EUMETNET OPERA weather radar information model for implementation with the HDF5 file format

Version 2.2

Original authors:

Daniel B. Michelson[1], Rafał Lewandowski[2],
Maciej Szewczykowski[2], and Hans Beekhuis[3]

Additional author:

Günther Haase[1]

[1]Swedish Meteorological and Hydrological Institute, Norrköping, Sweden
[2]Institute of Meteorology and Water Management, Warsaw, Poland
[3]Royal Netherlands Meteorological Institute, De Bilt, Netherlands

on behalf of EUMETNET OPERA

March 21, 2014

Abstract

This document specifies an information model with which the encoding, decoding and management of data and products from weather radar systems may be facilitated, primarily for the purposes of international exchange in Europe. An implementation of this information model is also specified which makes use of the HDF5 file format developed and maintained by the HDF Group. The result manifests itself in the form of truly self-describing weather radar data files highly suitable for environments where data exchange between radars from different manufacturers, different organizations, and/or different countries is conducted. The ability to include quality information, in the forms of metadata and binary arrays, is included in a powerful and flexible manner. This information model constitutes an official second-generation European standard exchange format for weather radar datasets. Because the netCDF file format is built on HDF5, we also try to ensure that our information model will be compliant with netCDF.

RELEASE NOTES

Version 2.2, March 2014

Table 8 contains additional `how Attributes` for all objects while new quantities have been added to Table 16. Some attributes (like TXloss, RXloss, SQI, SNR, VRAD, and WRAD) were assumed to be horizontally-polarized only. For consistency and clarity attributes that exist in both H and V polarisations are denoted accordingly. The new ODIM version also contains a minimum specification for a vertical profile and a polar RHI representation. The latter resembles sector objects and scans. Table 2 includes a new file object "ELEV" (Elevational object) to accommodate this polar representation. The ELEV object may only contain the polar product `how/product` "RHI" as already specified in Table 14. Previously existing quantities have been marked for deprecation. Optional `Attributes` with ambiguous polarities have been removed where new attributes supercede them.

Version 2.1, 26 April 2011

The "simple array" type has been introduced and some attributes have been redefined to use it. The `/what/source` attribute has undergone revision. Several new optional `how` metadata attributes have been added. Section 7 has been reformulated to also identify prioritized optional `how` attributes in support of various applications. Clarifications have been added where ambiguities have been found, and errors have been corrected. Due to the scope of the changes introduced, this version has been incremented to the next minor version number.

Version 2.0.1, 21 September 2010

The composite object and product are now officially accepted as OPERA standard. A couple of clarifications have been made, without changing any of the contents of the information model itself. The UML annex has been broken out to be its own working document. To mark this, and that we've released an updated document, a minor-subversion has been introduced.

Version 2.0, 9 June 2009

A few inconsistancies between metadata in the tables and the diagrams in the UML representation have been identified and corrected. These fixes do not motivate a version number increment.

Version 2.0, 1 June 2009

This is the first version of ODIM_H5 to be officially accepted as an OPERA standard. Notwithstanding this status, the official parts are the definitions of polar scans and volumes, along with all associated metadata. All other objects restain their draft status, and will achieve official status in due course, subject to approval in OPERA.

Contents

List of Tables

1 Introduction and motivation

During OPERA's second incarnation, the goal of work package 2.1 "HDF5 exchange software development" was to formulate a second-generation information model for use with weather radar data and the Hierarchical Data Format version 5 (HDF5) file format. This information model has since been adopted and implemented in operational software both inside and outside OPERA.

This document presents an information model developed for use with weather radar data and products. Its implementation is also presented, which makes use of the HDF5 file format. HDF5 is developed and maintained by the HDF Group. All references to attributes, data, types, and so on, are in relation to those defined and used in the HDF5 documentation. The official HDF5 format documentation should be consulted for details on this format. This information model is an elaboration of the model presented by COST 717 and used in real-time operations in the Nordic countries[1]. Several enhancements have been made based on operational experience, and based on data quality issues addressed in OPERA II and the EU Voltaire project. The model presented here is not backwardly compatible with previous versions (version 1.2 and earlier), but the advantages of its new structure outweigh this disadvantage.

The information model given here is designed from the point of view of trying to harmonize all relevant information independently of the radar manufacturer and organization from which the data originates. While the information model is intended to enable the representation of the data and products agreed upon today within the framework of EUMETNET OPERA, we also look ahead to future needs and have tried to ensure that they are appropriately met with this information model. This means that the known products are supported, as are polarization diversity variables and virtually any quality-related information characterizing a given dataset. It is also vital to recognize the importance of being able to represent polar (spherical coordinate space) data, and this is accomodated as well in a flexible manner.

This information model has become the modern European standard, and as such can be used in forthcoming meteorological standard exchange mechanisms, ie. the WMO Information System (WIS) and its infrastructure. There is also an important link to OPERA's operational data centre (Odyssey), in that quality information in the information model is being used by Odyssey, thus supporting efforts to improve the quality of operational products covering the European continent.

In this way, weather radar data data and products are well-organized for data exchange infrastructure, and we have tried to ensure that the use of the modern technology will facilitate access to and use of European radar data both within and outside the meteorological community.

[1] Michelson D.B., Holleman I., Hohti H., and Salomonsen M., 2003: HDF5 information model and implementation for weather radar data. COST 717 working document WDF_02_200204_1. version 1.2.

2 Information model concept

The information model attempts to achieve a general-purpose model for storing both individual scans, images and products, while also allowing series (time and/or space) of these types of information to be stored using the same building-blocks. The heirarchical nature of HDF5 can be described as being similar to directories, files, and links on a hard-drive. Actual metadata are stored as so-called “attributes”, and these attributes are organized together in so-called “groups”. Binary data are stored as so-called “datasets”. The challenge in formulating an information model is in defining the way in which these general building-blocks are organized. This section illustrates the information model defined in detail later in this document, in an attempt to present and clarify its structure.

The first example is given in Figure 1, which shows how a simple cartesian product, for example a CAPPI, is represented. At the top level, the HDF5 file contains the so-called “root” group. This group is always there. Following that, three groups contain metadata; these are called “what” (object, information model version, and date/time information), “where” (geographical information), and “how” (quality and optional/recommended metadata). The data is organized in a group called “dataset1” which contains another group called “data1” where the actual binary data are found in “data”. This organization may seem overly complicated at first, but we will see shortly how powerful and necessary this organization is.

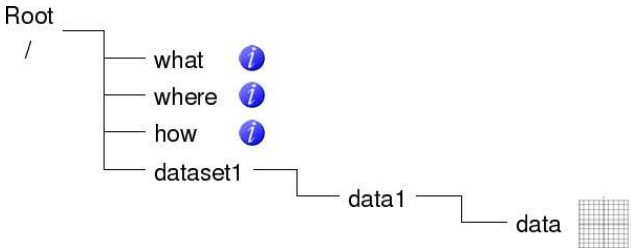

Figure 1: Cartesian product.

In terms of the analogy with a file system on a hard-disk, the HDF5 file containing this simple cartesian product is organized like this:

```
/
/what
/where
/how
/dataset1
/dataset1/data1
/dataset1/data1/data
```

In Figure 2, the exact same structure is used to represent a polar scan of data.

When we use this structure to represent a complete scan of data containing three parameters, the seemingly complicated organization used to store the binary data becomes easier to understand. Figure 3 shows how “dataset1” now contains three “data” groups, each containing a binary array of data. What we also see is the way in which the metadata groups “what” and “where” can be used recursively to hold metadata which are unique to their place in the hierarchy. In this case, “where” in “dataset1” contains the elevation angle of the scan used to collect the three parameters, and the three local “what” groups contain the information on which parameter is stored in each. In this way, “dataset1” can contain Z, V, and W in this case, but it can also contain an arbitrary number of other parameters.

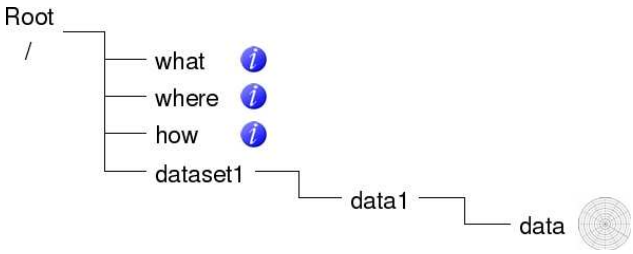

Figure 2: Simple polar scan.

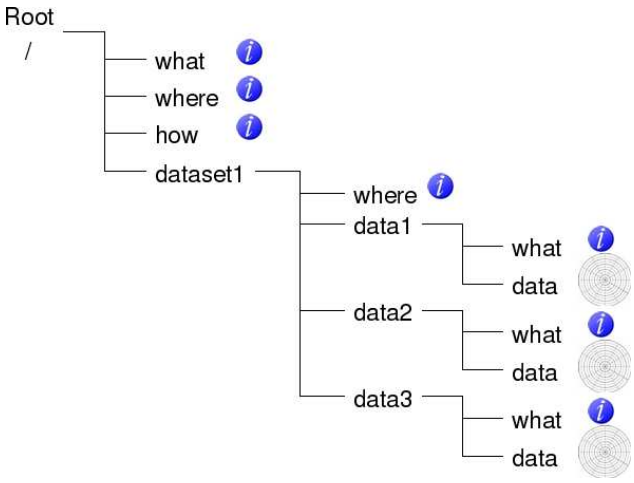

Figure 3: A complete polar scan containing three parameters.

The use of dataset-specific metadata groups is illustrated in Figure 4. Both methods of using “what”, “where” and “how” are acceptable. In the example on the left, the metadata groups contain attributes which are valid for every dataset in that group. In the case on the right, the metadata groups contain attributes which are valid only for that single dataset. In Figure 3, “where” is valid for all three datasets in the group, whereas “what” is only valid locally. The local metadata always have top priority, so if a mistake is made where a file contains the same metadata at different levels, the most local level will always take precedence.

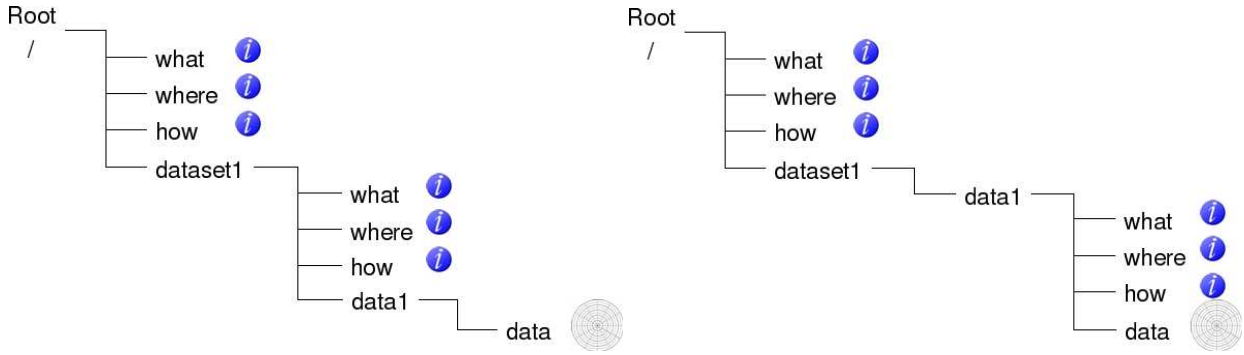

Figure 4: Use of dataset-specific metadata groups.

Continuing from the single scan in Figure 3, we can easily extend the same structure every time we want to add a new scan. This is illustrated in Figure 5. Here, the new elevation angle will be stored in `/dataset2/where`, and the information on the parameters stored in the datasets are found in each local

"what". Note that optional metadata can be added in a "how" group either directly under "dataset2" or together with each parameter, but this hasn't been included in this example. A complete volume containing an arbitrary number of scans, each containing an arbitrary number of parameters, is organized using this structure. A cartesian volume can be constructed in exactly the same way. Time series of polar or cartesian products can be constructed in the same way too.

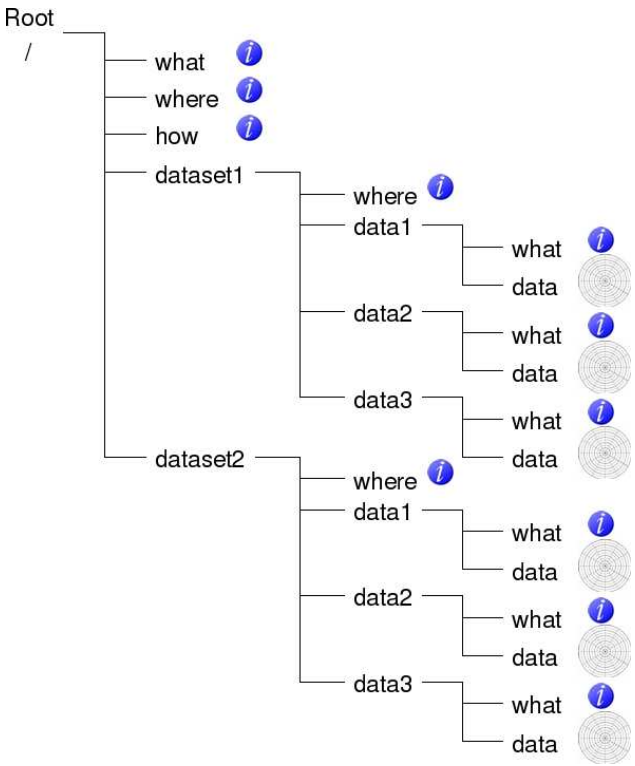

Figure 5: A polar volume containing two scans.

This information model uses the same logic to include the representation of **quality information**. Figure 6 illustrates how quality data can be added to polar data in a scan containing two parameters. In the same way that metadata can be applicable to all parameters in the scan, so too can quality information be representative either generally or locally. In this example, `/dataset1/quality1` can contain quality based on beam blockage since this is applicable to all parameters collected at the same elevation angle. However there may be different quality metrics which are applicable to each individual radar parameter, and these are stored locally. In this case, "data1" contains two such local quality metrics which are unique to that parameter, whereas "data2" only contains one. And we also see that each quality metric can be described using local metadata groups. The quality metrics should follow the guidelines set out in OPERA II[2].

Using this hierarchical structure, it becomes clear that we now have an information model capable of representing a volume containing an arbitrary number of scans/images, each of which can contain an arbitrary number of parameters which, in turn, can be characterized by an arbitrary number of quality metrics.

The examples provided in this discussion have focused on polar data, but they can also be applied to all the objects supported in this information model. These objects are polar scans, cartesian images, profiles, RHIs, cross-sections, side-panels, individual rays, sectors, and even embedded graphics images in an industrial

[2]Holleman I., Michelson D., Galli G., Germann U., and Peura M., 2006: Quality information for radars and radar data. OPERA II deliverable OPERA_2005_19.

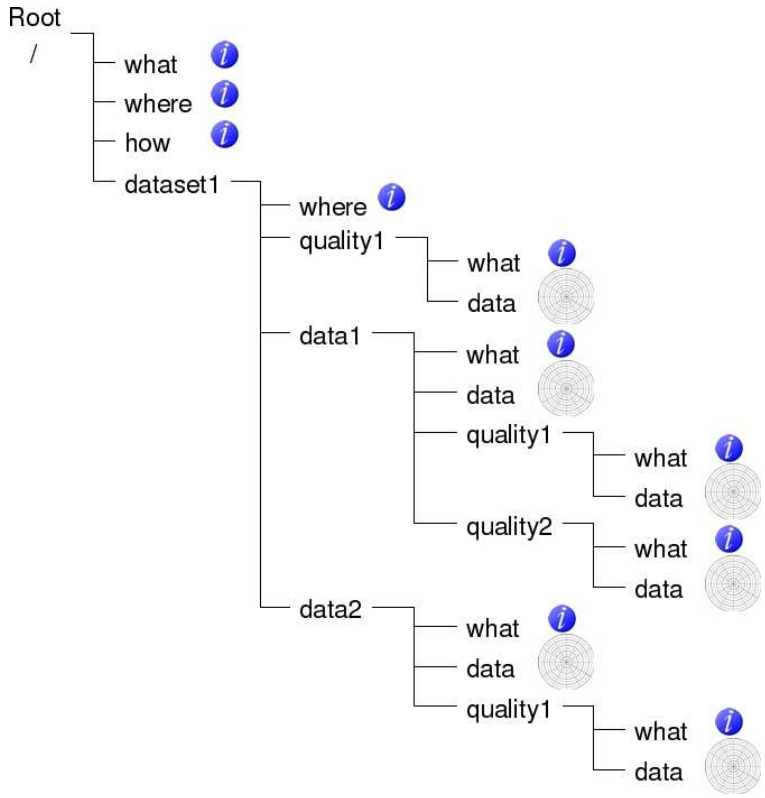

Figure 6: A polar scan containing two parameters and associated quality metrics.

graphics file format.

There are additional objects which are *complementary* to this information model and which are included since they are quite useful in various situations. One of them is the ability to add a color palette to an 8-bit dataset. There are standard mechanisms for this in the HDF5 library, and these are used without modification. Another object is a legend. This is a useful object when the data in a dataset is classified, discrete, or just level-sliced. The legend provides the ability to describe which quantitative values are represented by each qualitative value in the dataset, or which qualitative class is represented by a given value. The legend is used to tell the user that a certain value in the data represents e.g. “0.1–0.5 mm/h” in the case of a rainrate product, and e.g. “sleet” in the case of a hydrometeor classification. Both the palette and the legend objects are illustrated in Figure 7.

Finally, this information model enables the exchange of graphics representations of data. Using the same structure as that shown in Figure 1, Figure 8 illustrates this.

In the rest of this document, the detail specification of the information model is presented. With this specification, it shall be possible to read, write, and understand HDF5 files using this information model.

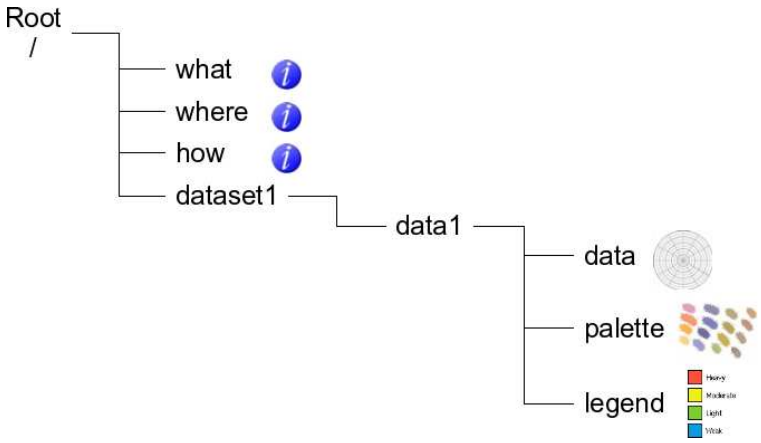

Figure 7: A polar scan containing a color palette and a legend.

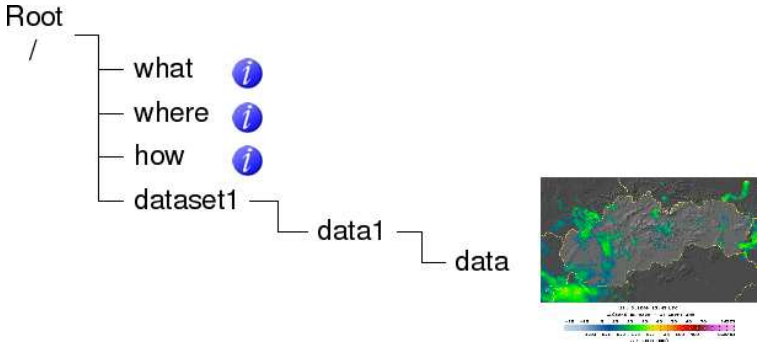

Figure 8: An embedded Slovak picture in an industrial graphics file format.

3 Definitions

HDF5 allows data to be stored as `Attribute`s, `Dataset`s, `Group`s, and user-defined `Compound` types. For use here, all types except user-defined `Compound` are permitted for use. Only `Compound` types defined in this document are permitted, in practise the optional legend object described in Section 6.2. In practice, all of these types are manipulated through the use of general-purpose `Node` objects, i.e., a `Node` may be any of the above mentioned types.

3.1 Scalars (integers, real values, and strings)

Scalar values are stored in `Attribute` objects and may be strings, integers, or real (floating-point) values. For the sake of consistency and simplicity, integer values shall be represented as 8-byte `long`, and real values shall be represented as `double`. These can be written to file as native types. They will be read and automatically returned as the corresponding native type on another platform. Endianness is therefore no cause for concern.

A clarification is however necessary; a `long` on one machine may have a different length than that on another machine. This is because a native `long` could, perhaps, be returned as an `int` in some cases. To prevent this, you should ensure that all integer scalar attributes are written with a length of 8 bytes. In some cases, this may require writing values of type `long long`, but it can also be enforced on POSIX-compliant systems by using the `int64_t` typedef.

Strings shall be encoded in ASCII or the ASCII representation of UTF-8. Strings should never be terminated by the application, because they shall be automatically null-terminated by the HDF5 library. This is done by setting the string termination to `STRPAD=H5T_STR_NULLTERM`. Other methods of termination available in the HDF5 library shall not be used. It is also necessary to specify the length of each string (`STRSIZE`). Even when `H5T_STR_NULLTERM` is used, the value of `STRSIZE` must be incremented manually by one to ensure that it includes the `NULL` character. Other methods of specifying the length of the string provided by the HDF5 library (ie. `H5T_VARIABLE`) shall not be used.

3.1.1 Booleans

A string is used to store truth value information. The string "`True`" is used to represent true, and "`False`" is used to represent false.

3.1.2 Sequences

A special kind of string may be used to store sequences. A sequence contains comma-separated scalar values in string notation. For example, a sequence is useful for storing the radar stations contributing to a composite image, and in storing the elevation angles used in a polar volume of data.

3.1.3 Simple arrays

Sometimes it is necessary to store numeric metadata whose geometry corresponds with the number of azimuth gates in a scan, or bins along a ray. The simple array provides this ability. Instead of using a complete dataset object provided by the HDF5 library (`H5D` dataset interface), the simple array uses the `H5S` dataspace interface. To simplify this further, only one-dimensional simple arrays are allowed, and to remain consistent with scalar integer and real value attributes, the simple arrays shall contain either "long" integers or "double" floats, as defined above.

3.2 Attributes

An `Attribute` is an HDF5 object used to store a header attribute or data. For our purposes, it is only used to store header attributes. In order to facilitate the management of so-called "atomic" attributes, ie. individual values, we use double precision for both integers (long) and floating-point (double) values. **Note** that the specification of strings is intended to be case sensitive.

3.3 Datasets

An HDF5 `Dataset` is a self-describing data object containing an *n*-dimensional binary array and attributes describing it. The array type may be any of `char`, `schar`, `uchar`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `llong`, `ullong`, `float`, `double` on a given platform.

In this document the text formatting of `Dataset` (in `Courier` font) means an HDF5 dataset, whereas the text formatting of **dataset** (in **bold face**) refers to the binary data in that dataset.

3.4 Groups

A `Group` is a top-level object which is used to organize other objects. For example, a `Group` may contain a collection of header `Attributes`, a collection of `Datasets`, or be the root object for the complete contents of an HDF5 file.

4 Metadata attribute specification

Header attributes are collected in three `Group` objects, all of which may be used recursively. `Attribute`s which describe a given file's contents and the time and place for which it represents are collected in a `Group` called `what`. `Attribute`s which describe a given file's geographical characteristics (projection, corner coordinates, dimensions, scales) are collected in a `Group` called `where`. The `what` and `where` `Group`s both contain mandatory `Attribute`s only. Those `Attribute`s which collectively describe additional data/product characteristics, such as radar system, chosen algorithm, and quality-related metadata, are stored in a `Group` called `how`. All `Attribute`s found in `how` may be considered optional, although the use of those given in this document is recommended. Additional `Attribute`s not specified in this document may only be stored in the `how` `Group`.

Top-level header attributes are collected in `what`, `where` and `how` `Group` objects located directly under the root `Group` '`/`'. Each attribute is stored as an `Attribute` object containing a scalar value. Some data/products may have `Dataset`-specific header attributes, in which case a the `Dataset` and its header `Attribute`s are collected in lower-level `what`, `where` and `how` `Group`s which are associated with that `Dataset`.

The concept used here is that each `Attribute` is identified with a string, the idea being that this is a more intuitive means of organizing data compared to numeric descriptors. It also means that a file may be queried using the *h5dump* utility to easily see and understand the contents of a file.

Mandatory header attributes for all files are given in Table 1. Note that all date and time information is for the **nominal time** of the data, ie. the time for which the data are valid. (The nominal time is not the exact acquisition time which is found elsewhere in the file.)

Note that all geographical longitude/latitude coordinates are specified with positive easting values east of the Greenwich meridian and positive northing values north of the equator.

4.1 Root `Group`

No HDF5 file can exist without this `Group`, since it is the starting point of the hierarchy. However, in order to take the first steps towards interoperatility with Climate and Forecast (CF) Conventions, we require an `Attribute` in the root `Group`. This `Attribute` is called "`Conventions`" and its value is a string containing the acronym of the information model being used, and its version, in a format which lends itself to a directory structure. This is done to comply with the documentation policy maintained by the community managing CF Conventions[3].

This information model is called the "OPERA Data Information Model for HDF5" which gives the acronym "ODIM_H5". Recognizing the history of this information model, we start at version 2.0. The resulting value for "`/Conventions`" is "`ODIM_H5/V2_0`", and this `Attribute` must be present in all ODIM_H5 files.

The present version is 2.2, which translates to "`ODIM_H5/V2_2`".

[3] http://cfconventions.org/

4.2 Top-level `what Group`

In this section the content of the top-level `what` is described.

This `Group` contains mandatory `Attribute`s only which collectively describe a given file's contents and time of acquisition. These `Attribute`s are given in Tables 1-14.

Table 1: Mandatory top-level `what` header `Attribute`s for all weather radar files.

Name	Type	Format	Description
object	string	-	According to Table 2
version	string	H5rad M.m	Format or information model version. "M" is the major version. "m" is the minor version. Software is encouraged to warn if it receives a file stored in a version which is different from the one it expects. The software should, however, proceed to read the file, ignoring `Attribute`s it does not understand.
date	string	YYYYMMDD	Nominal Year, Month, and Day of the data/product
time	string	HHmmss	Nominal Hour, Minute, and Second, in UTC of the data/product
source	string	TYP:VALUE	Variable-length string containing pairs of identifier types and their values, separated by a colon. Several pairs can be concatenated, separated by commas, in the form TYP:VALUE,TYP:VALUE, etc. Types and their values are given in Table 3. There are several identifiers that can be used to identify a given radar. Not all radars have been assigned all types, so it is hoped that at least one of these can be used for each radar. It recommended to use as many identifiers as have been assigned to a given radar.

Object strings may be any of those given in Table 2. For each `Dataset`, there may be an accompanying `what Group` with information specific to that `Dataset`, according to Table 13.

Table 2: File object strings and their meanings

String	Description
PVOL	Polar volume
CVOL	Cartesian volume
SCAN	Polar scan
RAY	Single polar ray
AZIM	Azimuthal object
ELEV	Elevational object
IMAGE	2-D cartesian image
COMP	Cartesian composite image(s)
XSEC	2-D vertical cross section(s)
VP	1-D vertical profile
PIC	Embedded graphical image

Note that a file containing a single scan of polar data may not be represented using object type PVOL; the

object type in these cases must be SCAN. **Note** also that an RHI polar volume can be represented using a number of `Dataset`s and the ELEV object.

Table 3: Source type identifiers and their associated values. It is mandatory for at least one of WMO, RAD, NOD, ORG, or CTY to be present.

Identifier	Description	Example
WMO	Combined WMO block and station number in the form A_1b_wnnnnn, or 0 if none assigned. The first two digits represent the block number, where the first digit A_1 is the regional association area and the second digit b_w is the sub-area. Remaining digits are the station number. (According to the WMO, numbers in the form A_1b_wnnn are considered equivalent to the form A_1b_w00nnn). One mandatory way to identify single-site data. References: 1, 2.	WMO:02954
RAD	Radar site as indexed in the OPERA database. One mandatory way to identify single-site data.	RAD:FI44
PLC	Place according to the left column of Table 9 of this document	PLC:Anjalankoski
NOD	Node according to the right column of Table 9 of this document. One mandatory way to identify single-site data.	NOD:fiang
ORG	Originating centre. One mandatory way to identify composites.	ORG:247
CTY	Country according to BUFR tables 14 0 1 101	CTY:613
CMT	Comment: allowing for a variable-length string	CMT:Suomi tutka

4.3 `where` Group

In this section the `Attribute`s for the `where` `Group` are described. These are different for polar or cartesian `Dataset`s, i.e. containing `Dataset` and `Dataset` `Group`s respectively.

Note that the use of the `where` `Group` is mandatory but that its placement will be at the top level of a given file, and at a lower level associated with a given `Dataset`. This is because some attributes are valid globally (e.g. the coordinates of a radar) whereas others are local (e.g. the elevation angle used for a given sweep).

4.3.1 `where` for polar data `Group`s

This `where` `Group` contains mandatory `Attribute`s only which collectively describe geographical and geometrical characteristics of a given `Dataset` **dataset**. **Note** that the **dataset** itself is the object containing the binary data and that `where` in this context describes that **dataset** but is located in the corresponding `Dataset` which contains all these objects. Section 2 illustrates how these two objects are related to each other.

Polar data, i.e. raw radar data as a function of azimuth and range, are stored in a `Dataset` `Group`, and polar volume data are stored as a stack of these `Dataset` `Group`s. Each `Dataset` `Group` contains azimuthal data from a single elevation.

Table 4: `where` `Attributes` for polar data objects.

Name	Type	Description
lon	double	Longitude position of the radar antenna (degrees), normalized to the WGS-84 reference ellipsoid and datum. Fractions of a degree are given in decimal notation.
lat	double	Latitude position of the radar antenna (degrees), normalized to the WGS-84 reference ellipsoid and datum. Fractions of a degree are given in decimal notation.
height	double	Height of the centre of the antenna in meters above sea level.
`Dataset` specific		
elangle	double	Antenna elevation angle (degrees) above the horizon.
nbins	long	Number of range bins in each ray
rstart	double	The range (km) of the start of the first range bin
rscale	double	The distance in meters between two successive range bins
nrays	long	Number of azimuth or elevation gates (rays) in the object
a1gate	long	Index of the first azimuth gate radiated in the scan
Sector specific		
startaz	double	The azimuth angle of the start of the first gate in the sector (degrees)
stopaz	double	The azimuth angle of the end of the last gate in the sector (degrees)

4.3.2 `where` for geographically referenced image `Groups`

This `where` `Group` contains mandatory `Attributes` only which collectively describe a given `Dataset`'s geographical and geometrical characteristics. **Note** that the **dataset** is the object containing the binary data and that `where` in this context describes that **dataset** but is located in the corresponding `Dataset` which contains all these objects. Section 2 illustrates how these two objects are related to each other.

The PROJ.4 cartographic projections library[4] is a comprehensive means of managing geographically referenced information which has become a *de facto* standard. PROJ.4 is being used increasingly throughout Europe and the world. As a result, the most straightforward way of representing projection information in radar files is by means of the projection definition string which is used with the library itself. For example, the arguments used with PROJ.4 to define the Google Maps projection are `+proj=merc +lat_ts=0 +lon_0=0 +k=1.0 +R=6378137.0 +nadgrids=@null +no_defs` so this is what should be found as an `Attribute` in the `where` `Group` associated with the `Dataset` used to store the data geolocated using this projection. Similarly, the standard "longitude, latitude" projection, also known as EPSG 4326, is defined as `+proj=latlong +ellps=WGS84 +datum=WGS84 +no_defs`.

Note that PROJ.4 contains a complete set of arguments for specifying a given projection, including false easting/northing, rescaling of coordinates, choice of ellipsoid (or defining your own), choice of geodetic datum, and defining oblique (rotated) projections.

[4] originally from the United States Geological Survey, now from MapTools

Table 5: `where` `Attributes` for geographical image data `Groups`

Name	Type	Description
projdef	string	The projection definition arguments, described above, which can be used with PROJ.4. See the PROJ.4 documentation for usage. Longitude/Latitude coordinates are normalized to the WGS-84 ellipsoid and geodetic datum.
xsize	long	Number of pixels in the X dimension
ysize	long	Number of pixels in the Y dimension
xscale	double	Pizel size in the X dimension, in projection-specific coordinates (often meters)
yscale	double	Pixel size in the Y dimension, in projection-specific coordinates (often meters)
LL_lon	double	Longitude of the lower left pixel corner
LL_lat	double	Latitude of the lower left pixel corner
UL_lon	double	Longitude of the upper left pixel corner
UL_lat	double	Latitude of the upper left pixel corner
UR_lon	double	Longitude of the upper right pixel corner
UR_lat	double	Latitude of the upper right pixel corner
LR_lon	double	Longitude of the lower right pixel corner
LR_lat	double	Latitude of the lower right pixel corner

Note that the corner coordinates given must be for the actual corners of the respective pixels.

4.3.3 `where` for cross-section data `Group`

RHI and cross-sections are treated as a special form of cartesian image. The x-dimension of the image represents the coordinate in the x/y-plane, while the y-dimension describes the vertical coordinate of the RHI or cross-section. To describe the geographical orientation and extend of a RHI or cross-section a dedicated set of `Attributes` in the `where` `Group` has been defined. The geographical location of cross-sections is just given by longitudes and latitudes of start and stop positions. The cross-sections are thus assumed to be taken along great-circles. In case they are taken along a line in a plane of a geographical projection, the deviation from the great-circle will be negligible for visualization purposes.

Table 6: `where` `Attributes` for cross-section data.

Name	Type	Description
Common attributes		
xsize	long	Number of pixels in the horizontal dimension
ysize	long	Number of pixels in the vertical dimension
xscale	double	Horizontal resolution in m
yscale	double	Vertical resolution in m
minheight	double	Minimum height in meters above mean sea level
maxheight	double	Maximum height in meters above mean sea level
RHI specific		

continued on next page

continued from previous = page

Name	Type	Description
lon	double	Longitude position of the radar antenna (degrees). Fractions of a degree are given in decimal notation.
lat	double	Latitude position of the radar antenna (degrees). Fractions of a degree are given in decimal notation.
az_angle	double	Azimuth angle
angles	simple array of doubles	Elevation angles, in degrees, in the order of acquisition.
range	double	Maximum range in km
Cross section and side panel specific		
start_lon	double	Start position's longitude
start_lat	double	Start position's latitude
stop_lon	double	Stop position's longitude
stop_lat	double	Stop position's latitude

4.3.4 `where` for vertical profiles

This `where` `Group` contains mandatory `Attribute`s only which collectively describe the geographical and geometrical characteristics of vertical profiles of horizontal winds and/or radar reflectivity.

Table 7: `where` `Attribute`s for vertical profiles.

Name	Type	Description
lon	double	Longitude position of the radar antenna (degrees). Fractions of a degree are given in decimal notation.
lat	double	Latitude position of the radar antenna (degrees). Fractions of a degree are given in decimal notation.
height	double	Height of the centre of the antenna in meters above sea level.
levels	long	Number of points in the profile
interval	double	Vertical distance (m) between height intervals, or 0.0 if variable
minheight	double	Minimum height in meters above mean sea level
maxheight	double	Maximum height in meters above mean sea level

4.4 `how` `Group`

This `Group` contains recommended and other `Attribute`s which provide additional and complimentary information which can be used to describe a given `Dataset` object, for example information related to an object's quality. Attributes in `how` can be added, but they won't offially constitute the OPERA standard until they are added to the tables in this information model. **Note** that the use of the `how` `Group` is optional and that its placement may be either at the top level of a given file, or at a lower level associated with a given `Dataset`. If `how` is found at the top level, then it is assumed that its contents apply to all `Dataset`s in the file. If `how` is found at a lower level, then it must be located in the `Dataset` `Group` to which its contents apply. If `how` exists at both the top and lower levels, then the contents of the local `how` override the contents of the top level `how`.

Note as well that there can often be cases where some `Attribute`s apply for all `Dataset`s in a file and others may be different for each `Dataset`. In such cases, the `Attribute`s which apply to all `Dataset`s may be held in a top level `how` `Group` and those that change may be held in local `how` `Group`s.

For clarity, Table 8 containing recommended `how` `Attribute`s is partitioned such that different partitions contain different product-specific `Attribute`s.

Table 8: `how` `Attribute`s for all objects.

Name	Type	Description
General		
task	string	Name of the acquisition task or product generator
task_args	string	Task arguments
startepochs	double	Seconds after a standard 1970 epoch for which the starting time of the data/product is valid. A compliment to "date" and "time" in Table 1, for those who prefer to calculate times directly in epoch seconds.
endepochs	double	Seconds after a standard 1970 epoch for which the ending time of the data/product is valid. A compliment to "date" and "time" in Table 1, for those who prefer to calculate times directly in epoch seconds.
system	string	According to Table 10
TXtype	string	Transmitter type [magnetron; klystron; solid state]
poltype	string	Polarization type of the radar [single; simultaneous-dual; switched-dual]
polmode	string	Current polarity mode [LDR-H; single-H; LDR-V; single-V; simultaneous-dual; switched-dual]
software	string	According to Table 11
sw_version	string	Software version in string format, e.g. "5.1" or "8.11.6.2"
zr_a	double	Z-R constant A in $Z = A\ R^b$, applicable to any product containing reflectivity or precipitation data
zr_b	double	Z-R exponent b in $Z = A\ R^b$, applicable to any product containing reflectivity or precipitation data
kr_a	double	K_{dp}-R constant A in $R = A\ {K_{dp}}^b$
kr_b	double	K_{dp}-R exponent b in $R = A\ {K_{dp}}^b$
simulated	boolean	"True" if data are simulated, otherwise "False"
Data from individual radars		
beamwidth	double	The radar's half-power beamwidth (degrees)
wavelength	double	Wavelength in cm
rpm	double	The antenna speed in revolutions per minute, positive for clockwise scanning, negative for counter-clockwise scanning
elevspeed	double	Antenna elevation speed (RHI mode) in degrees/s, positive values ascending, negative values descending
pulsewidth	double	Pulsewidth in μs
RXbandwidth	double	Bandwidth (MHz) that the receiver is set to when operating the radar with the above mentioned pulsewidth
lowprf	double	Low pulse repetition frequency in Hz
midprf	double	Intermediate pulse repetition frequency in Hz
highprf	double	High pulse repetition frequency in Hz

continued on next page

continued from previous page

Name	Type	Description
TXlossH	double	Total loss in dB in the transmission chain for horizontally-polarized signals, defined as the losses that occur between the calibration reference plane and the feed horn, inclusive.
TXlossV	double	Total loss in dB in the transmission chain for vertically-polarized signals, defined as the losses that occur between the calibration reference plane and the feed horn, inclusive.
injectlossH	double	Total loss in dB between the calibration reference plane and the test signal generator for horizontally-polarized signals.
injectlossV	double	Total loss in dB between the calibration reference plane and the test signal generator for vertically-polarized signals.
RXlossH	double	Total loss in dB in the receiving chain for horizontally-polarized signals, defined as the losses that occur between the feed horn and the receiver, inclusive.
RXlossV	double	Total loss in dB in the receiving chain for vertically-polarized signals, defined as the losses that occur between the feed horn and the receiver, inclusive.
radomelossH	double	One-way dry radome loss in dB for horizontally-polarized signals
radomelossV	double	One-way dry radome loss in dB for vertically-polarized signals
antgainH	double	Antenna gain in dB for horizontally-polarized signals
antgainV	double	Antenna gain in dB for vertically-polarized signals.
beamwH	double	Horizontal half-power (-3 dB) beamwidth in degrees
beamwV	double	Vertical half-power (-3 dB) beamwidth in degrees
gasattn	double	Gaseous specific attenuation in dB/km assumed by the radar processor (zero if no gaseous attenuation is assumed)
radconstH	double	Radar constant in dB for the horizontal channel. For the precise definition, see Appendix A
radconstV	double	Radar constant in dB for the vertical channel. For the precise definition, see Appendix A
nomTXpower	double	Nominal transmitted peak power in kW at the output of the transmitter (magnetron/klystron output flange)
TXpower	simple array of doubles	Transmitted peak power in kW at the calibration reference plane. The values given are average powers over all transmitted pulses in each azimuth gate. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
powerdiff	double	Power difference between transmitted horizontally and vertically-polarized signals in dB at the the feed horn.
phasediff	double	Phase difference in degrees between transmitted horizontally and vertically-polarized signals as determined from the first valid range bins
NI	double	Unambiguous velocity (Nyquist) interval (±m/s)
Vsamples	long	Number of samples used for radial velocity measurements
Polar data		
scan_index	long	Which scan this is in the temporal sequence (starting with 1) of the total number of scans comprising the volume.
scan_count	long	The total number of scans comprising the volume

continued on next page

continued from previous page

Name	Type	Description
astart	double	Azimuthal offset in degrees (°) from 0° of the start of the first ray in the sweep. This value is positive where the gate starts clockwise after 0°, and it will be negative if it starts before 0°. In either case, the value must be no larger than half a ray's width.
azmethod	string	How raw data in azimuth are processed to arrive at the given value, according to Table 12
elmethod	string	How raw data in elevation are processed to arrive at the given value, according to Table 12
binmethod	string	How raw data in range are processed to arrive at the given value, according to Table 12
elangles	simple array of doubles	Elevation angles (degrees above the horizon) used for each azimuth gate in an "intelligent" scan that e.g. follows the horizon. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
startazA	simple array of doubles	Azimuthal start angles (degrees) used for each azimuth gate in a scan. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
stopazA	simple array of doubles	Azimuthal stop angles (degrees) used for each azimuth gate in a scan. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
startazT	simple array of doubles	Acquisition start times for each azimuth gate in the sector or scan, in seconds past the 1970 epoch. The number of values in this array corresponds with the value of `where/nrays` for that dataset. The required precision is to the millisecond.
stopazT	simple array of doubles	Acquisition stop times for each azimuth gate in the sector or scan, in seconds past the 1970 epoch. The number of values in this array corresponds with the value of `where/nrays` for that dataset. The required precision is to the millisecond.
startelA	simple array of doubles	Elevational start angles (degrees) used for each elevation gate in a scan. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
stopelA	simple array of doubles	Elevational stop angles (degrees) used for each elevation gate in a scan. The number of values in this array corresponds with the value of `where/nrays` for that dataset.
startelT	simple array of doubles	Acquisition start times for each elevation gate in the sector or scan, in seconds past the 1970 epoch. The number of values in this array corresponds with the value of `where/nrays` for that dataset. The required precision is to the millisecond.
stopelT	simple array of doubles	Acquisition stop times for each elevation gate in the sector or scan, in seconds past the 1970 epoch. The number of values in this array corresponds with the value of `where/nrays` for that dataset. The required precision is to the millisecond.
Cartesian images including composites		
angles	simple array of doubles	Elevation angles in the order in which they were acquired, used to generate the product

continued on next page

continued from previous page

Name	Type	Description
arotation	simple array of doubles	Antenna rotation speed. The number of values in this array corresponds with the value of `how/angles` described above.
camethod	string	How cartesian data are processed, according to Table 12
nodes	sequence	Radar nodes (Table 9) which have contributed data to the composite, e.g. "'searl', 'noosl', 'sease', 'fikor"'
ACCnum	long	Number of images used in precipitation accumulation
Vertical profile specific		
minrange	double	Minimum range at which data is used when generating profile (km)
maxrange	double	Maximum range at which data is used when generating profile (km)
dealiased	boolean	"True" if data has been dealiased, "False" if not
Quality		
pointaccEL	double	Antenna pointing accuracy in elevation (degrees)
pointaccAZ	double	Antenna pointing accuracy in azimuth (degrees)
anglesync	string	Antenna angle synchronization mode [azimuth; elevation]
anglesyncRes	double	Resolution of angle synchronization in degrees
malfunc	boolean	Radar malfunction indicator, "True" if malfunction, otherwise "False"
radar_msg	string	Radar malfunction message
radhoriz	double	Radar horizon (maximum range in km)
NEZH	double	The total system noise expressed as the horizontally-polarized reflectivity (dBZ) it would represent at one km distance from the radar.
NEZV	double	The total system noise expressed as the vertically-polarized reflectivity (dBZ) it would represent at one km distance from the radar.
OUR	double	Overall uptime reliability (%)
Dclutter	sequence	Doppler clutter filters used when collecting data
clutterType	string	Description of clutter filter used in the signal processor
clutterMap	string	Filename of clutter map
zcalH	double	Calibration offset in dB for the horizontal channel
zcalV	double	Calibration offset in dB for the vertical channel
nsampleH	double	Noise sample in dB for the horizontal channel
nsampleV	double	Noise sample in dB for the vertical channel
comment	string	Free text description. Anecdotal quality information
SQI	double	Signal Quality Index threshold value
CSR	double	Clutter-to-signal ratio threshold value
LOG	double	Security distance above mean noise level (dB) threshold value
VPRCorr	boolean	"True" if vertical reflectivity profile correction has been applied, otherwise "False"
freeze	double	Freezing level (km) above sea level
min	double	Minimum value for continuous quality data
max	double	Maximum value for continuous quality data
step	double	Step value for continuous quality data
levels	long	Number of levels in discrete data legend
peakpwr	double	Peak power (kW)
avgpwr	double	Average power (W)
dynrange	double	Dynamic range (dB)
RAC	double	Range attenuation correction (dBm)

continued on next page

continued from previous page

Name	Type	Description
BBC	boolean	"True" if bright-band correction applied, otherwise "False"
PAC	double	Precipitation attenuation correction (dBm)
S2N	double	Signal-to-noise ratio (dB) threshold value
polarization	string	Type of polarization (H, V) transmitted by the radar

Table 9: Examples of radar "places" and their node designations. The first two letters in the "node" represent the country, following the Internet convention. The "place" name may use ASCII representations of UTF-8[6]. The "node" must only use ASCII. Radars not in this table can be added.

Place	Node
Hasvik	nohas
Andøya	noand
Røst	norst
Rissa	norsa
Bømlo	nobml
Hægebostad	nohgb
Oslo	noosl
Kiruna	sekir
Luleå	selul
Örnsköldsvik	seovi
Östersund	seosu
Hudiksvall	sehud
Leksand	selek
Arlanda	searl
Vilebo	sevil
Vara	sevar
Ase	sease
Karlskrona	sekkr
Ängelholm	seang
Korpo	fikor
Vantaa	fivan
Anjalankoski	fianj
Ikaalinen	fiika
Kuopio	fikuo
Vimpeli	fivim
Utajärvi	fiuta
Luosto	filuo
Bornholm	dkbor
Stevns	dkste
Sindal	dksin

continued on next page

[4]For example, the character ñ (UNICODE character U+00F1) is represented in binary UTF-8 as `\xf1` whereas its ASCII representation is `\xc3\xb1`.

continued from previous page

Place	Node
Rømø	dkrom
De Bilt	nldbl
Den Helder	nldhl
Tallinn	eetal
Sürgavere	eesur
Riga	lvrix
Legionowo	plleg
Poznan	plpoz
Gdansk	plgda
Swidwin	plswi
Rzeszow	plrze
Brzuchania	plbrz
Ramza	plram
Pastewnik	plpas

Table 10: Radar System abbreviations and their meanings. Radars not in this table can be added.

String	Meaning
GEMAxxx	Gematronik Meteor ACxxx Radar
EECxxx	EEC xxx Radar
ERICxxx	Ericsson xxx Radar
VAISxxx	Vaisala xxx Radar

Table 11: Processing Software abbreviations and their meanings. Systems not in this table can be added.

String	Meaning
BALTRAD	BALTRAD toolbox
CASTOR	Météo France's system
EDGE	EEC Edge
FROG	Gamic FROG, MURAN ...
IRIS	Vaisala Sigmet IRIS
METEOCELL	IRAM's system
NORDRAD	NORDRAD
RADARNET	UKMO's system
RAINBOW	Selex Gematronik Rainbow

Table 12: Method abbreviations and their meanings.

String	Meaning
NEAREST	Nearest neighbour or closest radar
INTERPOL	Interpolation

continued on next page

continued from previous page

String	Meaning
AVERAGE	Average of all values
RANDOM	Random
MDE	Minimum distance to earth
LATEST	Most recent radar
MAXIMUM	Maximum value
DOMAIN	User-defined compositing
VAD	Velocity azimuth display
VVP	Volume velocity processing
RGA	Gauge-adjustment

4.5 `what` `Group` for `Dataset` objects

In this section the content of the `what` `Group` to be used with each `Dataset` is described. Note that the linear transformation coefficients “gain” and “offset” should be set to 1 and 0 respectively if they are not intended for use with a given `Dataset`.

Table 13: `Dataset`-specific `what` header `Attributes`.

Name	Type	Format	Description
product	string	-	According to Table 14
prodpar	Tab. 15	-	According to Table 15 for products. Only used for cartesian products.
quantity	string	-	According to Table 16
startdate	string	Starting YYYYMMDD	Year, Month, and Day for the product
starttime	string	Starting HHmmss	Hour, Minute, and Second for the product
enddate	string	Ending YYYYMMDD	Year, Month, and Day for the product
endtime	string	Ending HHmmss	Hour, Minute, and Second for the product
gain	double	-	Coefficient ’a’ in y=ax+b used to convert to unit. Default value is 1.0.
offset	double	-	Coefficient ’b’ in y=ax+b used to convert to unit. Default value is 0.0.
nodata	double	-	Raw value used to denote areas void of data (never radiated). **Note** that this `Attribute` is always a float even if the data in question is in another format.
undetect	double	-	Raw value used to denote areas below the measurement detection threshold (radiated but nothing detected). **Note** that this `Attribute` is always a float even if the data in question is in another format.

continued on next page

continued from previous page

String	Meaning

Table 14: Product abbreviations and their meanings.

String	Meaning
SCAN	A scan of polar data
PPI	Plan position indicator (Cartesian)
CAPPI	Constant altitude PPI
PCAPPI	Pseudo-CAPPI
ETOP	Echo top
MAX	Maximum
RR	Accumulation
VIL	Vertically integrated liquid water
SURF	Information valid at the Earth's surface
COMP	Composite
VP	Vertical profile
RHI	Range height indicator
XSEC	Arbitrary vertical slice
VSP	Vertical side panel
HSP	Horizontal side panel
RAY	Ray
AZIM	Azimuthal type product
QUAL	Quality metric

Table 15: Product parameters.

Product	Type	Product parameter
CAPPI	double	Layer height (meters above the radar)
PPI	double	Elevation angle used (degrees)
ETOP	double	Reflectivity level (dBZ)
RHI	double	Azimuth angle (degrees)
VIL	simple array of doubles	Bottom and top heights (m) of the integration layer

Table 16: Quantity (variable) identifiers. Radar moments are those received by the radar or derived thereof.

String	Quantity [Unit]	Description
TH	T_h [dBZ]	Logged horizontally-polarized total (uncorrected) reflectivity factor
TV	T_v [dBZ]	Logged vertically-polarized total (uncorrected) reflectivity factor
DBZH	Z_h [dBZ]	Logged horizontally-polarized (corrected) reflectivity factor
DBZV	Z_v [dBZ]	Logged vertically-polarized (corrected) reflectivity factor
ZDR	ZDR [dBZ]	Logged differential reflectivity
RHOHV	ρ_{hv} [0-1]	Correlation between Z_h and Z_v

continued on next page

continued from previous page

String	Quantity [Unit]	Description
LDR	L_{dr} [dB]	Linear depolarization ratio
PHIDP	ϕ_{dp} [degrees]	Differential phase
KDP	K_{dp} [degrees/km]	Specific differential phase
SQIH	SQI_h [0-1]	Signal quality index - horizontally-polarized
SQIV	SQI_v [0-1]	Signal quality index - vertically-polarized
SNRH	SNR_h [0-1]	Normalized signal-to-noise ratio - horizontally-polarized
SNRV	SNR_v [0-1]	Normalized signal-to-noise ratio - vertically-polarized
CCORH	CC_h [dB]	Clutter correction - horizontally-polarized
CCORV	CC_v [dB]	Clutter correction - vertically-polarized
RATE	RR [mm/h]	Rain rate
URATE	URR [mm/h]	Uncorrected rain rate
HI	HI [dB]	Hail intensity
HP	HP [%]	Hail probability
ACRR	$RR_{accum.}$ [mm]	Accumulated precipitation
HGHT	H [km]	Height (of echotops)
VIL	VIL [kg/m^2]	Vertical Integrated Liquid water
VRAD	V_{rad} [m/s]	Radial velocity. Marked for DEPRECATION.
VRADH	$V_{rad,h}$ [m/s]	Radial velocity - horizontally-polarized. Radial winds towards the radar are negative, while radial winds away from the radar are positive (PANT).
VRADV	$V_{rad,v}$ [m/s]	Radial velocity - vertically-polarized. Radial winds towards the radar are negative, while radial winds away from the radar are positive (PANT).
VRADDH	$V_{rad,d}$ [m/s]	Dealiased horizontally-polarized radial velocity
VRADDV	$V_{rad,d}$ [m/s]	Dealiased vertically-polarized radial velocity
WRAD	W_{rad} [m/s]	Spectral width of radial velocity. Marked for DEPRECATION.
WRADH	$W_{rad,h}$ [m/s]	Spectral width of radial velocity - horizontally-polarized
WRADV	$W_{rad,v}$ [m/s]	Spectral width of radial velocity - vertically-polarized
UWND	U [m/s]	Component of wind in x-direction
VWND	V [m/s]	Component of wind in y-direction
RSHR	SHR_r [m/s km]	Radial shear
ASHR	SHR_a [m/s km]	Azimuthal shear
CSHR	SHR_c [m/s km]	Range-azimuthal shear
ESHR	SHR_e [m/s km]	Elevation shear
OSHR	SHR_o [m/s km]	Range-elevation shear
HSHR	SHR_h [m/s km]	Horizontal shear
VSHR	SHR_v [m/s km]	Vertical shear
TSHR	SHR_t [m/s km]	Three-dimensional shear
BRDR	0 or 1	1 denotes a border where data from two or more radars meet in composites, otherwise 0
QIND	Quality [0-1]	Spatially analyzed quality indicator, according to OPERA II, normalized to between 0 (poorest quality) to 1 (best quality)
CLASS	Classification	Indicates that data are classified and that the classes are specified according to the associated legend object (Section 6.2) which must be present.

continued on next page

continued from previous page

String	Quantity [Unit]	Description
Vertical profile specific		
ff	[m/s]	Mean horizontal wind velocity
dd	[degrees]	Mean horizontal wind direction (degrees)
ff_dev	[m/s]	Velocity variability
dd_dev	[m/s]	Direction variability
n	–	Sample size
TH	[dBZ]	Logged horizontally-polarized total (uncorrected) reflectivity factor
TV	[dBZ]	Logged vertically-polarized total (uncorrected) reflectivity factor
DBZH	[dBZ]	Logged horizontally-polarized (corrected) reflectivity factor
DBZV	[dBZ]	Logged vertically-polarized (corrected) reflectivity factor
DBZH_dev	[dBZ]	Variability of logged horizontally-polarized (corrected) reflectivity factor
DBZV_dev	[dBZ]	Variability of logged vertically-polarized (corrected) reflectivity factor
w	[m/s]	Vertical velocity (positive upwards)
w_dev	[m/s]	Vertical velocity variability
div	$[s^{-1}]$	Divergence
div_dev	$[s^{-1}]$	Divergence variation
def	$[s^{-1}]$	Deformation
def_dev	$[s^{-1}]$	Deformation variation
ad	[degrees]	Axis of dialation (0-360)
ad_dev	[degrees]	Variability of axis of dialation (0-360)
rhohv	ρ_{hv} [0-1]	Correlation between Z_h and Z_v
rhohv_dev	ρ_{hv} [0-1]	ρ_{hv} variation
ZDR	ZDR [dBZ]	Logged differential reflectivity
LDR	L_{dr} [dB]	Linear depolarization ratio
PHIDP	ϕ_{dp} [degrees]	Differential phase
KDP	K_{dp} [degrees/km]	Specific differential phase

5 Data specification

All data arrays, for polar, cartesian, and profile data, are stored as **dataset** objects. Any of the types/depths stated in Sec. 3.3 are allowed. Any of compression levels 1 to 6 are recommended. (HDF5's built-in compression levels range from 0 (none) to 9 (maximum); levels above 6 tend to result in the algorithm using disproportionate resources relative to gains in file size, which is why their use is not encouraged.) HDF5 provides support for SZIP compression in addition to default ZLIB compression, but SZIP compression library is proprietary and will therefore not be supported in any official OPERA software.

All `Dataset Group`s are called `dataset`*n*, no matter which kind of data they hold. The character *n* represents the index of the `Dataset Group` in acquisition order in terms of elevation angle (polar data), and ascending order in terms of height (Cartsian products) starting at 1. A `Dataset Group` can hold an arbitrary number of binary **dataset**s. In the case of radar parameters (e.g. Z, V, W, etc.) the **dataset**s containing each parameter are each contained in a `Group` called `data`*n*, where *n* is the index of the **dataset** holding the binary data. This **dataset** is simply called `data`.

Within each `data`*n* `Group`, there may be an arbitrary number of quality indicators, each of which is held in a `Group` called `quality`*n*, where *n* is the index of the **dataset** holding the binary data. This **dataset** is simply called `data`. Because the quality indicator(s) describe a given quantity, they need not be assigned values of "nodata and" "undetect", as this information is already contained in the quantity being described. However, if one wishes to represent a quality indicator or index as a stand-alone product, then this should be done by representing it in its own **dataset** and assigning it the `QIND` quantity according to Table 16.

Each `Dataset Group` can store local `what`, `where` and `how` `Group`s to store metadata which are unique to that `Dataset`.

Note that all data with 8-bit unsigned depth (`uchar`) shall be represented as an HDF5 Image (H5IM) (Table 17). This applies to any 2-D `Dataset`, be it polar, cartesian, sector, or cross-section. This is a `Dataset` with a few added `Attribute`s which facilitate the Image's management by third-party software.

Table 17: Eight-bit Image attributes. Note that these are part of a `Dataset` object.

Attribute	Type	Description
CLASS	string	Should be "IMAGE"
IMAGE_VERSION	string	Should be "1.2", the current version number

There are several optional `Attribute`s describing an HDF5 Image, including Palette information (see Section 6.1).

A schematic of this organization is found in Section 2. What follows is details pertaining to each type of data that may be represented as binary data.

5.1 Polar Data

If possible, the start of the first azimuth gate (the first pulse) always points due north and the first range bin is that starting at the radar. This azimuthal precision may not be achievable in practice, in which case

the first azimuth gate is the one in the sweep with a starting azimuthal angle closest to 0°. This applies to data representing a full sweep, but not sector scans. For full sweeps, this implies that the first azimuth gate covers the interval 0-1° assuming 360 azimuth gates per scan. Azimuth gates are ordered clockwise except for sector scans which can be clockwise or counter-clockwise. In other words, range bins are stored in the array's equivalent X-dimension and azimuth gates in the Y-dimension. There is no mechanism for specifying missing azimuth gates or range bins in SCAN objects. This means that partial scans/rays must be filled-in in order to complete them. Filled-in areas are identified by the "nodata" value specified in the corresponding `what Group`. Radiated areas with no echo are represented with the "undetect" value also in the corresponding `what Group`. Alternatively, collections of rays can be stored in the RAY object, and sectors can be stored in the AZIM object. The first azimuth gate in the scan does not have to be the first ray of data collected for that scan, but the `/dataset`*n*`/where/a1gate Attribute` contains the information on this, thereby allowing a temporal reconstruction of the data collection.

All full-sweep data must be sorted according to the above description: clockwise and starting from north, even if they haven't been acquired that way. Azimuthal overlap is not allowed. Rays whose starting azimuthal angle go beyond 360 degrees from `a1gate`, ie. start a second sweep or replicate previous rays, must be omitted. It is therefore the responsibility of each supplier to comply with this specification.

All parameters collected in a single scan of data are contained in one `Dataset Group` with the same index number, as seperate `data Dataset Group`s.

5.2 Image Data

An image product in this context refers to 2-D cartesian quantitative data and not a visual graphic product (PIC, see Section 5.6 below).

Binary arrays are stored as one long unpadded binary string starting in the upper-left corner and proceeding row by row (north to south), from left (west) to right (east). Areas void of the specified variable are flagged using the "nodata" value specified in the corresponding `what Group`. Radiated areas with no echo are represented with the "undetect" value also in the corresponding `what Group`.

5.3 RHIs, cross sections and side panels

RHI, cross sections and side panels are a special form of image. RHIs taken with the antenna pointing east start on the left and end on the right. If the antenna points west, then the RHI starts on the right and ends on the left. RHIs pointing exactly south or north always start on the left and end on the right. For cross sections, the left side of the image is the starting point and the right side is the finishing point, regardless of the antenna's azimuth angle. For side panels, the starting point is north for vertical panels and west for horizontal panels. As with other image files, the first pixel is the upper-left one and image content is ordered row-wise from top to bottom and left to right.

5.4 Profiles

In contrast to polar or cartesian data which use `Dataset`s to store 2-D arrays, profiles use several `Dataset`s to store 1-D arrays representing different variables along a given profile. One variable is stored

in each `Dataset`. Levels in the profile are ordered sequentially in ascending order. A profile is a `Group` containing several `Dataset`s, each of which stores a variable for that level. This profile formulation is not restricted to wind variables, but can easily accommodate reflectivity and other variables as well.

Each `Dataset` containing a parameter making up the the profile contains a quantity designator according to Table 16.

Note that all **dataset**s must be of equal length, in order to match the heights given in the **dataset** containing the quantity `HGHT`.

5.5 Rays and sectors

It is unlikely that individual rays or sectors of data will be exchanged operationally and internationally, but these objects are added for the sake of completeness and in cases where quality information can be efficiently represented using these objects. Also, there may be production chains where the radar transmits each ray individually, so the availability of this means of representation can support them.

An individual ray of data is stored as a **dataset** with the most proximate data first. Missing data along the ray must be assigned the “nodata” and/or “undetect” values.

A sector is similar to a scan with the difference being that the sector doesn’t cover the horizon completely. A sector is stored the same was as a scan, the only clarification being the object name (Table 2 or 14) and the metadata in

Alternatively, sectors can be be represented as a collection of rays. This might be handy if the rays are few and with variable starting and ending rays, and/or with different range bin spacing from one ray to the next.

5.6 Embedded graphical images

Image files in industrial graphics formats, e.g. PNG, JPEG, GIF, TIFF, etc., can be written directly into a **dataset** as is. When this file is then retrieved, all that is necessary is to read the **dataset** contents as is and write a new file containing them.

6 Optional objects

There are two kind of objects which may be included in the HDF5 files and which are considered optional in this information model. These are palettes which may be attached to a given **dataset** to "color" them intuitively, and discrete legends which may complement a given **dataset** with what can be considered quality-related information. Each of these objects is specified below.

6.1 Palettes

The HDF5 Image API contains functionality for associating palettes (or color tables) with 8-bit and 24-bit arrays. Since we have not defined 24-bit graphic images explicitely in this information model, the treatment of palettes will be limited to 8-bit **dataset**s. Most radar data exchanged in Europe today is 8-bit, so the ability to complement them with palettes can be considered an enhancement. This is particularly interesting since the standard HDF5 binary tool *h52gif* converts an 8-bit **dataset** to a GIF file with or without an associated palette. And the *hdfview* tool visualizes the data with the palette if one is available. The reader is referred to the HDF5 documentation for complete information.

In order for an 8-bit `Dataset` to work with a palette, it must be defined as an `Image`. This is done by adding a few attributes to the `Dataset`, and this process does not impact at all on the binary array data. In other words, applications that only read the data won't be affected. This means that those applications that wish to use the palette can do so and those that don't can do so without having to modify any routines. The palette is defined as a seperate `Dataset` and then the palette is linked to the binary array `Dataset`. This does not affect the binary data either. The result is a straight-forward and unobtrusive mechanism for adding color to data.

An example of how a palette complements an 8-bit dataset is given in Table 19.

6.2 Legends

The HDF5 library enables users to define and store so-called "compound" data type objects. Compound data type objects are heterogeneous elements - a multi-dimensional array of fields. Such arrays consist of fields that either represent atomic data types defined in the HDF5 API, or compound data types. In the latter case such construction is referred to as "nested compound type". The reader is referred to the HDF5 documentation for complete information; what follows here is summarized.

Compound data type objects are handled by the HDF5 library which provides functions for reading and writing data to and from a **dataset** of a compound type. With these functions, either the whole record (an instance of compond data type) can be read or written or particular fields can be accessed in both read and write mode.

The compound data type is designed for use mainly in C language applications. This is a direct consequence of the approach to memory organization and access, which is based on C structures. However, it is still possible to use compound data types in Java applications, as well as to map compound data types to XML node structures.

The compound data type is suitable for holding and processing legend data relating to discrete quality information. In most cases, such a legend would contain two elements: *key* and *value*. *Key* is an abbreviated

character code for a given level of quality parameter it describes. *Value* holds the actual parameter value in the form of a character string. This pair of fields can be represented by the following C structure:

```
struct legend
{
    char[] key;
    char[] value;
};
```

The structure describes a given discrete quality parameter which, for example, can be a data quality indicator depending on distance to the radar. Such quality indicator can be stratified into several discrete categories, represented by character strings of a given contstant length. Another example is a hydrometeor classification based on polarimetric variables (rain, snow, hail, clutter ...). In both case, the key field of the legend structure will hold the category string, while the corresponding value will be stored in value field, which is also an array of characters. The length of particular fields can be either given explicitly (e.g. described in dedicated quality parameters table) or can be determined at runtime.

Since HDF5 approaches data object in the way the C language does, it is crutial to meet C standards while designing and processing compound data structures. Because C does not support string objects (strings of characters of variable length), the string object can only be represented by an array of characters. In addtition, to provide the possibility to determine the length of a legend dataset at runtime, an instance of a legend struture object should have fixed and constant byte length. This can be achieved by using character arrays of constant length. For the purpose of legend dataset, it seems reasonable to use 64-byte array for the key field, and 32-byte array for value field. The space used by the actual content of each legend structure depends on the user. The only requirement is that character is null terminated (the last character in array is null character).

```
struct legend
{
    char[64] key;
    char[32] value;
};
```

Once the legend structure is defined, instances of the legend structure are stored in a dedicated **dataset**. When defining a **dataset** data type, `H5T_COMPOUND` type must be used.

Endnote on software compatibility and versions

The use of the netCDF file format has also gained momentum within the meteorological and climatological communities. Today netCDF is a high-level layer built upon HDF5. Starting with HDF5 version 1.8.0 and netCDF version 4, HDF5 is able to manage netCDF files. There are also Java tools with netCDF which enable reading and writing of some kinds of HDF5 files. Since we recognize the benefits of being able to manage HDF5 files with netCDF (and vice versa), we require that any implementation of the information model specified in this document must use HDF5 version 1.8.0 or later. By organizing data in HDF5 in the straight-forward way specified in this document, the simplicity should facilitate for netCDF to manage such attributes and datasets.

7 Mandatory and prioritized optional metadata per product

The purpose of this section is to clarify exactly which metadata attributes are mandatory for each type of product. So far, we have only mentioned that `what` and `where` `Group`s are mandatory, whereas `how` is optional/voluntary. Here, we are specific about which metadata **must always** be present for each kind of product. In order for this to be a reasonable level of ambition, only those metadata which are fundamentally necessary to be able to manage the product at all, ie. at a purely functional level, are mandatory.

When it comes to optional `how` attributes, this section identifies which of them are considered prioritized and therefore strongly recommended to include. However, we don't forget that all `how` attributes are recommended and should be included if they are available. In the presentations that follow, the recommended locations of these prioritized `how` attributes are included.

The method used to present the metadata is in the form of terse listings, using the file-system analogy shown in Section 2 on page 2. The exact format of each node in the HDF5 file is according to the tables referred to in this document.

Datasets are also included in the following tables, for clarity.

7.1 Polar volume

The following example represents a polar volume consisting of two scans, each of which contains two parameters. The polar geometry is the same in both scans, the only differences being the elevation angle and the first measured azimuth gate in each scan. Using different pulsewidths and/or antenna rotation speeds between scans would necessitate placing the `how` attributes in their respecitive **dataset**s instead of at the top level.

Table 18: Polar volume

Node	Type
`/`	Root Group
`/Conventions`	Attribute, Section 4.1
`/what`	Group
`/what/object`	Atttribute, Table 2
`/what/version`	Atttribute, Table 1
`/what/date`	Atttribute, Table 1
`/what/time`	Atttribute, Table 1
`/what/source`	Atttribute, Table 3
`/where`	Group
`/where/lon`	Attribute, Table 4
`/where/lat`	Attribute, Table 4
`/where/height`	Attribute, Table 4
`/how`	Group
`/how/beamwidth`	Attribute, Table 8
`/how/NEZH`	Attribute, Table 8
`/how/pulsewidth`	Attribute, Table 8
`/how/radconstH`	Attribute, Table 8

continued on next page

continued from previous page

Attribute name	Type
/how/radconstV	Attribute, Table 8
/how/NI	Attribute, Table 8
/how/rpm	Attribute, Table 8
/dataset1	Group
/dataset1/what	Group
/dataset1/what/product	Attribute, Table 14
/dataset1/what/startdate	Attribute, Table 13
/dataset1/what/starttime	Attribute, Table 13
/dataset1/what/enddate	Attribute, Table 13
/dataset1/what/endtime	Attribute, Table 13
/dataset1/where	Group
/dataset1/where/elangle	Attribute, Table 4
/dataset1/where/a1gate	Attribute, Table 4
/dataset1/where/nbins	Attribute, Table 4
/dataset1/where/rstart	Attribute, Table 4
/dataset1/where/rscale	Attribute, Table 4
/dataset1/where/nrays	Attribute, Table 4
/dataset1/data1	Group
/dataset1/data1/what	Group
/dataset1/data1/what/quantity	Attribute, Table 16
/dataset1/data1/what/gain	Attribute, Table 13
/dataset1/data1/what/offset	Attribute, Table 13
/dataset1/data1/what/nodata	Attribute, Table 13
/dataset1/data1/what/undetect	Attribute, Table 13
/dataset1/data1/data	Dataset
/dataset1/data1/data/CLASS	Attribute, Table 17
/dataset1/data1/data/IMAGE_VERSION	Attribute, Table 17
/dataset1/data2	Group
/dataset1/data2/what	Group
/dataset1/data2/what/quantity	Attribute, Table 16
/dataset1/data2/what/gain	Attribute, Table 13
/dataset1/data2/what/offset	Attribute, Table 13
/dataset1/data2/what/nodata	Attribute, Table 13
/dataset1/data2/what/undetect	Attribute, Table 13
/dataset1/data2/data	Dataset
/dataset1/data2/data/CLASS	Attribute, Table 17
/dataset1/data2/data/IMAGE_VERSION	Attribute, Table 17
/dataset2	Group
/dataset2/what	Group
/dataset2/what/product	Attribute, Table 14
/dataset2/what/startdate	Attribute, Table 13
/dataset2/what/starttime	Attribute, Table 13
/dataset2/what/enddate	Attribute, Table 13
/dataset2/what/endtime	Attribute, Table 13
/dataset2/where	Group

continued on next page

continued from previous page

Attribute name	Type
/dataset2/where/elangle	Attribute, Table 4
/dataset2/where/a1gate	Attribute, Table 4
/dataset2/where/nbins	Attribute, Table 4
/dataset2/where/rstart	Attribute, Table 4
/dataset2/where/rscale	Attribute, Table 4
/dataset2/where/nrays	Attribute, Table 4
/dataset2/data1	Group
/dataset2/data1/what	Group
/dataset2/data1/what/quantity	Attribute, Table 16
/dataset2/data1/what/gain	Attribute, Table 13
/dataset2/data1/what/offset	Attribute, Table 13
/dataset2/data1/what/nodata	Attribute, Table 13
/dataset2/data1/what/undetect	Attribute, Table 13
/dataset2/data1/data	Dataset
/dataset2/data1/data/CLASS	Attribute, Table 17
/dataset2/data1/data/IMAGE_VERSION	Attribute, Table 17
/dataset2/data2	Group
/dataset2/data2/what	Group
/dataset2/data2/what/quantity	Attribute, Table 16
/dataset2/data2/what/gain	Attribute, Table 13
/dataset2/data2/what/offset	Attribute, Table 13
/dataset2/data2/what/nodata	Attribute, Table 13
/dataset2/data2/what/undetect	Attribute, Table 13
/dataset2/data2/data	Dataset
/dataset2/data2/data/CLASS	Attribute, Table 17
/dataset2/data2/data/IMAGE_VERSION	Attribute, Table 17

7.2 Composite

The following example shows how a composite image is represented. It also contains the standard HDF5 mechanisms for including a palette, assuming the image payload is 8-bit, along with a quality indicator field describing the given composite product. The COMP object can use /dataset1/data1/what/quantity, if the object contains more than one parameter/quantity.

Table 19: Cartesian image with palette

Attribute name	Type
/	Root Group
/Conventions	Attribute, Section 4.1
/what	Group
/what/object	Atttribute, Table 2
/what/version	Atttribute, Table 1
/what/date	Atttribute, Table 1
/what/time	Atttribute, Table 1

continued on next page

continued from previous page

Attribute name	Type
/what/source	Atttribute, Table 3
/where	Group
/where/projdef	Attribute, Table 5
/where/xsize	Attribute, Table 5
/where/ysize	Attribute, Table 5
/where/xscale	Attribute, Table 5
/where/yscale	Attribute, Table 5
/where/LL_lon	Attribute, Table 5
/where/LL_lat	Attribute, Table 5
/where/UL_lon	Attribute, Table 5
/where/UL_lat	Attribute, Table 5
/where/UR_lon	Attribute, Table 5
/where/UR_lat	Attribute, Table 5
/where/LR_lon	Attribute, Table 5
/where/LR_lat	Attribute, Table 5
/how	Group
/how/nodes	Atttribute, Table 8
/dataset1	Group
/dataset1/what	Group
/dataset1/what/product	Attribute, Table 14
/dataset1/what/prodpar	Attribute, Table 15
/dataset1/what/quantity	Attribute, Table 16
/dataset1/what/startdate	Attribute, Table 13
/dataset1/what/starttime	Attribute, Table 13
/dataset1/what/enddate	Attribute, Table 13
/dataset1/what/endtime	Attribute, Table 13
/dataset1/what/gain	Attribute, Table 13
/dataset1/what/offset	Attribute, Table 13
/dataset1/what/nodata	Attribute, Table 13
/dataset1/what/undetect	Attribute, Table 13
/dataset1/data1	Group
/dataset1/data1/data	Dataset
/dataset1/data1/data/CLASS	Attribute, Table 17
/dataset1/data1/data/IMAGE_VERSION	Attribute, Table 17
/dataset1/data1/quality1	Group
/dataset1/data1/quality1/data	Dataset
/dataset1/data1/quality1/what	Group
/dataset1/data1/quality1/what/gain	Attribute, Table 13
/dataset1/data1/quality1/what/offset	Attribute, Table 13
/dataset1/data1/quality1/how/task	Atttribute, Table 8
H5IM optional attributes	
/dataset1/data1/data/PALETTE	Link to /dataset1/data1/palette
/dataset1/data1/data/IMAGE_SUBCLASS	Attribute
/dataset1/data1/palette	Dataset
/dataset1/data1/palette/CLASS	Attribute

continued on next page

continued from previous page

Attribute name	Type
/dataset1/data1/palette/PAL_VERSION	Attribute

7.3 Vertical profile

The following example shows how a vertical profile is represented.

Table 20: Vertical profile

Attribute name	Type
/	Root Group
/Conventions	Attribute, Section 4.1
/what	Group
/what/object	Atttribute, Table 2
/what/version	Atttribute, Table 1
/what/date	Atttribute, Table 1
/what/time	Atttribute, Table 1
/what/source	Atttribute, Table 3
/where	Group
/where/lon	Attribute, Table 7
/where/lat	Attribute, Table 7
/where/height	Attribute, Table 7
/where/levels	Attribute, Table 7
/where/interval	Attribute, Table 7
/where/minheight	Attribute, Table 7
/where/maxheight	Attribute, Table 7
/dataset1	Group
/dataset1/what	Group
/dataset1/what/product	Attribute, Table 14
/dataset1/what/startdate	Attribute, Table 13
/dataset1/what/starttime	Attribute, Table 13
/dataset1/what/enddate	Attribute, Table 13
/dataset1/what/endtime	Attribute, Table 13
/dataset1/data1	Group
/dataset1/data1/what	Group
/dataset1/data1/what/quantity	Attribute, Table 16
/dataset1/data1/what/gain	Attribute, Table 13
/dataset1/data1/what/offset	Attribute, Table 13
/dataset1/data1/what/nodata	Attribute, Table 13
/dataset1/data1/what/undetect	Attribute, Table 13
/dataset1/data1/data	Dataset
/dataset1/data2	Group
/dataset1/data2/what	Group
/dataset1/data2/what/quantity	Attribute, Table 16
/dataset1/data2/what/gain	Attribute, Table 13

continued on next page

continued from previous page

Attribute name	Type
/dataset1/data2/what/offset	Attribute, Table 13
/dataset1/data2/what/nodata	Attribute, Table 13
/dataset1/data2/what/undetect	Attribute, Table 13
/dataset1/data2/data	Dataset
/dataset1/data3	Group
/dataset1/data3/what	Group
/dataset1/data3/what/quantity	Attribute, Table 16
/dataset1/data3/what/gain	Attribute, Table 13
/dataset1/data3/what/offset	Attribute, Table 13
/dataset1/data3/what/nodata	Attribute, Table 13
/dataset1/data3/what/undetect	Attribute, Table 13
/dataset1/data3/data	Dataset

7.4 RHI

The following example shows how a polar RHI is represented.

Table 21: Range-height indicator

Attribute name	Type
/	Root Group
/Conventions	Attribute, Section 4.1
/what	Group
/what/object	Atttribute, Table 2
/what/version	Atttribute, Table 1
/what/date	Atttribute, Table 1
/what/time	Atttribute, Table 1
/what/source	Atttribute, Table 3
/where	Group
/where/lon	Attribute, Table 4
/where/lat	Attribute, Table 4
/where/height	Attribute, Table 4
/how	Group
/how/task	Attribute, Table 8
/how/system	Attribute, Table 8
/how/software	Attribute, Table 8
/how/sw_version	Attribute, Table 8
/how/simulated	Attribute, Table 8
/how/beamwidth	Attribute, Table 8
/how/wavelength	Attribute, Table 8
/how/rpm	Attribute, Table 8
/how/pulsewidth	Attribute, Table 8
/how/RXbandwidth	Attribute, Table 8
/how/lowprf	Attribute, Table 8

continued on next page

continued from previous page

Attribute name	Type
/how/highprf	Attribute, Table 8
/how/TXlossH	Attribute, Table 8
/how/TXlossV	Attribute, Table 8
/how/RXlossH	Attribute, Table 8
/how/RXlossV	Attribute, Table 8
/how/radomelossH	Attribute, Table 8
/how/radomelossV	Attribute, Table 8
/how/antgainH	Attribute, Table 8
/how/antgainV	Attribute, Table 8
/how/beamwH	Attribute, Table 8
/how/beamwV	Attribute, Table 8
/how/gasattn	Attribute, Table 8
/how/radconstH	Attribute, Table 8
/how/radconstV	Attribute, Table 8
/how/nomTXpower	Attribute, Table 8
/how/TXpower	Attribute, Table 8
/how/NI	Attribute, Table 8
/how/Vsamples	Attribute, Table 8
/how/elmethod	Attribute, Table 8
/how/binmethod	Attribute, Table 8
/how/startelA	Attribute, Table 8
/how/stopelA	Attribute, Table 8
/how/startelT	Attribute, Table 8
/how/stopelT	Attribute, Table 8
/how/malfunc	Attribute, Table 8
/how/radar_msg	Attribute, Table 8
/how/NEZH	Attribute, Table 8
/how/Dclutter	Attribute, Table 8
/how/SQI	Attribute, Table 8
/how/CSR	Attribute, Table 8
/how/LOG	Attribute, Table 8
/how/RAC	Attribute, Table 8
/how/PAC	Attribute, Table 8
/how/S2N	Attribute, Table 8
/how/polarization	Attribute, Table 8
/dataset1	Group
/dataset1/what	Group
/dataset1/what/product	Attribute, Table 14
/dataset1/what/startdate	Attribute, Table 13
/dataset1/what/starttime	Attribute, Table 13
/dataset1/what/enddate	Attribute, Table 13
/dataset1/what/endtime	Attribute, Table 13
/dataset1/where	Group
/dataset1/where/az_angle	Attribute, Table 6
/dataset1/where/nbins	Attribute, Table 4

continued on next page

continued from previous page

Attribute name	Type
/dataset1/where/nrays	Attribute, Table 4
/dataset1/where/rstart	Attribute, Table 4
/dataset1/where/rscale	Attribute, Table 4
/dataset1/data1/what/quantity	Attribute, Table 16
/dataset1/data1/what/gain	Attribute, Table 13
/dataset1/data1/what/offset	Attribute, Table 13
/dataset1/data1/what/nodata	Attribute, Table 13
/dataset1/data1/what/undetect	Attribute, Table 13
/dataset1/data1	Group
/dataset1/data1/data	Dataset
/dataset1/data1/data/CLASS	Attribute, Table 17
/dataset1/data1/data/IMAGE_VERSION	Attribute, Table 17

A Derivation of the radar calibration constant

It is very important for the radar constant to be defined properly. The following definition is used.

$$C = 10\log_{10}\left(\frac{2.025 \cdot 2^{14} \cdot \ln(2) \cdot \lambda_{\mathrm{cm}}^2}{\pi^5 \cdot 10^{-23} \cdot c \cdot Pt_{\mathrm{kW}} \cdot \theta_{\mathrm{deg}} \cdot \phi_{\mathrm{deg}} \cdot \tau_{\mu\mathrm{s}} \cdot |K|^2}\right) - 2G + 2L_r + L_{\mathrm{Tx}} + L_{\mathrm{Rx}} \qquad (1)$$

where λ_{cm}^2 is the wavelength in cm,
c is the speed of light in vacuum in m/s (3.0×10^8),
Pt_{kW} is the nominal transmitted power in kW,
θ_{deg} and ϕ_{deg} are horizontal and vertical -3 dB beamwidths in degrees,
$\tau_{\mu\mathrm{s}}$ is the pulse length in microseconds,
$|K|^2$ is the dielectric factor (0.93 for liquid water),
G is the antenna gain in dB,
L_r is the one-way radome loss in dB,
L_{Tx} is the loss in the complete transmission chain, in dB, and
L_{Rx} is the loss in the complete reception chain, in dB.

This definition of the radar constant is based on the following radar equation

$$dBZ = C + 20\log_{10} r_{\mathrm{km}} + 10\log_{10} Pr_{\mathrm{mW}} \qquad (2)$$

where r_{km} is the distance from the radar in km, and
Pr_{mW} is the received power in mW.