WMO Data Model for Radial Radar and Lidar Data

Mark Curtis, Bureau of Meteorology, Melbourne, Australia
Michael Dixon, National Center for Atmospheric Research, Boulder, Colorado, USA
Daniel Michelson, Environment and Climate Change Canada, Toronto, Canada

1 May 2018

Release Notes

Version 1.0, 1 May 2018

Finalized draft

Version 0.1-0.6, 4 March 2017 – 8 February 2018

Early drafts

1 Introduction

This document describes a data model for the representation of weather radar and scanning lidar data, metadata and products. While effort has been made to be general, the weather-radar technology in question is assumed to be that commonly used in real-time operations throughout the world: scanning X, C, and S-band systems. Radar and lidar together in this context are referred to here as Pulsed Polar Systems (PPS).

The data model is based upon the WMO Information Model for Radial Radar and Lidar Data which describes the key objects, relationships and metadata necessary to facilitate the exchange of PPS data. This document introduces logical, structural and representational constraints which act as a bridge between the primarily conceptual information model and the primarily technical file formats which may be used to implement it.

As is noted in section 1.1 of the information model, there are several types of data which may be relevant to the exchange of PPS data. Of these only Level 2, related to information organised in native polar coordinates by rays, bins, and quantities, is addressed by this data model.

1.1 Relationship to information model

The information model introduces the fundamental objects required to represent PPS data and the metadata associated with them. The relationships between objects are defined in an idealized manner as a simple hierarchy of types. Each instance of a type contains any number of instances of the type at the next level of the hierarchy. This arrangement is shown in Figure 1.

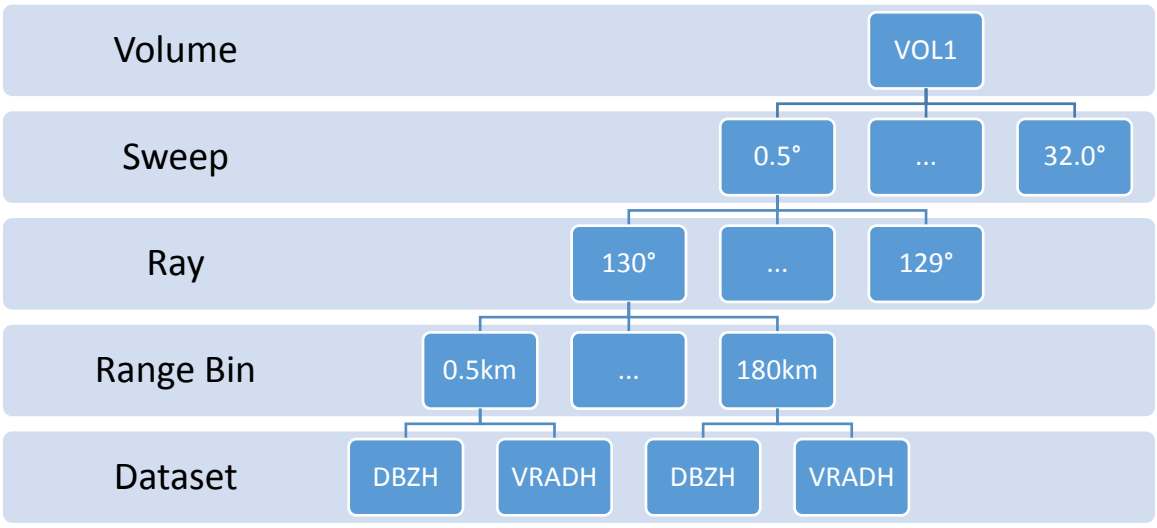

Figure 1. Information Model, Object Model Hierarchy

While this arrangement is conceptually simple, it leads to a degree of flexibility in the relationships between objects which is unnecessary for practical PPS data exchange purposes. Supporting the full flexibility of the idealized relationships within a file format is difficult, and such a format is likely to be very complex. Although such formats may be relatively easy for a data producer to write, they tend to be difficult for a data consumer to use. In this way excessive flexibility in the object model hinders the goal of facilitating international PPS data exchange.

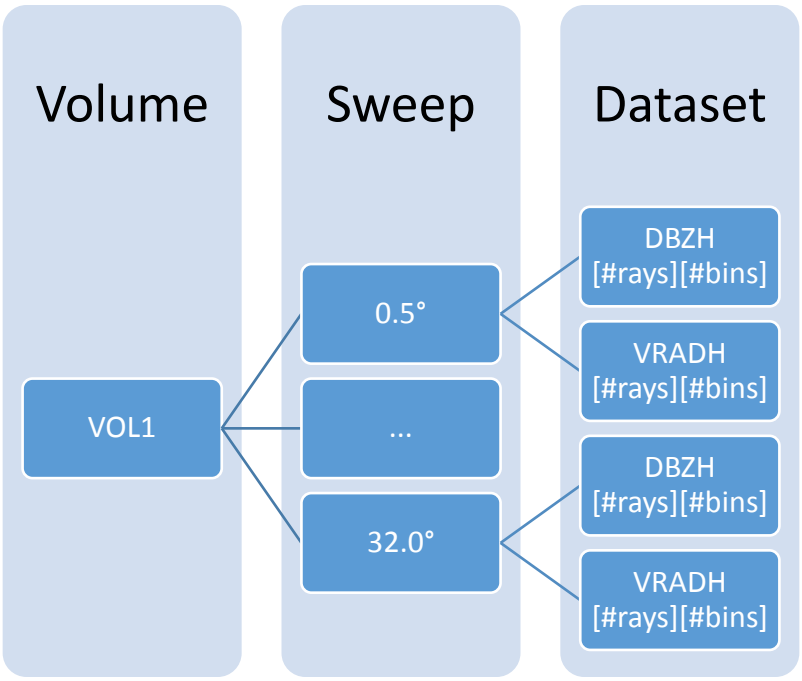

Figure 2. Information Model, Object model hierarchy as refined for efficient storage

The need for a simplified representation is recognized by the information model which also provides the object model hierarchy shown in Figure 2. This model facilitates efficient storage and lower complexity of use by considering the ways that PPS data are acquired and distributed in practice. It is this refined view of the information model object hierarchy which is addressed by the data model.

1.2 Relationship to file format

This document does not attempt to specify a file format that should be used to represent the data. Rather, it provides a set of structural and representational requirements which aid in mapping between the conceptual objects of the information model and the corresponding physical representation that would be specified by a file format.

In this regard, the data model may be thought of as providing guidance which aligns with the way a user expects to access and manipulate the data. In principle, this document could be used as the basis for the development of an API through which PPS data may be accessed independent of underlying file formats.

For example, the data model may specify that the values of a dataset object are represented as an array of 32-bit floats ordered in a specific way. An implementing file format is free to store this data in any way provided that it is accessed in a manner consistent with the array of 32-bit floats specified by the data model.

2 Data Model

2.1 Overview

2.1.1 Data type encodings

The information model defines each metadata attribute as belonging to one of the following fundamental classes: “integer”, “real”, “Boolean”, “string”, or “enum”. Additionally, a special data type of “time” is identified with subtypes for absolute and relative times. The data model nominates a set of specific encodings for each of these types. One or more of these encodings is associated with each data or metadata item in the data model. Implementing file formats must expose the data and metadata items to the user (whether directly or via an API) using the specified encoding.

Table 1. Data type encodings

Type	Encoding	Description
integer	int8	8-bit signed integer. Minimum representable range of -128 to 127 inclusive.
integer	int16	16-bit signed integer in host native endianness. Representable range of -2^{15} to $2^{15}-1$ inclusive.
integer	int32	32-bit signed integer in host native endianness. Representable range of -2^{31} to $2^{31}-1$ inclusive.
integer	int64	64-bit signed integer in host native endianness. Representable range of -2^{63} to $2^{63}-1$ inclusive.
integer	uint8	8-bit unsigned integer. Representable range of 0 to 255 inclusive.
integer	uint16	16-bit unsigned integer in host native endianness. Representable range of 0 to $2^{16}-1$ inclusive.
integer	uint32	32-bit unsigned integer in host native endianness. Representable range of 0 to $2^{32}-1$ inclusive.
integer	uint64	64-bit unsigned integer in host native endianness. Representable range of 0 to $2^{64}-1$ inclusive.
real	float32	32-bit floating point value in IEEE 754-2008 binary32 format.
real	float64	64-bit floating point value in IEEE 754-2008 binary64 format.
Boolean	bool	Boolean value in native or conventional Boolean type if available.
string	utf8	Null-terminated UTF-8 character string.
enum	enum	Null-terminated UTF-8 character string.
time (absolute)	iso8601	Absolute UTC time as a null-terminated UTF-8 character string conforming to ISO 8601 standard. The representation used is ‘YYYY-MM-DDThh:mm:ssZ’ for low precision and ‘YYYY-MM-DDThh:mm:ss.ddddddddZ’ for high precision.
time (relative)	float64	A relative time from a reference absolute time (above) is expressed as a float64 offset in seconds with nanosecond precision (eight decimal places), where a positive value indicates a time after the reference.

2.1.2 Freedom of implementation of storage scheme

Implementing file formats are free to store data and metadata using any storage scheme provided such schemes are transparent to the user of the data. This means that the data must be presented to the user (typically through an API) in the encoding type specified for the item by the data model.

Additionally, any storage scheme must retain the full semantics of the encoded type as specified in Table 1. Specifically:

- The storage scheme must provide precision equal to or greater than the data model encoding.
- Special values available in the nominated encoding type must be representable in the storage scheme. For example, the floating-point NaN and infinity values.
- The storage scheme must not introduce a dependency on data which is external to the stored file itself. For example, an enum type may be stored as an integer only if a mapping between the integer values and enumerate labels is also stored within the file.

2.1.3 Representation of optional metadata

Where metadata is nominated to be optional it must be possible for the user to distinguish between presence and non-presence of the value. This means that implementing file formats should simply omit missing values rather than providing defaults.

2.1.4 Inversion of arrays of metadata groups permitted

Metadata associated with objects are collected into logical groups which are detailed in section 3. It is necessary for some objects to contain multiple instances of a single metadata group. For example, the sweep object must contain multiple instances of the "ray characteristics" metadata group; one per ray. Within the data model this arrangement is presented as if the object contains an array of the metadata group itself.

To facilitate efficient storage and ease of use, it is permissible for implementations to invert this arrangement such that an object contains several arrays (one per metadata item) which together make up the logical group.

A pseudo code demonstration of such an inversion is shown in Figure 3. Although the data model is specified in the "array of groups" form, file formats may implement this in the "group of arrays" form. This inversion is permitted at both the storage and user levels. That is, the inversion may be apparent to the user and does not need to be hidden.

Array of Groups

```
type md-group
{
  metadata a
  metadata b
  metadata c
}

type object
{
  md-group group[N]
}
```

Group of Arrays

```
type md-group
{
  metadata a[N]
  metadata b[N]
  metadata c[N]
}

type object
{
  md-group group
}
```

Figure 3. Inversion of metadata group arrays

2.1.5 Hierarchical metadata permitted

Implementations are permitted, although not required, to allow metadata which is identical in value for all objects at a level of the object hierarchy to be stored at a higher level. Metadata which has been promoted in this way shall be considered to apply to all lower objects. Metadata provided at a higher level it must be either omitted entirely from lower levels, or provided with identical values.

For example, if all rays within a volume have the same pulse width then the implementation may allow the pulse width metadata to be stored at the volume level and omitted from all sweep and ray metadata.

Where permitted by the implementation, promotion of common metadata to a higher level should be detectible by the user. This allows users to optimize usage patterns and processing chains based on the consistency of metadata over multiple objects. Implementations are also encouraged to ensure that users are not burdened with the task of searching the object hierarchy for desired metadata. This would typically be achieved via redundant storage of the metadata at lower levels, or through an API which automates the search process.

2.1.6 User defined metadata

The information model (Section 3.1.4) specifies that user defined metadata may be freely associated with any object. The data model restricts the use of user defined metadata in the following ways:

- All user defined metadata must be identified by a name in the utf8 encoding as per Table 1.
- The names assigned to user defined metadata must be unique within the context of the object to which they are applied. This includes both standard metadata and other user defined metadata.
- The values of user defined metadata must be represented using one of the encodings specified by Table 1. The value may be either a scalar or one dimensional array.

Implementing file formats shall not explicitly distinguish between standard metadata that is defined by the data model and user defined metadata. This ensures that user metadata with wide community acceptance may be officially adopted into the information and data models without triggering changes at the file format level. In effect, implementing file formats must be forward compatible to the standardization of new metadata.

An implication of this requirement is that file formats must provide a mechanism for discovery of the metadata associated with an object. Practically this means that a user must be able to iterate over the metadata of an object to retrieve the name, encoding type and value of each item.

2.1.7 Self-describing files

Implementations are strongly encouraged to store extended metadata within the file format to provide contextual information about the object metadata itself. Such ‘meta-metadata’ will allow the file to be self-describing. A self-describing file provides sufficient information about its contents that it may be understood and used correctly without the need for external documentation or context.

For example, the Nyquist velocity metadata attribute is specified to be encoded as float64. A minimally conforming file format need only store the name, encoding type and value of this attribute. This is sufficient for a user to find and read the value, but insufficient to understand the value semantically. A

self-describing file will also provide meta-metadata such as the units of the quantity (in this case m/s) so that the value can be understood without reference the information model.

2.2 Volume object

The volume is the top-level object represented by the information and data models. This object contains a sequence of any number of logically associated sweep objects. The order of the sweeps within the sequence is unspecified, however will typically be the acquisition order of the data. It is permissible for a volume object to contain zero sweeps.

Although the order of sweeps is not specified, it is required that the order be well defined for any given volume. This allows a user of the volume object to access a sweep by its index and be sure that the index will not change upon closing and reopening the (unmodified) file. Implementing file formats are required to allow random access to individual sweep objects. Users must not be required to load or otherwise process preceding sweeps to access the nth sweep of a volume.

Table 2. Volume object contents

Description	Type	Reference
Product information	metadata	3.2.1
Geographic reference information	metadata	3.2.2
Instrument characteristics (radar or lidar depending on IMID 1.0)	metadata	3.2.3/3.2.4
User defined metadata	metadata	2.1.6
Number of sweeps	uint64	
Sweep objects	sweep[]	2.3

2.3 Sweep object

The sweep object represents the middle level of the object hierarchy. This object contains a collection of datasets for which certain fundamental properties must remain constant. These properties include the ray/range bin geometries and target fixed angle (e.g. elevation angle of a PPI sweep). If a single scan is to be represented during which any of these properties is modified, then it must be stored as multiple sweeps.

The simplification of the object model shown in Figure 1 into the more storage efficient model shown in Figure 2 results in the removal of the ray and range bin levels of the type hierarchy. Since these objects are no longer explicitly expressed by the model, the metadata for these levels is stored directly within the sweep.

Additionally, a special case of ‘almost’ constant metadata is recognized by the data model. When multiple pulse widths are used within a sweep some metadata may be related directly to the instrument calibration at that pulse width rather than the sweep itself. An attribute of the ray metadata (IMID 8.7) allows the association of calibration specific metadata with individual rays.

Because of the above, metadata which is stored by the sweep may be stored per sweep, per calibration, per ray or per range bin. User defined metadata is permitted to be stored under any of these regimes.

Dataset objects contained by a sweep must be accessible via index and dataset identifier. As with volumes, the order of the contained objects is not specified but must be well defined to support random access patterns.

The dataset identifier (IMID 12.0) must be unique over all datasets of a sweep. It is permissible for multiple datasets to represent the same quantity (IMID 12.1) if each dataset is given a separate identifier (IMID 12.0).

Table 3. Sweep object contents

Description	Type	Reference
Sweep characteristics	metadata	3.3.1
User defined metadata	metadata	2.1.6
Number of instrument calibrations	uint64	
Instrument calibration (radar or lidar)	metadata[calibrations]	3.3.2/3.3.3
Calibration specific user defined metadata	metadata[calibrations]	2.1.6
Number of rays	uint64	
Ray characteristics	metadata[rays]	3.4.1
Moving platform geographic reference information	metadata[rays]	3.4.2
Radar monitoring information	metadata[rays]	3.4.3
Ray specific user defined metadata	metadata[rays]	2.1.6
Number of range bins	uint64	
Range bin metadata	metadata[bins]	3.5.1
Range bin specific user defined metadata	metadata[bins]	2.1.6
Datasets	dataset[]	2.4

2.4 Dataset object

2.4.1 Scalar dataset object

The scalar dataset object contains all the measurements of a single PPS data type or variable for every ray and range bin within a sweep. The data is represented as a 2D array of values where each row of the array corresponds to a ray, and each column of the array corresponds to a range bin.

The values stored in the array do not directly represent the dataset quantity as specified by the quantity name and units (IMID 12.1, 12.2). Rather, the stored value is converted to the dataset quantity through use of the following equation:

$$quantity = stored \times gain + offset$$

The gain and offset are constants which are selected by the user when the dataset is written and stored as part of the dataset object. The fundamental type of the array is also selected by the user and must be one of the following encodings from Table 1: int8, int16, int32, int64, uint8, uint16, uint32, uint64, float32, float64.

This flexibility in choice of gain, offset and encoding type for a dataset allows the user to select a quantization for the dataset which retains an acceptable accuracy for the quantity while minimizing the size of the resulting file. Implementations are encouraged to support additional lossless compression of dataset arrays.

It is expected that reading and writing quantity data will always involve the entire dataset. Implementing file formats are therefore not required to support partial or random access to the data array.

Table 4 Scalar dataset object contents

Description	Type	Reference
Basic dataset information	metadata	3.6.1
Quality dataset information	metadata	3.6.2
User defined metadata	metadata	2.1.6
Gain used in conversion of encoded values to quantity	float64	
Offset used in conversion of encoded values to quantity	float64	
Data	T[rays][bins]	

2.4.2 Spectrum dataset object

Conceptually the spectrum dataset object contains a vector of samples of a single PPS data type or variable for every ray and range bin within a sweep. Due to the potentially large size of this dataset it is common practice to filter out spectra corresponding to locations which fail some interest criteria (such as a minimum signal to noise ratio).

To facilitate efficient storage of filtered spectrum datasets this object is represented as two densely packed arrays; a data array and an index array. The data array contains the vectors of samples making up each spectrum while the index array is used to map between ray/bin and the corresponding spectrum in the data array. A spectra index value of -1 is used to indicate that there is no spectrum stored for the ray/bin. This arrangement ensures that filtered (missing) spectra consume no space in the data array.

The sample values contained by the spectrum data array are subject to the same gain and offset based packing as the scalar dataset object described in section 2.4.1.

Table 5 Spectrum dataset object contents

Description	Type	Reference
Basic dataset information	metadata	3.6.1
Quality dataset information	metadata	3.6.2
Spectrum dataset information	metadata	3.6.3
User defined metadata	metadata	2.1.6
Gain used in conversion of encoded values to quantity	float64	
Offset used in conversion of encoded values to quantity	float64	
Number of samples per spectra	uint64	
Number of spectra	uint64	
Spectrum index	Int64[rays][bins]	
Spectrum data	T[spectra][samples]	

3 Standard Metadata Encodings

This section describes the data type encoding for each metadata which may be associated with the objects detailed in section 2.

3.1 Overview

3.1.1 Relationship to information model

Where a metadata item was originally identified within the information model the information model metadata ID is also provided (IMID).

Data model specifies only the encoding for the standardized metadata. This document should be used in conjunction with the information model which details the semantics of these value such as units and minimum required precision.

The information model convention of indicating required metadata using a shaded background is retained.

3.2 Volume Object Metadata

3.2.1 Product information

IMID	Description	Encoding
1.0	Instrument type, distinguishing between “radar” and “lidar”	utf8
1.1	Site identifier, e.g. WIGOS identifier*	utf8
1.2	Volume start time	iso8601
1.3	Volume end time	iso8601
1.4	Scan strategy name	utf8
1.5	Instrument identifier (e.g. WSR-88D)	utf8
1.6	Whether instrument has malfunctioned	bool
1.7	Instrument error message	utf8
1.8	Whether acquired data are simulated	bool

*The WIGOS identifier structure consists of four parts. The part of the structure called “Local identifier” is the only part consisting of characters. Following the ODIM convention (Michelson et al., 2014), it is suggested as a best practice that the local identifier be harmonized to a five-character string, where the first two characters are the member country’s ISO 3166-1 alpha 2 ccTLD code (lower case), and the latter three characters are freely-selectable (also lower case).

3.2.2 Geographical reference information

IMID	Description	Encoding
2.0	Site longitude	float64
2.1	Site latitude	float64
2.2	Site altitude above geodetic datum. For a scanning instrument this is the center of rotation of the antenna.	float64
2.3	Geodetic datum name	utf8
2.4	Site altitude above ground level	float64
2.5	Magnetic declination at site, positive clockwise	float64
2.6	Whether platform is moving	bool

3.2.3 Radar characteristics

The metadata in this section only apply to instrument type 'radar'.

IMID	Description	Encoding
3.0	Nominal antenna gain H	float64
3.1	Nominal antenna gain V	float64
3.2	Antenna beam width H	float64
3.3	Antenna beam width V	float64
3.4	Bandwidth of radar receiver	float64
3.5	Frequency	float64
3.6	Transmitter type, ie. Magnetron, Klystron, or Solid state	utf8
3.7	Manufacturer name	utf8
3.8	Model name	utf8
3.9	Signal processor name	utf8
3.10	Signal processor version	utf8

3.2.4 Lidar characteristics

IMID	Description	Encoding
4.0	Beam divergence (transmit side)	float64
4.1	Field of view (receive side)	float64
4.2	Aperture diameter	float64
4.3	Aperture efficiency	float64
4.4	Peak power	float64
4.5	Pulse energy	float64

3.3 Sweep Object Metadata

3.3.1 Sweep characteristics

IMID	Description	Encoding
5.0	Sweep mode	enum (Table 6)
5.1	Target fixed angle (elevation angle for PPI mode, azimuth angle for RHI mode)	float64
5.2	Target scan rate	float64
5.3	Polarization mode	enum (Table 7)
5.4	PRT mode	enum (Table 8)
5.5	Distance to centre of first range bin	float64

3.3.2 Radar calibration

The metadata in this section only apply to instrument type 'radar'.

ID	Description	Encoding
6.0	Time of calibration	time

		(absolute)
6.1	Pulse width	float64
6.2	Derived antenna gain H	float64
6.3	Derived antenna gain V	float64
6.4	Nominal transmit power H	float64
6.5	Nominal transmit power V	float64
6.6	2-way waveguide loss measurement plane to feed horn H	float64
6.7	2-way waveguide loss measurement plane to feed horn V	float64
6.8	2-way radome loss H	float64
6.9	2-way radome loss V	float64
6.10	Receiver filter bandwidth mismatch loss H	float64
6.1	Receiver filter bandwidth mismatch loss V	float64
6.12	Radar constant H	float64
6.13	Radar constant V	float64
6.14	Probert Jones correction	float64
6.15	Dielectric factor $\|K^2\|$	float64
6.16	Measured noise level H co-polar	float64
6.17	Measured noise level V co-polar	float64
6.18	Measured noise level H cross-polar	float64
6.19	Measured noise level V cross-polar	float64
6.20	Measured receiver gain H co-polar	float64
6.21	Measured receiver gain V co-polar	float64
6.22	Measured receiver gain H cross-polar	float64
6.23	Measured receiver gain V cross-polar	float64
6.24	Reflectivity at 1km for SNR=0dB H co-polar	float64
6.25	Reflectivity at 1km for SNR=0dB V co-polar	float64
6.26	Reflectivity at 1km for SNR=0db H cross-polar	float64
6.27	Reflectivity at 1km for SNR=0db V cross-polar	float64
6.28	Calibrated sun power H co-polar	float64
6.29	Calibrated sun power V co-polar	float64
6.30	Calibrated sun power H cross-polar	float64
6.31	Calibrated sun power V cross-polar	float64
6.32	Noise source power H	float64
6.33	Noise source power V	float64
6.34	Power measurement loss in coax and connectors H	float64
6.35	Power measurement loss in coax and connectors V	float64
6.36	Coupler loss into waveguide H	float64
6.37	Coupler loss into waveguide V	float64
6.38	ZDR correction	float64
6.39	LDR correction H	float64
6.40	LDR correction V	float64
6.41	System PhiDP as seen in drizzle close to the radar	float64
6.42	Calibration test power H	float64
6.43	Calibration test power V	float64
6.44	Computed receiver slope H co-polar	float64
6.45	Computed receiver slope V co-polar	float64

6.46	Computed receiver slope H cross-polar	float64
6.47	Computed receiver slope V cross-polar	float64

3.3.3 Lidar calibration

No calibration metadata for lidar instruments are currently identified.

3.4 Ray Object Metadata

3.4.1 Ray characteristics

IMID	Description	Encoding
8.0	Elevation angle	float64
8.1	Azimuth angle	float64
8.2	Time of acquisition (relative to volume start time)	relative time
8.3	Width of ray (dwell)	float64
8.4	Measured scan rate, positive clockwise and/or ascending	float64
8.5	Whether the antenna is in transition to fixed angle during this ray	bool
8.6	Whether geographic reference information for moving platforms has been applied to correct the elevation and azimuth angles	bool
8.7	Calibration index Note: This metadata is used to index into the instrument calibration and calibration specific user defined metadata arrays of the containing sweep (see section 2.3).	uint64
8.8	Pulse repetition time(s), ordered by transmission sequence (if known)	float64[]
8.9	Nyquist velocity	float64
8.10	Unambiguous range	float64
8.11	Number of samples used to compute moments	int64

3.4.2 Moving platform geographic reference information

The shaded metadata of this section are only required for moving platforms.

IMID	Description	Encoding
9.0	Latitude of the instrument	float64
9.1	Longitude of the instrument	float64
9.2	Altitude of the instrument above the geodetic datum. For scanning PPS, this is the center of rotation of the antenna.	float64
9.3	Heading of the platform relative to true north, looking down from above	float64
9.4	Roll about longitudinal axis of platform. Positive is left side up, looking forward.	float64
9.5	Pitch about the lateral axis of the platform. Positive is up at the front.	float64
9.6	Difference between heading and track over the ground (drift). Positive drift implies track is clockwise from heading, looking from above. Not applicable	float64

	to land-based moving platforms.	
9.7	Angle between the PPS beam and the vertical axis of the platform (rotation). Zero is along the vertical axis, positive is clockwise looking forward from behind the platform.	float64
9.8	Angle between the radar beam (when it is in a plane containing the longitudinal axis of the platform) and a line perpendicular to the longitudinal axis (tilt). Zero is perpendicular to the longitudinal axis, positive is towards the front of the platform.	float64
9.9	East/west velocity of the platform. Positive is eastwards.	float64
9.10	North/south velocity of the platform. Positive is northwards.	float64
9.11	Vertical velocity of the platform. Positive is upwards.	float64
9.12	East/west wind at the platform location. Positive is eastwards.	float64
9.13	North/south wind at the platform location. Positive is northwards.	float64
9.14	Vertical wind at the platform location. Positive is upwards.	float64
9.15	Rate of change of heading	float64
9.16	Rate of change of roll of the platform	float64
9.17	Rate of change of pitch of the platform	float64
9.18	Correction in azimuth	float64
9.19	Correction in elevation	float64
9.20	Correction in range	float64
9.21	Correction in longitude	float64
9.22	Correction in latitude	float64
9.23	Correction in pressure altitude	float64
9.24	Correction in radar altitude	float64
9.25	Correction in east-west ground speed	float64
9.26	Correction in north-south ground speed	float64
9.27	Correction in vertical velocity	float64
9.28	Correction in heading	float64
9.29	Correction in roll	float64
9.30	Correction in pitch	float64
9.31	Correction in drift	float64
9.32	Correction in rotation	float64
9.33	Correction in tilt	float64

3.4.3 Radar monitoring

If it is not possible to obtain the following metadata at the ray or sweep level, they may be represented at the volume level. Some of these attributes may be more relevant to the higher-order object levels. Some are diagnostic in nature, i.e. analyzed after data have been acquired.

IMID	Description	Encoding
10.0	Measured transmit power H	float64
10.1	Measured transmit power V	float64
10.2	Noise measured at the receiver when connected to the antenna with no noise source connected	float64
10.3	Noise measured at the receiver when connected to the noise source which is disabled	float64

10.4	Noise measured at the receiver when it is connected to the noise source which is enabled	float64
10.5	Phase difference between transmitted horizontally and vertically-polarized signals as determined from the first valid range bins	float64
10.6	Antenna-pointing accuracy in elevation	float64
10.7	Antenna-pointing accuracy in azimuth	float64
10.8	Calibration offset for the horizontal channel	float64
10.9	Calibration offset for the vertical channel	float64
10.10	ZDR offset	float64

3.5 Range Bin Object Metadata

3.5.1 Range bin characteristics

IMID	Description	Encoding
11.0	Length of range bin	float64

3.6 Dataset Object Metadata

3.6.1 Basic dataset information

IMID	Description	Encoding
12.0	Dataset identifier (user specified)	utf8
12.1	Quantity name (see section 4)	utf8
12.2	Quantity units	utf8
12.3	Quantity value used to indicate missing data	float64
12.4	Quantity value used to indicate no signal	float64
12.5	Whether dataset is represented by discrete values	bool
12.6	Discrete values used in dataset	float64[]
12.7	Labels for discrete values used in dataset	utf8[]
12.8	Whether dataset is a quality dataset	bool
12.9	Identifiers of quality datasets which qualify this dataset	utf8[]

3.6.2 Quality dataset information

IMID	Description	Encoding
13.0	Identifiers of datasets which are qualified by this dataset	utf8[]
13.1	Identifier of the algorithm that generated the dataset (see below)	utf8
13.2	Arguments or configuration provided to the algorithm that generated the dataset	utf8[]
13.3	Literature reference to the algorithm that generated the dataset	utf8

3.6.3 Spectrum dataset

IMID	Description	Encoding
15.0	Value represented by each point in the spectrum	float64[]
15.1	Length of FFT used to compute the spectrum	int64
15.2	Length of averaging block used to compute the spectrum	int64

4 Standard Dataset and Enumerate Names

The information model identified a list of standard datasets associated with polar pulsed systems. The data model nominates standard strings to be used as the quantity name (IMID 12.2) for each dataset.

Additionally, standard strings are specified for the values of each enum encoded metadata attribute. Users are free to select any utf8 string that does not clash with one of the standard strings listed here when adding specialized enum values.

4.1 Scalar dataset quantities

IMID	Description	Quantity name
16.0	Equivalent reflectivity factor	DBZ
16.1	Linear equivalent reflectivity factor	Z
16.2	Radial velocity of scatterers away from instrument	VEL
16.3	Doppler spectrum width	WIDTH
16.4	Log differential reflectivity H/V	ZDR
16.5	Log-linear depolarization ratio HV	LDR
16.6	Log-linear depolarization ratio H	LDRH
16.7	Log-linear depolarization ratio V	LDRV
16.8	Differential phase HV	PHIDP
16.9	Specific differential phase HV	KDP
16.10	Cross-polar differential phase	PHIHX
16.11	Cross-correlation ratio HV	RHOHV
16.12	Co-to-cross polar correlation ratio H	RHOHX
16.13	Co-to-cross polar correlation ratio V	RHOXV
16.14	Log power	DBM
16.15	Log power co-polar H	DBMHC
16.16	Log power cross-polar H	DBMHX
16.17	Log power co-polar V	DBMVC
16.18	Log power cross-polar V	DBMVX
16.10	Signal-to-noise ratio	SNR
16.20	Signal-to-noise ratio co-polar H	SNRHC
16.21	Signal-to-noise ratio cross-polar H	SNRHX
16.22	Signal-to-noise ratio co-polar V	SNRVC
16.23	Signal to noise ratio cross polar V	SNRVX
16.24	Normalized coherent power	NCP
16.25	Rain rate	RR
16.26	Radar echo classification	REC

4.2 Spectrum dataset quantities

IMID	Description	Quantity name
17.0	Spectrum of co-polar H	SPEC_HC
17.1	Spectrum of co-polar V	SPEC_VC
17.2	Spectrum of cross-polar H	SPEC_HX
17.3	Spectrum of cross-polar V	SPEC_VX
17.4	Cross spectrum of co-polar H	XSPEC_HC

17.5	Cross spectrum of co-polar V	XSPEC_VC
17.6	Cross spectrum of cross-polar H	XSPEC_HX
17.7	Cross spectrum of cross-polar V	XSPEC_VX

4.3 Metadata enumerate names

Table 6. Sweep mode (IMID 5.0) enumerate values

Description	Standard String
Plan Position Indicator (PPI)	ppi
Range-Height Indicator (RHI)	rhi
Vertical	vertical_pointing
Sun scan	sunscan
Note: other specialized sweep modes are permitted	

Table 7. Polarization mode (IMID 5.3) enumerate values

Description	Standard String
Horizontal	horizontal
Vertical	vertical
Horizontal-vertical alternating	hv_alt
Horizontal-vertical simultaneous	hv_sim
Circular	circular
Note: other specialized polarization modes are permitted	

Table 8. PRT mode (IMID 5.4) enumerate values

Description	Standard String
Fixed	fixed
Staggered	staggered
Dual	dual
Note: other specialized PRT modes are permitted	

5 References

Michelson D., Curtis M., Dixon M., Haase G., Horvat C., Joe P., Umehara A., 2016: WMO Information Model for Radial Radar and Lidar Data. Version 1.0. WMO TT-WRDE. 20pp.

Dixon M., Lee, W-C., Rilling B., Burghart C., and Van Andel J., 2013: CfRadial Data File Format. Proposed CF-compliant netCDF Format for Moments Data for RADAR and LIDAR in Radial Coordinates. Version 1.3. EOL, NCAR. 66 pp.

Michelson D.B., Lewandowski R., Szewczykowski M., Beekhuis H., and Haase G., 2014: EUMETNET OPERA weather radar information model for implementation with the HDF5 file format. Version 2.2. EUMETNET OPERA Output O4. 38 pp.