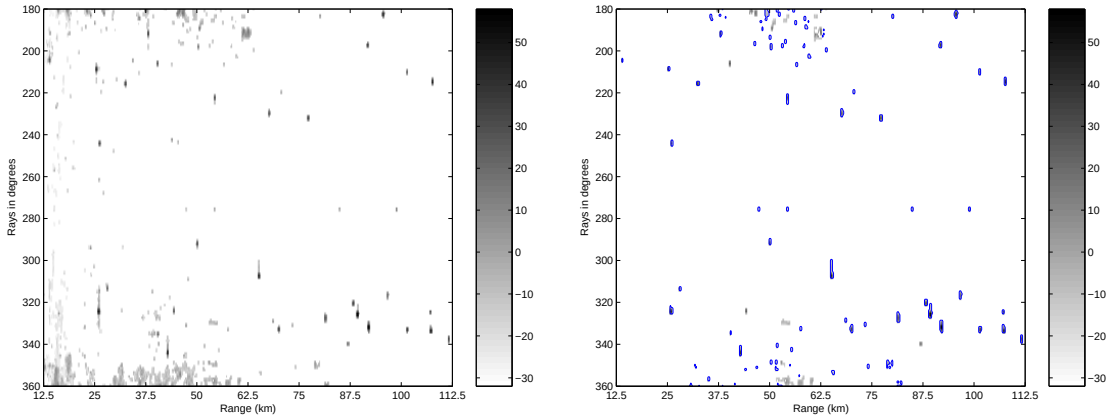


(a) Precipitation and clutter.

(b) Removed clutter flagged pixels.

Figure 3: Radar Stad with precipitation and clutter from sea and ground.

with lower reflectivity. In the figure to the right 4b, the flagged data points from the fast marching method algorithm has been removed. The blue contour plots shows the successfully identified spurious noise from the pattern recognition method.



(a) Sea clutter and ship traffic

(b) Removed sea clutter and identified the ships with blue contours

Figure 4: Radar Bømlo with high intensity ship traffic in the lowest elevation.

2.4 Beam blockage

The amount of a radar beam which is blocked by *e.g.* mountains is an interesting quality flag for radar data. Beam blockage is calculated using the beam height relative to different Digital Elevation Models (DEMs). ProRad is using 3 different DEMs: one model with 25m spatial resolution covering the Norwegian mainland, one with 100m spatial resolution covering Norway and the western parts of Sweden, and finally the GTOPO30 dataset.

For each pixel along an azimuth gate, or beam, at a given elevation, the height of the radar beam is

calculated and compared to the height of best available DEM for each range bin[3]. If the DEM height is lower than the radar beam, no blockage for this range bin is set. If the radar beam is partly lower than the DEM height, the geometric percentage of the area blocked is calculated[4]. This partly blocked value is considered for the rest of the range bins along that azimuth gate and can only increase from this point on. If the radar beam is completely below the DEM height, the rest of the range bins along that azimuth gate are marked as blocked.

In ProRad a block/no block flag, and a blocking percent amount is calculated for each pixel and stored as flags. These flags can then be used when putting several radars together in a mosaic selecting the radar with best quality if two or more radars is overlapping. The calculations can be done both for 2d data (*i.e.* PCAPPI) and 3d (volume data).

3 ConRad

ConRad (Conversion of Radar data) [5], is a tool for converting radar data between various formats. The development started at met.no in 2010, as part of a partially externally financed project with the hydro power industry (Energy Norway, SINTEF and met.no, with funding from The Research Council of Norway, NFR ES439901/193048). The reason for creating ConRad, was to take advantage of the methods developed by Meteo-France [6, 10, 1] for assimilating radar data observations in the Harmonie NWP (numerical weather prediction) model. For this to be possible, the local radar data had to be converted to the format recognized by the model, which currently is Meteo-France BUFR (MF-BUFR).

Initially, ConRad only supported conversion from PRORAD XML to MF-BUFR, but because other HIRLAM member countries also needed such a tool, ConRad was rewritten to allow for more modular use, where support for any local format can be added using a common interface. Table 1 and 2 show what output and input formats that is currently supported. Although ConRad has its own internal types, it does not define a file format.[5]

ConRad is currently being used and developed by several HIRLAM member countries; Austria, Croatia, Denmark, Ireland, The Netherlands, Slovenia, Spain, Sweden and maybe more, in addition to Norway.

Format	Sub type
MF-BUFR	Cartesian reflectivity only
MF-BUFR	Cartesian radial wind only
MF-BUFR	Polar combined reflectivity and radial wind
MF-BUFR	Polar reflectivity only
MF-BUFR	Polar radial wind only
PRORAD XML	Polar reflectivity
PRORAD XML	Polar radial wind

Table 1: Output formats supported by ConRad.

The ProRad read and write methods have been made available for ConRad through a wrapper, mapping the C functions to Fortran 95. The internal data types of ConRad closely resemble those

[3]The DEM with highest spatial resolution.

[4]The radar beam is considered a geometric circle.

[5]This is a feature, as there is really no need for yet another radar file format.

Format	Sub type	Comment
HDF5	OPERA/ODIM H5	Polar data*
	SMHI	Polar data
	KNMI	Polar data, reflectivity and radial wind
BUFR	MF-BUFR (Meteo-France)	Cartesian data, reflectivity and radial wind
	OPERA BUFR (AEMET)	Polar data, reflectivity and radial wind
	OPERA BUFR (Met Éireann)	Polar data, reflectivity and radial wind
IRIS RAW	Croatia, Slovenia	Cartesian data (interpolated), reflectivity and radial wind
PRORAD XML		Polar data, reflectivity and radial wind

Table 2: Input formats supported by ConRad. *Supported using the ProRad library (direct support is under development).

of ProRad. This has the advantage that quality flags can be transferred directly without loss between ProRad and ConRad. Also, since the raw data structures themselves are quite universal, both tools are well suited to host data originating from other formats.

Although there is no one-to-one correspondence between the quality flags in ProRad/ConRad and the Meteo-France BUFR format, ConRad has the advantage to make qualified decisions on the flag mapping. Obviously, one should avoid to perform such mappings repeatedly between non-conforming formats in order to reduce the loss of quality flag precision. In this case, the mapping is done only once.

In addition to the quality flags (as described in 1.4.1) important radar specific quantities such as the radar constant and the radar sensitivity are transferred in the ProRad→ConRad→MF-BUFR chain. More on this in Section 5.

ConRad [5] is licensed with GPLv2+ (GNU General Public License, version 2, or at your option any later version) [4]. This means that anyone is free to use and extend/modify ConRad as they please, as long as the code is contributed back. See the license for the full requirements.

3.1 Flexibility

One of the most important features of ConRad is that it serves as a buffer with respect to possible future changes of model system or model input format. Now, as several institutes have invested time and efforts in creating support for their local format in ConRad, a possible change in model input format would only require *one* output plugin to be made for ConRad. That new output format would then immediately be available for all local formats supported by ConRad.

This way, ConRad serves as a flexible tool for serving various model input formats.

4 From radar via ProRad to NWP assimilation

A ProRad node for processing polar volumes for NWP assimilation will typically include an import module where polar volumes are ingested, modules to perform various filtering and flagging of the data, and last but not least a module invoking ConRad to convert the polar volume data to for example the BUFR format required by the HARMONIE system, MF-BUFR. All processing will be event

based, which means that as soon as the polar volume is ingested, each step of the chain will trigger the next step until the data are stored as MF-BUFR by ConRad.

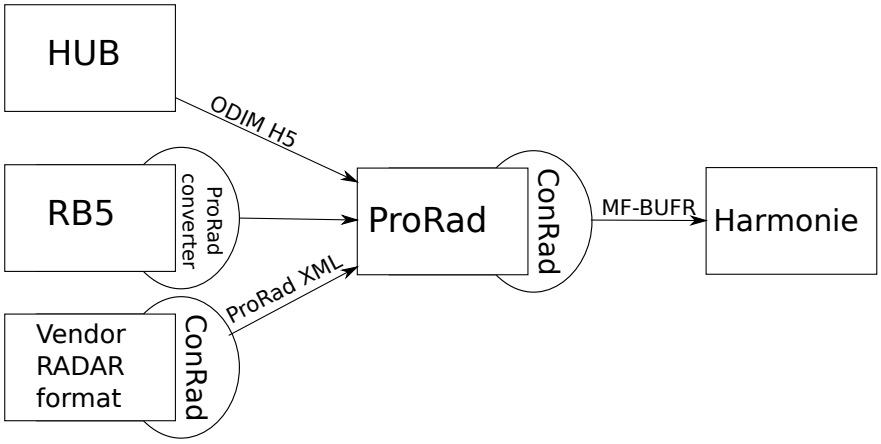


Figure 5: Conversion ad processing of radar data with ProRad and ConRad

The figure illustrates the setup for conversion and processing of radar data, from raw formats, via ProRad/ConRad and to the NWP model Harmonie.

5 Assimilation of radar data

When radar data is to be assimilated in a NWP model, the quality requirements are much stricter than when *e.g.* presenting the data as an animation on the web. Whereas a human eye quite easily can filter out non-meteorological echos, the model system has no way of knowing if an observation is to be trusted or not, unless it is explicitly marked. If raw radar observations of *e.g.* reflectivity were to be used directly in the model, it could be disastrous for the forecast, as *e.g.* non-meteorological echos (clutter) would be interpreted as true precipitation observations. Quality control is therefore crucial for radar data assimilation.

Within the Energy Norway project mentioned earlier, a conversion chain for Norwegian radars has been set up and tested in a pre-operational setup. This setup includes both reflectivity and radial wind data from Norwegian radars and have shown successful assimilation (3DVAR; three-dimensional variational) in the Harmonie/Arome model in a non-hydrostatic setup and 2.5 km grid spacing. Harmonie model will be the successor of the currently operational HIRLAM model.

One important aspect of assimilating radar data in NWP, is to be able to use observations where no echo is measured but is within the visibility of the radar. These pixels should be marked as "no echo measured" as opposed to "no measurement done". If there is no way to distinguish between these two, the model cannot know if an observation represents a dry spot, or if it is *e.g.* outside the range of the radar. ProRad and ConRad, as well as the Meteo-France BUFR format, all support such a distinction.

In addition to this rather obvious requirement, the assimilation system demands even more from the radar data, namely information about the radar constant and the sensitivity of the specific radar. These quantities are used to extract as much information as possible from each observed pixel, and to make qualified decisions based on the range of each pixel in combination with the actual signal to noise ratio of the radar. This is very important in order for the radar simulator in the model to function correctly, and for the assimilation system to carry out a realistic drying of the model when assimilating non-rainy echoes.

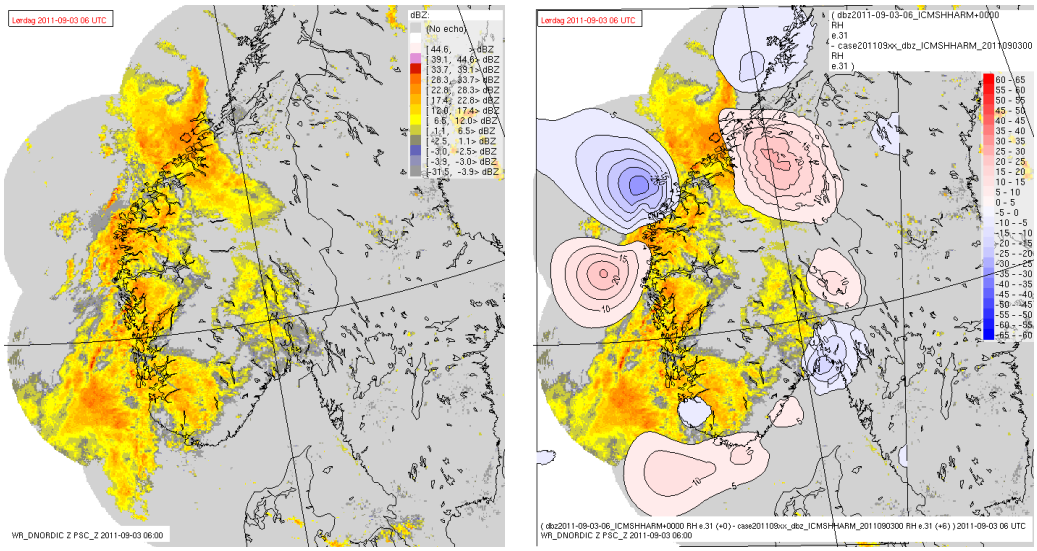
Unfortunately, not all radars provide these quantities in the meta data, and not all radars define these quantities the same way (differences can be found even among radar software from the same manufacturer). It is therefore essential to carefully check the definitions used, and only transfer corresponding values or recalculate them.

5.1 Assimilation of radar reflectivity

Results from radar reflectivity assimilation are presented in [8]. The report demonstrates a technically working pre-operational system for assimilating radar data. Verification scores of a two week impact study show positive impact when including radar reflectivity in the analysis system.

The method used for reflectivity assimilation was developed by Meteo-France [6, 10, 1], and consists of combining 1D Bayesian and 3D-variational assimilation schemes, where reflectivity data is assimilated as unidimensional (1D) humidity retrievals into the three-dimensional (3DVAR) assimilation system.

Figure 6 is taken from [8] and shows an example of how radar reflectivity observations are able to both dry and moist the model at appropriate places.



(a) Observed reflectivity at analysis time (Pseudo-CAPPI, *i.e.* reflectivity at about 500-700 m)

(b) Analysis minus first guess (model background) in relative humidity (RH) at model level 31

Figure 6: Analysis increments of relative humidity.

One can see that assimilation of radar reflectivity is increasing/decreasing relative humidity at appropriate places. Note that the pseudo-CAPPI plot does not reflect the full volume of observed reflectivity (which is used in the assimilation).

5.2 Assimilation of radar radial winds

The conversion of radar wind observations closely follows that of reflectivity, and the conversion chain from local formats to the format the model expects has been tested in a pre-operational environment. These observations have not yet been quality controlled beyond the internal algorithms of the radar software itself (which for the Norwegian radars are in fact quite good, especially for de-aliasing). However, the model itself does perform some filtering and quality control of radial wind data.

Further improvements of this will be conducted within a newly started project “RapidWind”, which is in collaboration between met.no and Norwea with partial funding from The Research Council of Norway. Norwea is an interest and lobbying organisation working to promote Norwegian renewable energy production.

References

[1] Olivier Caumont, Véronique Ducrocq, Geneviève Jaubert, and Stéphanie Pradier-Vabre. 1D+3DVar assimilation of radar reflectivity data: a proof of concept. *Tellus A*, 62(2):173–187, 2010.

[2] Richard J. Doviak and Dušan S. Zrnić. *Doppler Radar and Weather Observations*. Academic Press, 2 edition, 1993.

[3] Christoffer A. Elo. Correcting and quantifying radar data. met.no report 2/2012, met.no, January 2012.

[4] Free Software Foundation, Inc. GPLv2. http://www.gnu.org/licenses/gpl-2.0.txt, 1989, 1991.

[5] met.no/M. S. Grønsleth et.al. ConRad. ftp://ftp.met.no/projects/radar/conrad-snapshots (open) https://svn.hirlam.org/branches/harmonie-36h1_radar/util/conrad (restricted access) http://lists.met.no/mailman/listinfo/conrad (mailing list, registration open).

[6] Thibaut Montmerle, Eric Wattrelot, Claudia Faccani, Olivier Caumont, Marian Urasek, and Günther Haase. Regional scale assimilation of radar data at Météo-France. HIRLAM Technical Report 68, Météo-France, July 2008.

[7] Martin S. Grønsleth and Christoffer A. Elo. Assimilating radar observations into the next generation numerical weather prediction model. *NOTUR II*, META Number 4:17–19, 2011.

[8] Martin S. Grønsleth and Roger Randriamampianina. Assimilation of radar refectivity data in harmonie. met.no report 1/2012, The Norwegian Meteorological Institute, January 2012.

[9] James A. Sethian. A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 93, pages 1591–1595, 1996.

[10] Eric Wattrelot, O. Caumont, S. Pradier-Vabre, M. Jurasek, and G. Haase. 1D+3Dvar assimilation of radar reflectivities in the pre-operational AROME model at Météo-France. In *ERAD2008*, 2008.

A Operational issues

A.1 Error handling

The ProRad system is designed to run with minimum level of operator intervention. To achieve this, certain features have been built into the system to handle error situations like overflowing disks and modules crashing unexpectedly.

Each module is implemented with a watchdog functionality, which means that the module will restart itself if, for some reason or other, it should crash. At restart a message will be sent to syslog. The MOM is the most crucial part of a ProRad node. If the MOM stops, all modules will detect this and stop after sending a message to syslog.

If the the disk, where a given module writes its files should become full, the module will continue to run but fail to write the datasets to disk. To resolve this situation, all the operators need to do is to delete files to free disk space. As soon as disk space is freed, the module will resume its processing[6].

A.2 Track record

On average the processing performed at the met.no operational ProRad node generates roughly 49000 events per day. Over the period stretching from June 2011 to October 2011 we had 72 events resulting in an error condition that could not be explained by the information obtained through syslog messages and other log messages created by the ProRad modules. After a code review on the suspected modules, a memory leak was found, which explained 63 of these 72 error conditions.

A.3 Ease of configuration and reconfiguration

A ProRad node consist of a group of modules. Each of the ProRad modules have their own separate configuration. The only link between the modules is the messages passed between the modules. All modules can be stopped or restarted independently of all the other modules. So if the configuration for one or more modules needs to be changed, these modules may be restarted without having to restart any other modules. If there is a need to extend the processing capability of the ProRad node, new modules may be added without disruption to existing modules running at the ProRad node.

B ProRad at met.no

The setup of met.no's operational ProRad node is illustrated in figure 7 on page 15. The following modules are part of the current configuration (clockwise starting with the Import module)

Import This is the module responsible for ingesting radar data from external systems, *e.g.* RainBow, and data collected from DMI.

N2 import This module communicates with the old NORDRAD software NRAD2 (NRAD2 is currently used for transporting data between FMI, SMHI and met.no. The NORDRAD community is discussing the replacement of the NRAD2 software.)

Precipitation phase This module uses NWP data to determine the most probable type of precipitation found at ground level. The classification is stored as flags in the dataset.

[6]However there is no backlog for the datasets which may have arrived during the time the disk was full. Data will be lost, but ProRad has functionality which may enable a replay of the messages which where available for processing during the error state.

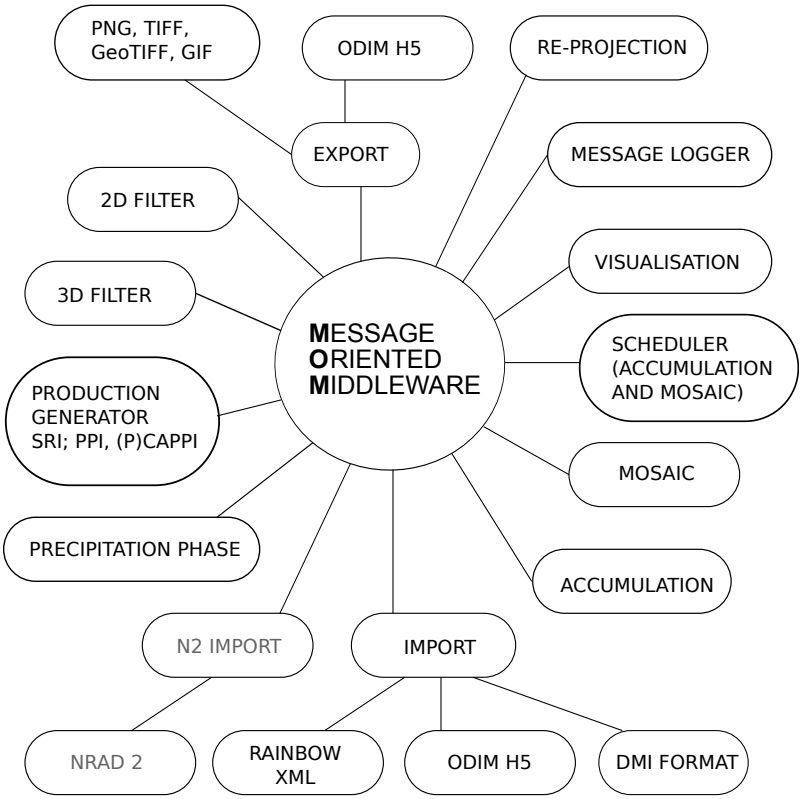


Figure 7: ProRad at met.no

Product generator This module will make 2D products from the polar volume data. The polar volume may be QC filtered or not, the configuration will decide which version will be used for product generation.

3D filter This module will perform QC on polar volume data.

2D filter This module can perform a subset of QC algorithms on 2D data imported from external systems such as RainBow (used for datasets where the polar volume is not available.)

Export This module can be used to convert datasets to graphics, such as GIF, or trigger file format converters to write the dataset as ODIM H5 files, or MF-BUFR for NWP assimilation (the latter through invoking ConRad.)

Re-projection This module will re-project 2D products from one map projection to another, which is necessary when the datasets from several radars are to be used in a mosaic, or composite.

Message logger This module will record all messages passed by the other modules and write them to disk.

Visualisation This module is used to make visualisations, such as TIFF/geoTIFF.

Scheduler This module is central to the production of mosaics and datasets for accumulated precipitation. The messages sent from this module will be used by the modules making the mosaics and datasets for accumulated precipitation. This module will determine what datasets will be

used for the mosaics and accumulations. There is one set of schedulers for mosaics, and one set of schedulers for accumulations.

Mosaic This module will make the mosaics based on the information in the messages from the Scheduler module.

Accumulation This module will make datasets with accumulated precipitation based on the information information given in the message from the scheduler responsible for selecting products for the accumulated precipitation datasets.

C The ProRad XML file format

The ProRad XML file format basically consists of two main sections. One section for the meta data, `<productmeta>`, and one section for the datasets, `<datasection>`.

C.1 Meta data

The information in the meta data section, `<productmeta>`, have been grouped into a set of sections of which some may be omitted depending on the type of product. There is a section for radar specific meta data, `<radarinfo>`, described in table 3 on page 17, and this section is found in single site 2D and 3D products.

Furthermore there is a section with details on the extent of the area covered by the product, `<area>`, described in table 4 on page 17; the content of this section is omitted for polar volumes and WRWP products.

Then there is a section `<datasets>` described in table 5 on page 18 for 2D products and in table 6 on page 19 for 3D, describing the datasets and flags stored in in the data section. The XML XPATH addresses for the corresponding datasets and flags are also given in this section.

C.2 Datasets

All datasets are compressed and BAS64 encoded before they are added to the XML file. In the current version of the XML format, the flags are stored as 24 bit words, the datasets as 8 bit words, and the start angles dataset for the elevations as a C type double. The ProRad interface for reading and writing radar products as XML files, utilises the libxml2 compression feature to minimise file size. Compression and decompression of the files are transparent to the end user.

The QC flags described in section 1.4.1, on page 4, are stored as 24 bit words, distributed over three 8-bit unsigned chars. The ProRad file IO API handles packing and unpacking these flags transparently to the end user, and in the ProRad dataset objects the flags are stored as bit-fields to enable a simple access to the individual flags of the individual data points.

Tag	Sub-tag	Attribute	Content
radarinfo		sitename	Name of the place or radar site
		wmoid	WMO ID
	site	geoid	sphere or WGS84
		lat	Latitude for given geoid
		long	Longitude for given geoid
		datum	Default value WGS84
	height		Station height msl
	radar_system		Radar manufacturer
	software		software system used to process data
	wavelength		Wavelength of the radar system
	beamwidth		Antenna beam width

Table 3: Overview of the <radarinfo> section tag, sub-tags and attributes currently supported

Tag	Sub-tag	Attribute	Content
area		name	radar alias or name of mosaic area
	projdef	alias	Alias for the projection
			Proj4 projection definition
	cartesian	north	Cartesian coordinates in the projection plane
		south	
		east	
		south	
	size	xsize	Size of the area, and resolution in the x- and y-direction
		ysize	
		xres	
		yres	
	extent	north	The lower left hand and upper right hand corners of the area given as latitude and longitude values. This may be calculated from the values in cartesian
		south	
		east	
		south	

Table 4: Overview of the <area> section tag, sub-tags and attributes currently supported

Tag	Sub-tag	Attribute	Content
nodes			The nodes, or radars which are included in a mosaic, or composite. **Omitted for single site products, polar volumes and WRWP products**
processing		classification	Whether if precipitation phase is set in the flags section or not, "0" or "1" *i.e.* True or False
		clutter_filtered	Whether if the product has been clutter filtered or not, "0" or "1" *i.e.* True or False
		vpr_corrected	Whether if the polar volume has been vpr corrected or not, "0" or "1" *i.e.* True or False
set		id	Id number of the corresponding dataset
	convparams	alfa beta	$data\_value = \alpha X + \beta$
	data	descr	View, current default value top
		datatype	What datatype is stored *e.g.* dbz, v
		min	Lowest data value stored
		max	Largest data value stored
		levels	Number of levels
		nomeasure	Value for scanned datapoints with no measurement
	address	set	XML XPATH address of the dataset, values data and flags

Table 5: Overview of the 2D products <datasets> section tag, sub-tags and attributes currently supported. **Please note that the <datasets> tag itself does not contain any data or attributes**. All tags listed in the table are sub-tags of the <datasets> tag.

Tag	Sub-tag	Attribute	Content
processing		classification	Whether if precipitation phase is set in the flags section or not, "0" or "1" *i.e.* True or False
		clutter_filtered	Whether if the product has been clutter filtered or not, "0" or "1" *i.e.* True or False
		vpr_corrected	Whether if the polar volume has been vpr corrected or not, "0" or "1" *i.e.* True or False
set		id	Id number of the corresponding dataset, *e.g.* elevation number
	convparams	alfa beta	$data\_value = \alpha X + \beta$
	data	datatype	What datatype is stored *e.g.* dbz, v
		min	Lowest data value stored
		max	Largest data value stored
		levels	Number of levels
		nomeasure	Value for scanned datapoints with no measurement
	elevation	angle	Elevation angle
		rangstep	Size of each rangebin, unit km
		stoprange	Scan range for the elevation
		rangebins	Number rangebins
	azimuth	azispeed	Azimuth speed, unit degrees/second
		anglestep	Azimuth resolution, unit degrees
		startangle	Angle of the first azimuth gate
		rays	Number azimuth gates
	radarconst	noise_power_dbz	Noise power
		radar_constant	Radar constant
	time	start	Start time for elevation scan
		end	End time for elevation scan
	address	set	XML XPATH address of the dataset, values data, flags and startangles

Table 6: Overview of the 3D products <datasets> section tag, sub-tags and attribute currently supported. **Please note that the <datasets> tag itself does not contain any data or attributes**. All tags listed in the table are sub-tags of the <datasets> tag.

C.3 Example meta data sections for mosaic and polar volumes

C.3.1 Mosaic, or composite image

```
<?xml version="1.0" encoding="UTF-8"?>
<radardata version="1.0.1" time="20120504T124500Z" epoch="1336135500" type="mos" product="pcappi">
  <productmeta>
    <radarinfo sitename="utm32ebl" wmoid="000000">
      <site geoid="sphere" lat="0.000000" long="0.000000" datum="WGS84"/>
      <site geoid="WGS84" lat="0.000000" long="0.000000" datum="WGS84"/>
      <height>0.0</height>
      <radar_system>NA</radar_system>
      <software>NA</software>
    </radarinfo>
    <area name="utm32ebl">
      <projdef alias="utm32">+proj=utm +zone=32 +datum=WGS84 +ellps=WGS84</projdef>
      <cartesian north="7302000.0000" south="6230000.0000" east="827000.0000" west="41000.0000"/>
      <size xsize="787" ysize="1073" xres="1000.00" yres="1000.00"/>
      <extent UR_lat="65.672789" LL_lat="55.994661" UR_lon="16.125904" LL_lon="1.633815"/>
    </area>
    <datasets>
      <nodes>nosta,norsa,nobml,nohgb,nohur</nodes>
      <processing classification="0" clutter_filtered="0" vpr_corrected="0"/>
      <set id="0">
        <convparams alfa="0.500000" beta="-32.000000"/>
        <data descr="top" datatype="dbz" min="0.00" max="0.00" levels="255" nomeasure="0"/>
        <epochs start="0" end="0"/>
        <address set="data">//datasection/dataset</address>
        <address set="flags">//datasection/flagset</address>
      </set>
      <flags bits="24">
        <flag name="nodata" pos="0" size="1">
          <values val="1" descr="nodata"/>
```

```
    </flag>
    <flag name="lowele" pos="1" size="1">
      <values val="1" descr="lowele"/>
    </flag>
    <flag name="highele" pos="2" size="1">
      <values val="1" descr="highele"/>
    </flag>
    <flag name="blocked" pos="3" size="1">
      <values val="1" descr="blocked"/>
    </flag>
    <flag name="block_percent" pos="4" size="7">
      <values descr="5 percent steps"/>
    </flag>
    <flag name="seaclutter" pos="11" size="1">
      <values val="1" descr="clutter"/>
    </flag>
    <flag name="groundclutter" pos="12" size="1">
      <values val="1" descr="clutter"/>
    </flag>
    <flag name="otherclutter" pos="13" size="1">
      <values val="1" descr="clutter"/>
    </flag>
    <flag name="clutterprobability" pos="14" size="7">
      <values descr="1 percent steps"/>
    </flag>
    <flag name="classification" pos="21" size="2">
      <values val="0" descr="no data"/>
      <values val="1" descr="rain"/>
      <values val="2" descr="sleet"/>
      <values val="3" descr="snow"/>
    </flag>
  </flags>
```

```
    </datasets>
  </productmeta>
  <datasection>
    <dataset id="0" datatype="unsigned char" b64size="37904" zsize="28427" uzsize="844451">...</dataset>
    <flagset id="0" datatype="unsigned char" flagversion="1.0" flagwordsize="24"
 b64size="148448" zsize="111335" uzsize="2533353">...</flagset>
  <datasection>
</radardata>
```

C.3.2 Polar volume

```
<?xml version="1.0" encoding="UTF-8"?>
<radardata version="1.0.2" time="20120504T080053Z" epoch="1336118453" type="ss" product="vol">
  <productmeta>
    <radarinfo sitename="Hegebostad" wmoid="000000">
      <site geoid="sphere" lat="58.188669" long="7.166100" datum="WGS84"/>
      <site geoid="WGS84" lat="58.360802" long="7.166100" datum="WGS84"/>
      <height>631.0</height>
      <radar_system>GEMA</radar_system>
      <software>Rainbow</software>
      <wavelength>0.053190</wavelength>
      <beamwidth>0.940000</beamwidth>
    </radarinfo>
    <area name="hgb"/>
    <datasets>
      <processing classification="0" clutter_filtered="0" vpr_corrected="0"/>
      <set id="0">
        <convparams alfa="0.500000" beta="-32.000000"/>
        <data datatype="dbz" min="-31.500" max="95.500" levels="255" nomeasure="0"/>
        <elevation angle="0.500" rangestep="0.250" stoprange="240.000" rangebins="960"/>
        <azimuth azispeed="8.000" anglestep="1.000" startangle="288.027" rays="360"/>
        <radarconstants noise_power_dbz="0.0000" radar_constant="0.0000"/>
        <time start="1336118408" end="1336118453"/>
        <address set="data">//datasection/dataset</address>
        <address set="startangles">//datasection/startangles</address>
        <address set="flags">//datasection/flagset</address>
      </set>
      <set id="1">
        <convparams alfa="0.500000" beta="-32.000000"/>
        <data datatype="dbz" min="-31.500" max="95.500" levels="255" nomeasure="0"/>
        <elevation angle="1.000" rangestep="0.250" stoprange="230.000" rangebins="920"/>
        <azimuth azispeed="8.000" anglestep="1.000" startangle="311.028" rays="360"/>
```

```
      <radarconstants noise_power_dbz="0.0000" radar_constant="0.0000"/>
      <time start="1336118456" end="1336118501"/>
      <address set="data">//datasection/dataset</address>
      <address set="startangles">//datasection/startangles</address>
      <address set="flags">//datasection/flagset</address>
    </set>
    <set id="2">
      <convparams alfa="0.500000" beta="-32.000000"/>
      <data datatype="dbz" min="-31.500" max="95.500" levels="255" nomeasure="0"/>
      <elevation angle="1.600" rangestep="0.250" stoprange="185.000" rangebins="740"/>
      <azimuth azispeed="12.000" anglestep="1.000" startangle="338.033" rays="360"/>
      <radarconstants noise_power_dbz="0.0000" radar_constant="0.0000"/>
      <time start="1336118503" end="1336118533"/>
      <address set="data">//datasection/dataset</address>
      <address set="startangles">//datasection/startangles</address>
      <address set="flags">//datasection/flagset</address>
    </set>
...
    <flags bits="24">
      <flag name="nodata" pos="0" size="1">
        <values val="1" descr="nodata"/>
      </flag>
      <flag name="lowele" pos="1" size="1">
        <values val="1" descr="lowele"/>
      </flag>
      <flag name="highele" pos="2" size="1">
        <values val="1" descr="highele"/>
      </flag>
      <flag name="blocked" pos="3" size="1">
        <values val="1" descr="blocked"/>
      </flag>
      <flag name="block_percent" pos="4" size="7">
```

```
          <values descr="5 percent steps"/>
        </flag>
        <flag name="seaclutter" pos="11" size="1">
          <values val="1" descr="clutter"/>
        </flag>
        <flag name="groundclutter" pos="12" size="1">
          <values val="1" descr="clutter"/>
        </flag>
        <flag name="otherclutter" pos="13" size="1">
          <values val="1" descr="clutter"/>
        </flag>
        <flag name="clutterprobability" pos="14" size="7">
          <values descr="1 percent steps"/>
        </flag>
        <flag name="classification" pos="21" size="2">
          <values val="0" descr="no data"/>
          <values val="1" descr="rain"/>
          <values val="2" descr="sleet"/>
          <values val="3" descr="snow"/>
        </flag>
      </flags>
    </datasets>
  </productmeta>
  <datasection>
    <dataset id="0" datatype="unsigned char" b64size="100952" zsize="75712" uzsize="345600">...</dataset>
    <startangles id="0" datatype="double" b64size="2008" zsize="1505" uzsize="2880">...</startangles>
    <flagset id="0" datatype="unsigned char" flagversion="1.0" flagwordsize="24"
b64size="1372" zsize="1028" uzsize="1036800">...</flagset>
...
  </datasection>
</radardata>
```

D Example code of a ConRad program

A minimalistic example on how to use ConRad to convert PRORAD XML→ MF-BUFR.

```
!! Copyright (C) 2010-2012, met.no.
!!
!! This file is part of CONRAD.
!!
!! CONRAD is free software: you can redistribute it and/or modify
!! it under the terms of the GNU General Public License as published by
!! the Free Software Foundation, either version 2 of the License, or
!! (at your option) any later version.
!!
!! CONRAD is distributed in the hope that it will be useful,
!! but WITHOUT ANY WARRANTY; without even the implied warranty of
!! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
!! GNU General Public License for more details.
!!
!! You should have received a copy of the GNU General Public License
!! along with CONRAD.  If not, see <http://www.gnu.org/licenses/>.
!>
!! @file minimalistic.F90
!! @author Martin S. Gronsleth <martinsg@met.no>
!! @date Tue May 15 14:32:07 2012
!! @brief Example code on how to  interact with the conrad library.
!<
program minimalistic
  use conrad_kinds
  use conrad_types,only:conrad_polar_meta
  use conrad_constants,only:CFALSE,CTRUE,CUNSET,&
       &NULL,CRITICAL,ERROR,WARN,INFO,INFO2
  use conrad,only:conrad_settings,conradInit,conradOpenWrite,&
       &conradWrite,conradCloseWrite,conradOpenRead,conradRead,conradCloseRead
  use conrad_conv,only: makeSameNumberOfRangeBins
  use conrad_statuscodes        ! for status code constants etc
  implicit none
  type(conrad_polar_meta) ::conrad_polar !< conrad product
  character(len=255)          ::infile   !< input filename
  character(len=255)          ::outfile  !< output filename
  integer                     ::iret              !< return value
  integer                     ::verbosity =  INFO2 !< verbositylevel
  character(len=20)           ::myname='minimalistic' !< program name
  type(conrad_settings)       ::conradSettings !< Settings for the
  infile =  "test.xml"
  outfile = "test.bfr"
  ! Setup conrad
  conradSettings%verbosity    = verbosity
  conradSettings%inputformat  = "PRORADXML"
  conradSettings%outputformat = "MFBUFR"
  conradSettings%writeBUFRsettings%legacy_wind_encoding = CFALSE
  call conradInit(conradSettings)

  ! Open, read and close input PRORAD XML file
  call conradOpenRead(infile,iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1

  call conradRead(conrad_polar,iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1

  call conradCloseRead(iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1

  ! Adapt to the MFBUFR format:
  call makeSameNumberOfRangeBins(conrad_polar,iret)
  if (iret /= STATUSOK) write(*,*) myname, trim(getStatusMsg(iret))
  if (iret > STATUSOK)  stop 1

  ! Open, write and close output PRORAD XML file
  call conradOpenWrite(outfile,iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1

  call conradWrite(conrad_polar,iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1

  call conradCloseWrite(iret)
  write(*,*) myname, trim(getStatusMsg(iret))
  if (iret /= STATUSOK) stop 1
end program minimalistic
```

minimalistic.F90