

# **CfRadial2 Data File Format**

## **CF2 NetCDF Format for RADAR and LIDAR data in Radial Coordinates**

**Version 2.1 - DRAFT**

Mike Dixon, EOL, NCAR, Boulder, Colorado, USA

Mark Curtis, Bureau of Meteorology, Melbourne, Australia

Daniel Michelson, Environment and Climate Change Canada, Toronto, Canada

Joe Hardin, PNNL, DOE, Richland, Washington, USA

Ken Kehoe, University of Oklahoma, Norman, Oklahoma, USA

Samuel Haimov, University of Wyoming, Laramie, Wyoming, USA

**Updated 2019-09-01**

## Table of Contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
1.1	PURPOSE .....	5
1.2	ON-LINE LOCATION .....	5
1.3	TERMINOLOGY – CFRADIAL1 AND CFRADIAL2 .....	5
1.4	HISTORY .....	5
1.5	CF2 AND CFRADIAL2 .....	6
1.6	ON-LINE URLS.....	7
<b>2</b>	<b>RADAR/LIDAR DATA INFORMATION MODEL.....</b>	<b>8</b>
2.1	LOGICAL ORGANIZATION OF DATA IN A VOLUME, USING SWEEPS AND RAYS .....	8
2.2	LOGICAL ORGANIZATION OF DATA IN A VOLUME, USING SWEEPS WITH 2-D DATA .....	10
2.3	FIELD DATA BYTE REPRESENTATION.....	12
2.4	SCANNING MODES .....	12
2.5	GEO-REFERENCE VARIABLES.....	13
<b>3</b>	<b>STRUCTURE OF CFRADIAL2 USING NETCDF4 WITH GROUPS.....</b>	<b>14</b>
3.1	GROUP-BASED DATA STRUCTURE .....	14
3.2	PRINCIPAL DIMENSIONS AND COORDINATE VARIABLES.....	17
3.3	_FILLVALUE AND MISSING_VALUE ATTRIBUTES FOR DATA FIELDS .....	18
3.4	REQUIRED VS. OPTIONAL VARIABLES.....	18
3.5	GRID MAPPING VARIABLE – RADAR_LIDAR_RADIAL_SCAN.....	18
3.6	EXTENSIONS TO THE CF CONVENTION .....	19
<b>3.7</b>	<b>STRING TYPES .....</b>	<b>19</b>
<b>4</b>	<b>ROOT GROUP.....</b>	<b>21</b>
4.1	GLOBAL ATTRIBUTES.....	21
4.2	GLOBAL DIMENSIONS.....	22
4.3	GLOBAL VARIABLES .....	22
<b>5</b>	<b>SWEEP GROUPS.....</b>	<b>25</b>
5.1	SWEEP-SPECIFIC DIMENSIONS .....	25
5.2	SWEEP COORDINATE VARIABLES .....	25
5.2.1	<i>Attributes for time coordinate variable .....</i>	<i>25</i>
5.2.2	<i>Attributes for range coordinate variable.....</i>	<i>26</i>
5.3	SWEEP VARIABLES.....	26
5.3.1	<i>Attributes for azimuth(time) variable .....</i>	<i>29</i>
5.3.2	<i>Attributes for elevation(time) variable .....</i>	<i>30</i>
5.4	THE GEOREFERENCE SUB-GROUP .....	30

5.5	THE <i>MONITORING</i> SUB-GROUPS .....	32
5.6	FIELD DATA VARIABLES .....	33
5.6.1	<i>Use of scale_factor and add_offset</i> .....	36
5.6.2	<i>Use of coordinates attribute</i> .....	36
5.6.3	<i>Use of flag values - optional</i> .....	36
5.6.4	<i>Flag mask fields - optional</i> .....	37
5.6.5	<i>Quality control fields - optional</i> .....	37
5.6.6	<i>Thresholding XML</i> .....	37
5.6.7	<i>Legend XML</i> .....	38
<b>6</b>	<b>SPECTRUM GROUPS.....</b>	<b>39</b>
6.1	SPECTRUM-SPECIFIC INFORMATION ON SWEEP .....	39
6.2	<i>SPECTRUM_INDEX</i> VARIABLE .....	39
6.3	SPECTRUM GROUP DIMENSIONS .....	39
6.4	SPECTRUM FIELD VARIABLES .....	40
6.5	SPECTRUM FIELD ATTRIBUTES .....	40
<b>7</b>	<b>ROOT GROUP METADATA GROUPS .....</b>	<b>42</b>
7.1	THE <i>RADAR_PARAMETERS</i> SUB-GROUP .....	42
7.2	THE <i>LIDAR_PARAMETERS</i> SUB-GROUP .....	42
7.3	THE <i>RADAR_CALIBRATION</i> SUB-GROUP .....	42
7.3.1	<i>Dimensions</i> .....	43
7.3.2	<i>Variables</i> .....	43
7.4	THE <i>LIDAR_CALIBRATION</i> SUB-GROUP .....	46
7.5	THE <i>GEOREFERENCE_CORRECTION</i> SUB-GROUP .....	46
<b>8</b>	<b>STANDARD NAMES.....</b>	<b>48</b>
<b>9</b>	<b>COMPUTING THE DATA LOCATION FROM GEO-REFERENCE VARIABLES.....</b>	<b>49</b>
9.1	SPECIAL CASE – GROUND-BASED, STATIONARY AND LEVELED SENSORS .....	50
9.1.1	<i>LIDARs</i> .....	50
9.1.2	<i>RADARs</i> .....	50
9.2	MOVING PLATFORMS .....	51
9.3	COORDINATE TRANSFORMATIONS FOR THE GENERAL CASE .....	52
9.3.1	<i>Coordinate systems</i> .....	52
9.3.2	<i>The earth-relative coordinate system</i> .....	52
9.3.3	<i>The platform-relative coordinate system</i> .....	52
9.3.4	<i>The sensor coordinate system</i> .....	55
9.4	COORDINATE TRANSFORMATION SEQUENCE .....	56
9.4.1	<i>Transformation from <math>X_i</math> to <math>X_a</math></i> .....	56
9.4.2	<i>Rotating from <math>X_a</math> to <math>X</math></i> .....	57

9.5	SUMMARY OF TRANSFORMING FROM $X_I$ TO $X$ .....	59
9.5.1	<i>For type Z radars:</i> .....	59
9.5.2	<i>For type Y radars:</i> .....	59
9.5.3	<i>For type Y-prime radars:</i> .....	59
9.5.4	<i>For type X radars:</i> .....	60
9.5.5	<i>Computing earth-relative azimuth and elevation</i> .....	60
9.6	SUMMARY OF SYMBOL DEFINITIONS .....	60
10	REFERENCES .....	61

# 1 Introduction

---

## 1.1 Purpose

---

The CfRadial2 format is intended to provide a standard, modern, self-describing way to store radar and lidar data in native radial (polar) coordinates.

CfRadial2 is built on NetCDF-4, which is in turn built on HDF5. NetCDF-4 and HDF5 are both up-to-date technology for efficiently and easily storing scientific and engineering data.

The purpose of this document is to specify a CF2-compliant NetCDF format for radar and lidar moments data in radial (i.e. polar) coordinates.

The intention is that the format should, as far as possible, comply with the CF conventions for gridded data. However, the current CF 2.0 convention does not support radial radar/lidar data. Therefore, extensions to the conventions will be required.

The current CF conventions are documented at:

<http://cfconventions.org/>

<http://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/cf-conventions.pdf>

## 1.2 On-line location

---

This document, and other related information, is on-line at:

<https://github.com/NCAR/CfRadial/docs>

## 1.3 Terminology – CfRadial1 and CfRadial2

---

We will refer to CfRadial 2.1, and future 2.x versions, collectively as CfRadial2.

The previous versions, 1.1 through 1.4, will be referred to collectively as CfRadial1.

## 1.4 History

---

CfRadial was introduced in 2011 as a format designed to store data from scanning weather radars and lidars in an accurate and lossless manner.

Since digital weather radars made their debut in the 1970s, a wide variety of data formats has emerged for pulsed instruments (radar and lidar) in polar coordinates. Researchers and manufacturers have tended to develop formats unique to their instruments, and since 1990 NCAR has supported over 20 different data formats for radar and lidar data. Researchers, students and operational users spend unnecessary time handling the complexity of these formats.

CfRadial grew out of the need to simplify the use of data from weather radars and lidars and thereby to improve efficiency. CfRadial adopts the well-known NetCDF framework, along with the Climate and Forecasting (CF) conventions. It is designed to accurately store the metadata and data produced by the instruments, in their native polar coordinates, without any loss of information. Mobile platforms are supported. Data field identification is facilitated by the

'standard\_name' convention in CF, so that fields derived from algorithms (such as hydrometeor type) can be represented just as easily as the original fields (such as radar reflectivity).

**Table 1.1: History of CfRadial versions**

Date	Version	Remarks
2011/02/01	1.1	First operational version. NetCDF classic data model.
2011/06/07	1.2	Minor changes / additions. NetCDF classic data model.
2013/07/01	1.3	Major changes / additions. NetCDF classic data model.
2016/08/01	1.4	Major additions – data quality, spectra. NetCDF classic data model.
2017/06/02	2.0 DRAFT	Major revision – not backward compatible with CfRadial1 Uses NetCDF 4 and groups. Combines CfRadial 1.4 and ODIM_H5 version 2.2.
2019/02/03	2.0 DRAFT	(a) ray_angle_res changed to ray_angle_resolution (b) fixed_angle changed to sweep_fixed_angle (c) Added sweep modes: doppler_beam_swinging, complex_trajectory and electronic_steering. <b>2.0 deprecated on 2019/09/01.</b>
<b>2019/09/01</b>	<b>2.1 DRAFT</b>	<b>A number of issues were raised related to draft version 2.0. The 2.1 draft addresses those issues. It is not backward compatible with version 2.0.</b>

CfRadial1 has been adopted by NCAR, as well as NCAS (the UK National Center for Atmospheric Science) and the US DOE Atmospheric Radiation Measurement (ARM) program. CfRadial development was boosted in 2015 through a two-year NSF EarthCube grant to improve CF in general. Following the CF user community workshop in Boulder Colorado in May 2016, version 1.4 was agreed upon, adding explicit support for quality fields and spectra.

**CfRadial2 has been proposed as an official exchange format for the WMO.**

## 1.5 CF2 and CfRadial2

In 2015 the US National Science Foundation (NSF) funded the project “EarthCube IA: Advancing NetCDF-CF for the Geoscience Community”. This project aims to update the CF conventions to make better use of NetCDF4 capabilities, such as groups, to organize data in CF-compliant files. CF2.0 standards will be developed to guide this process. The development of CfRadial2 is part of this work. A workshop was held in Boulder Colorado, in May 2016, to assess and collaborate on progress to Cf2.0.

Shortly after that, in July 2016, a WMO-sponsored meeting was held at NCAR in Boulder, during which the WMO Task Team on Weather Radar Data Exchange (TT-WRDE) considered the adoption of a single WMO-recommended format for radar and lidar data in polar coordinates. The two modern formats discussed as options were CfRadial and the European operational radar

community ODIM\_H5 (HDF5) format (Michelson et al. 2014), in addition to the older and more rigid table-driven BUFR and GRIB2 formats. TT-WRDE recommended that CfRadial 1.4 be merged with the sweep-oriented structure of ODIM\_H5, making use of groups to produce a single WMO format that will encompass the best ideas of both formats. That has led to the emergence of CfRadial2. This format should meet the objectives of both the NSF EarthCube CF 2.0 initiative and the WMO.

## 1.6 On-line URLs

---

This document, older versions, full history and other related information, are available on-line at:

<https://github.com/NCAR/CfRadial/tree/master/docs>

These include detailed documentation of versions 1.1 through 1.4, as well as the current CfRadial2 development.

The current NetCDF CF conventions are documented at:

<http://cfconventions.org/>

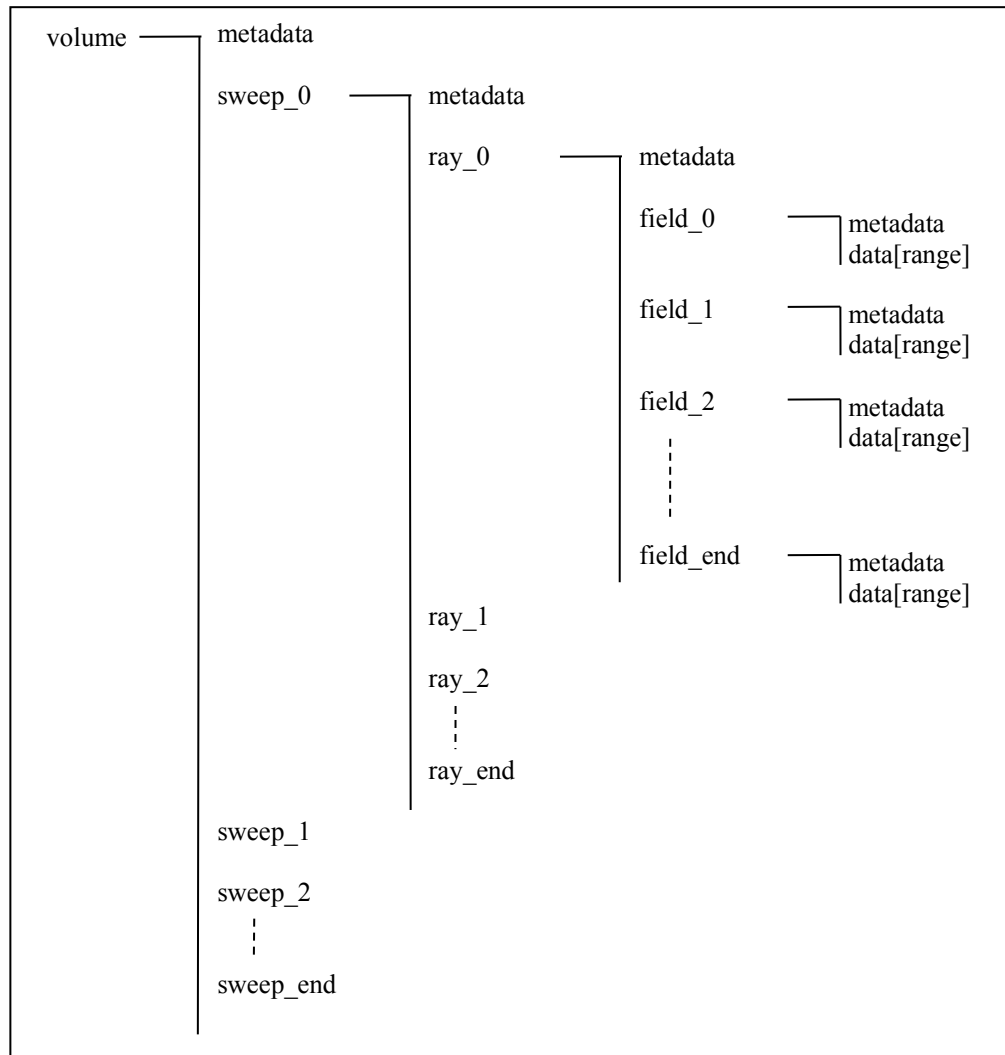
<http://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/cf-conventions.pdf>

## 2 Radar/Lidar Data Information Model

### 2.1 Logical organization of data in a volume, using sweeps and rays

Radars and lidars are pulsing instruments that either scan in polar coordinates, or stare in a fixed direction.

A **volume scan** (or simply a **volume**) is defined as a scanning sequence that repeats over time.



**Figure 2.1: logical data structure for a volume scan,  
using sweep and ray abstractions and 1-D data field arrays**

Figure 2.1 shows an idealized data model for a radar or lidar **volume**, representing the data fields as 1-D arrays on a ray object.

A **volume** consists of a series of 1 or more **sweeps** (defined below).



As a radar or lidar scans (or points), the data **fields** (commonly known as ‘moments’) are computed over limits specified by a time interval or angular interval.

We refer to this entity as a **ray**, beam or dwell. In this document we will use the term **ray**.

A **sweep** is a collection of **rays**, for which certain properties remain constant. Examples are:

- PPI 360-degree surveillance (target elevation angle constant)
- PPI sector (target elevation angle constant)
- RHI (target azimuth angle constant)
- time period for vertically pointing instrument (azimuth and elevation both constant)

The following *always* remain constant for all **rays** in a **sweep**:

- number of gates
- range geometry (range to each gate)
- sweep mode (*surveillance, sector, RHI, etc.*)
- target angle(s)

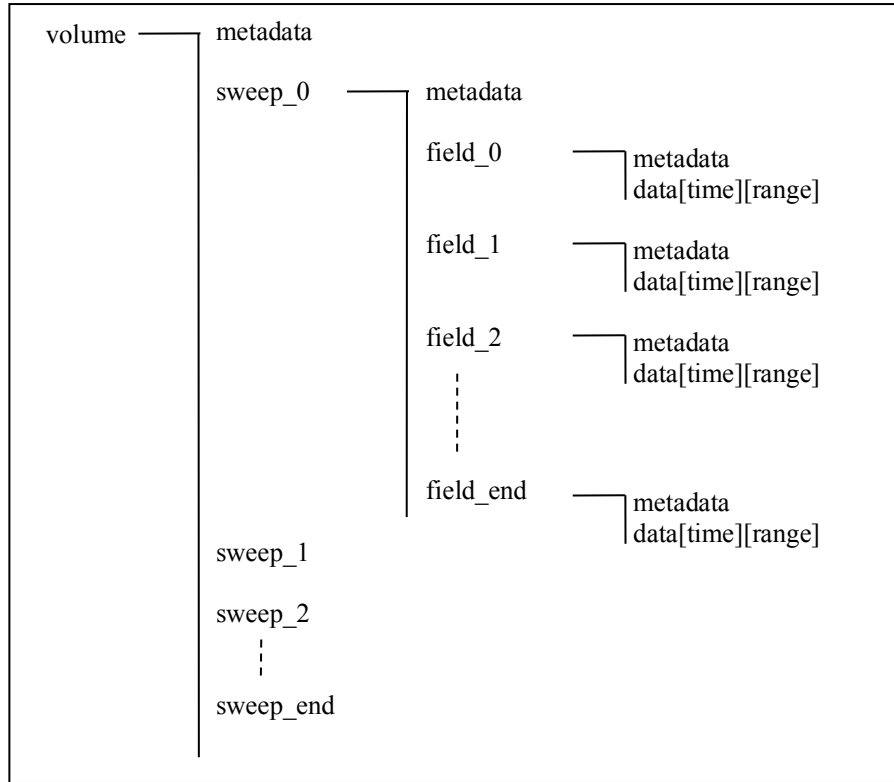
The following would *usually* remain constant for the **rays** in a **sweep**:

- nominal scan rate
- pulse width
- pulsing scheme
- Nyquist velocity
- data quality control procedures

For a given **ray**, the **field** (or moments) data are computed for a sequence of **ranges** increasing radially away from the instrument. These are referred to as range **gates**.

A **ray** contains a number of **fields**, with a value for each **field** at each **gate**. In the ray abstraction, fields are represented as 1-D arrays, with length **range**.

## 2.2 Logical organization of data in a volume, using sweeps with 2-D data

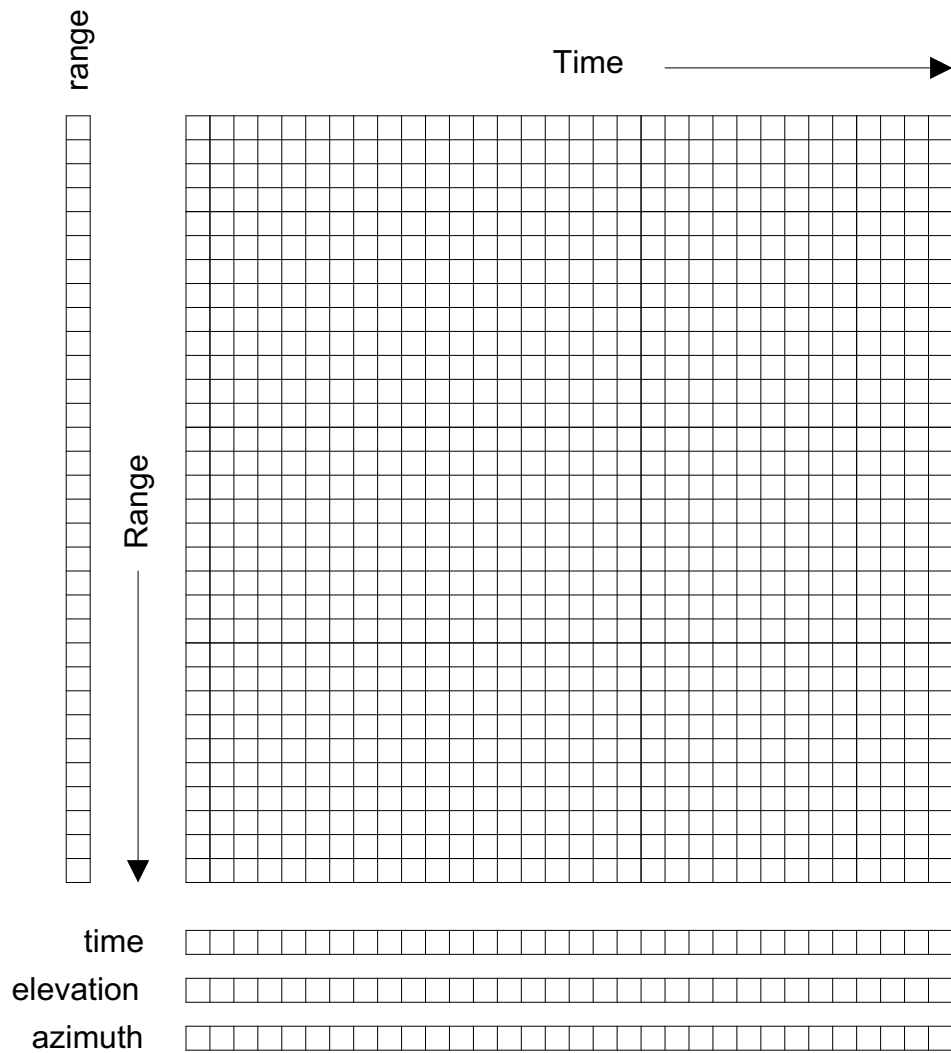


**Figure 2.2: logical data structure for a volume scan,  
using the sweep abstraction with 2-D data fields**

In contrast to Figure 2.1, Figure 2.2 shows a modified data model, in which the **sweeps** contain the **field** (moments) data directly, stored as 2-D arrays of [time][range]. This requires that the number of gates be constant for all rays in a sweep.

This is the data model that has been adopted for CfRadial2. Figure 2.3 below has more detail.

This representation has the advantage that the 2-D arrays per sweep simplify the data storage mechanism, and allow for more efficient data compression than do 1-D arrays per ray.



**Figure 2.3 Data field for a sweep, represented in time and range,  
with a constant number of range gates**

## 2.3 Field data byte representation

---

The field data will be stored using one of the following:

**Table 2.1: field data representation**

NetCDF type	Byte width	Description
signed char	1	scaled signed integer
unsigned char	1	scaled unsigned integer
signed short	2	scaled signed integer
unsigned short	2	scaled unsigned integer
signed int	4	scaled signed integer
unsigned int	4	scaled unsigned integer
signed long	8	scaled signed integer
signed long	8	scaled unsigned integer
float	4	floating point
double	8	floating point

For the integer types, the stored data values are interpreted as:

$$\text{data\_value} = (\text{integer\_value} * \text{scale\_factor}) + \text{add\_offset}.$$

The *scale\_factor* and *add\_offset* are provided as metadata attributes on the field.

## 2.4 Scanning modes

---

Scanning may be carried out in a number of different ways. For example:

- horizontal scanning at fixed elevation (PPI mode), sector or 360 degree surveillance
- vertical scanning at a constant azimuth (RHI mode)
- antenna stationary, i.e. constant elevation and azimuth (staring or pointing)
- aircraft radars which rotate around the longitudinal axis of the aircraft (e.g. ELDORA, NOAA Tail Radar)
- sun scanning in either PPI or RHI mode.

For each of these modes a **sweep** is defined as follows:

- PPI mode: a sequence of rays at a fixed elevation angle, but changing azimuth angles
- RHI mode: a sequence of rays at a fixed azimuth angle but changing elevation angles
- pointing mode: a sequence of rays over some time period, at fixed azimuth and elevation
- aircraft tail-type radars: a sweep starts at a rotation angle of 0 (antenna pointing vertically upwards) and ends 360 degrees later.

As the antenna transitions between sweeps, some rays may be recorded during the transition. For many radars, especially operational types, these rays are filtered out at the source. For some research radars, however, the rays are retained, and an **antenna\_transition flag** is set for these rays to allow them to be optionally filtered out at a later processing stage.

## 2.5 Geo-reference variables

---

Metadata variables in CfRadial are used to locate a radar or lidar measurement in space.

These are:

- range
- elevation
- azimuth
- latitude
- longitude
- altitude

See sections 4.3 and 5.4 for details on these variables.

For moving platforms, extra variables are required for geo-referencing. These are:

- heading
- roll
- pitch
- rotation
- tilt

See section 5.4 for details on these variables.

The mathematical procedures for computing data location relative to earth coordinates are described in detail in section 9.

## 3 Structure of CfRadial2 using NetCDF4 with groups

---

### 3.1 Group-based data structure

---

In order to be readily accessible to scientists in the user community using legacy applications, CfRadial1 uses the flat classic model of NetCDF3. A drawback of this approach is that the implementation quickly becomes complicated, since much of the metadata is placed at the top level in the data structure. Namespace clashes can easily occur.

By contrast, the European ODIM\_H5 format makes extensive use of HDF5 groups to provide logical separation between data at different levels in the structure. In fact ODIM tends to rather over-use this approach, leading to complexity of a different type.

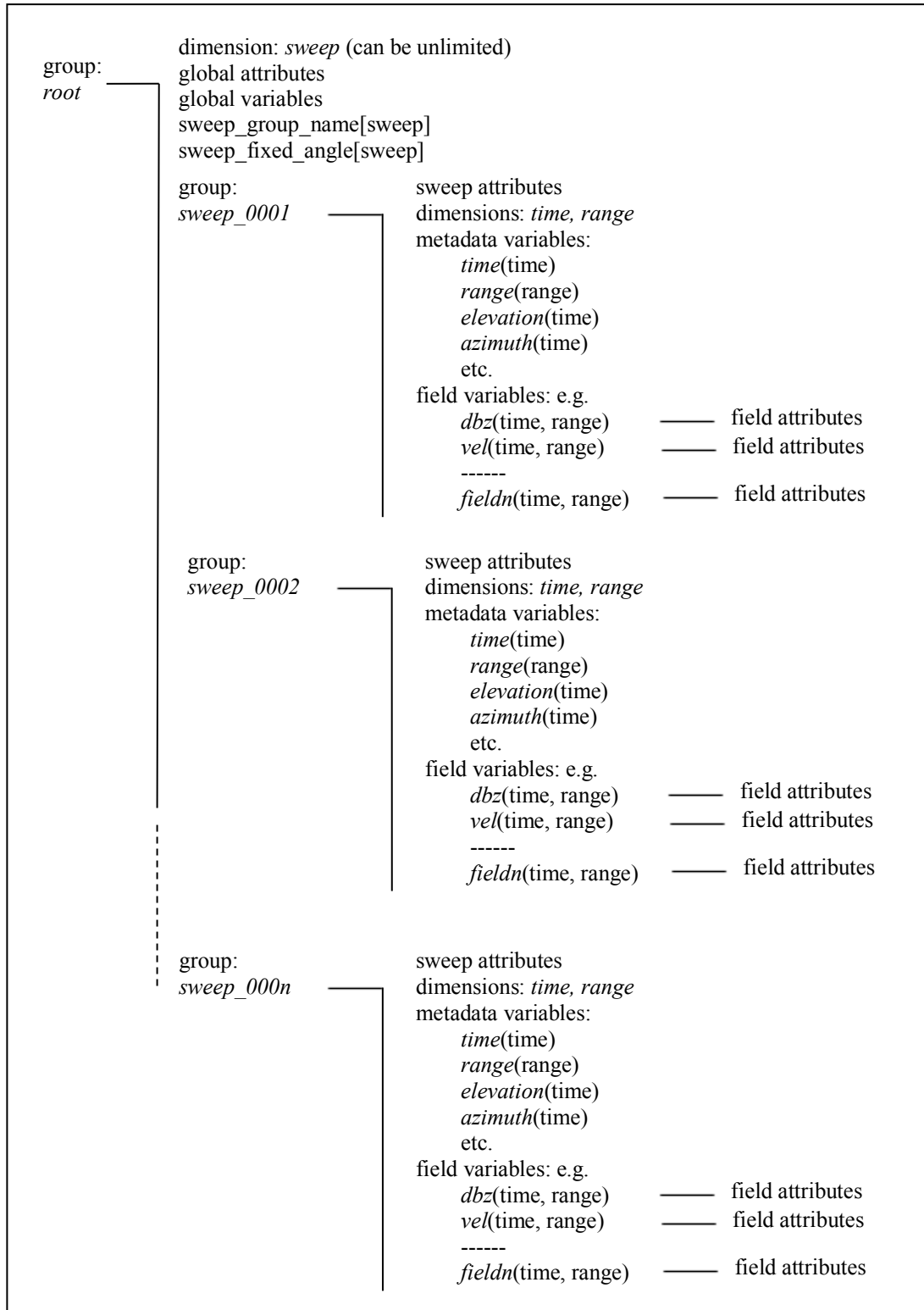
In designing CfRadial2, we chose to merge CfRadial1 and ODIM\_H5, adopting the best features of each to produce a clear implementation that is as simple as reasonable, but no simpler. CfRadial2 makes use of the group capability available in NetCDF4.

CfRadial2 adopts the sweep-based model of Figure 2.2. The overall structure is shown in Figure 3.1 below.

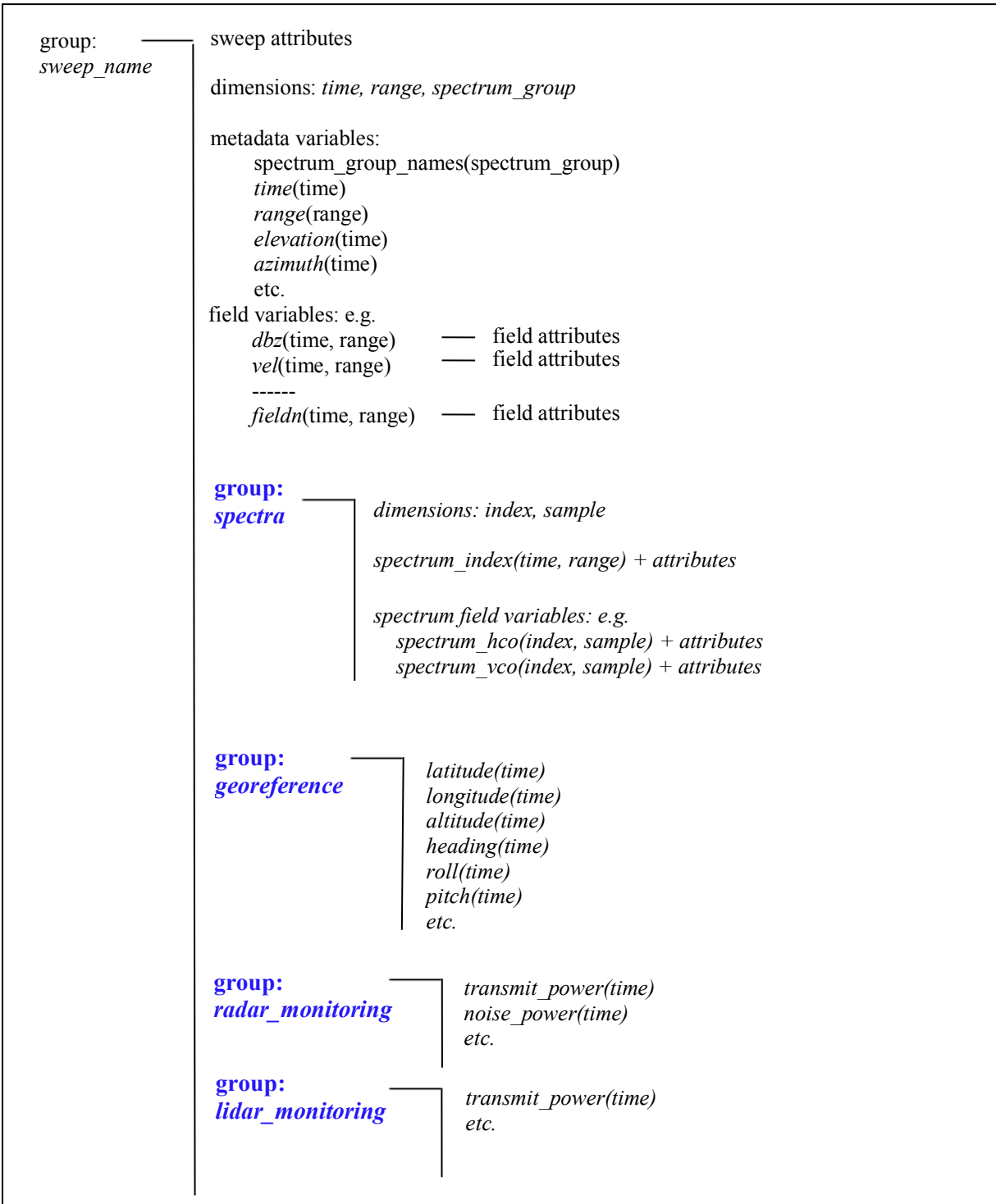
The top-level (default) root group holds the global dimensions, attributes and variables. Nested below this group is a sub-group for each sweep. The name of the sweep sub-groups is provided in a string array named 'sweep\_group\_name', also at the top level. This allows the user to locate the sweep groups directly.

CfRadial2 supports moving platforms (aircraft, ships and vehicles), which requires storing the georeference data accurately at each ray time. Furthermore, storage of spectra on a gate-by-gate basis, for example for vertically-pointing precipitation radars, is supported. Figure 3.2 shows these details. Both of these are specializations, and are not required for most fixed operational radars.

A number of optional groups are available in the root group, to support radar and lidar parameters, calibrations, and corrections to the georeferenced data. These are shown in figure 3.3.



**Figure 3.1: Group structure showing top-level dimensions, attributes, variables and sweep groups**



**Figure 3.2: Sweep group structure in more detail, showing support for geo-reference metadata for moving platforms, spectra, and monitoring data. Optional groups are in blue.**



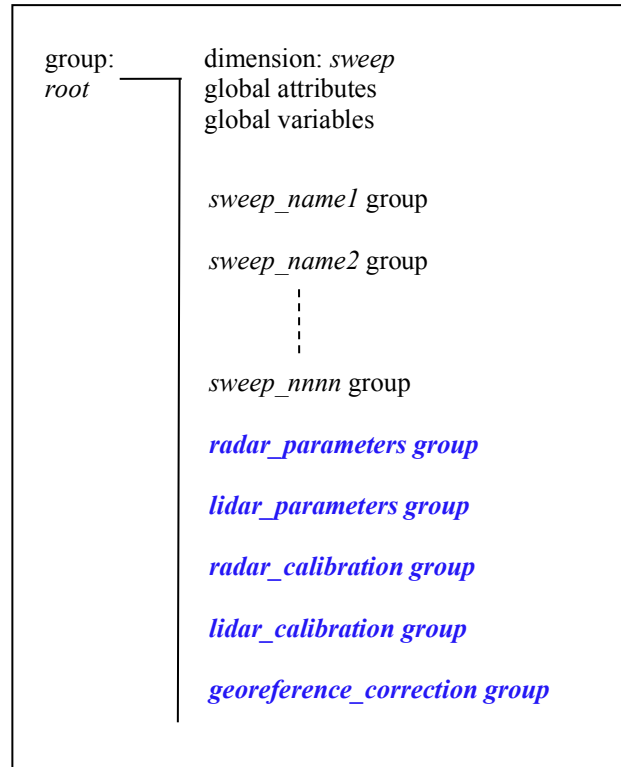


Figure 3.3: Optional metadata groups (in blue) in the root group

## 3.2 Principal dimensions and coordinate variables

The principal dimensions for data in a sweep are **time** and **range**. In CF terminology these are referred to as *coordinate variables*, which must have the same name for both the dimension and the variable.

The primary coordinate is **time** and the secondary coordinate is **range**.

The length of the **time** coordinate indicates the number of rays in the sweep.

The length of the **range** coordinate indicates the number of gates for the rays in the sweep.

The **time(time)** coordinate variable stores the double precision time of each ray, in seconds, from a reference time, which is normally the start of the volume (**time\_coverage\_start**). The units attribute for time indicates the time from which it is referenced.

The **range(range)** coordinate variable stores the range to the center of each gate. All rays in the sweep must have the same range geometry.

The **elevation(time)** coordinate variable stores the elevation angle for each ray.

The **azimuth(time)** coordinate variable stores the azimuth angle for each ray.

The data fields are stored as 2-D arrays, with dimensions (**time, range**).

### 3.3 **\_FillValue and missing\_value attributes for data fields**

---

CF 1.6 states that the use of **missing\_value** is deprecated, and that only **\_FillValue** should be used.

For CfRadial2, **\_FillValue** is preferred. However, **missing\_value** may be used.

Only one or the other should be specified, not both.

Applications reading CfRadial data should check for both of these attributes.

NetCDF 4 is built on HDF5, which supports compression. Where data are missing or unusable, the data values will be set to a constant well-known **\_FillValue** (or **missing\_value**) code. This facilitates efficient compression.

### 3.4 **Required vs. optional variables**

---

Required variables are shown shaded in this document.

All other variables are optional.

If an optional variable is not provided, reader applications should set the variable to a missing value as appropriate.

### 3.5 **Grid mapping variable – radar\_lidar\_radial\_scan**

---

The data in this format is saved in the native coordinate system for radars and lidars, i.e. radial (or polar) coordinates, with the instrument at the origin.

In order to properly support this type of data, the *radar\_lidar\_radial\_scan* grid mapping has been introduced to the CF Conventions:

The mapping parameters are:

- *latitude\_of\_projection\_origin*
- *longitude\_of\_projection\_origin*
- *height\_of\_projection\_origin*

The relevant coordinate variables are as follows:

Coordinate variable	Standard name	Auxiliary reference
range (line-of-sight slant range)	line_of_sight_distance_from_instrument	
azimuth	sensor_to_target_azimuth_angle	Auxiliary coordinate on the time dimension
elevation	sensor_to_target_elevation_angle	Auxiliary coordinate on the time dimension

The relevant grid mapping attributes are:

Attribute	Type	Description
height_of_projection_origin	N	Records the height, in meters, of the map projection origin point above the ellipsoid (or sphere). Used by radial scan type projections to indicate the altitude of the sensor to which the polar coordinate system is referenced.

A general description of radar projection is given in Doviak[1984], page 13, equation 2.28b. Note: there is no corresponding projection in PROJ.4.

### 3.6 Extensions to the CF convention

This convention requires the following extensions to CF:

- The following axis attribute types:
  - axis = "radial\_azimuth\_coordinate";
  - axis = "radial\_elevation\_coordinate";
  - axis = "radial\_range\_coordinate";
- For CfRadial to follow CF properly, support for the following must be added to the supported units:
  - dB (ratio of two values, in log units. For example, ZDR).
  - dBm (power in milliwatts, in log units)
  - dBZ (radar reflectivity in log units)
- Additional standard names – see section 8.

Given the above extensions, CfRadial2 files will be CF2 compliant.

**NOTE on units:** in the following tables, for conciseness, we do not spell out the **units** strings exactly as they are in the NetCDF file. Instead, the following abbreviations are used:

Units string in NetCDF file	Abbreviation in tables
“degrees per second”	degrees/s
“meters per second”	m/s

### 3.7 String types

In NetCDF 4, strings may be represented either as an array of chars (NC\_CHAR), or as a string (NC\_STRING).

In CfRadial2, all string *variables* must be of type NC\_STRING. Use of full string types means that no dimension is required for the string length.

String *attributes* may be of type NC\_STRING or NC\_CHAR. **NC\_STRING** is the preferred option.

In C, a string attribute is written using the NetCDF API function `nc_put_att_string`, whereas a char array attribute is written using the `nc_put_att_text` function.

## 4 Root group

---

The following sections present the details of the information in the top-level (root) group of the data set.

NOTE that in the tables below, shading indicates required items. The non-shaded items are optional.

### 4.1 Global attributes

---

Attribute name	Type	Convention	Description
Conventions	string	CF	Conventions string will specify “Cf/Radial”.
version	string	CF/Radial	CF/Radial version number (“2.0”)
title	string	CF	Short description of file contents
institution	string	CF	Where the original data were produced
references	string	CF	References that describe the data or the methods used to produce it
source	string	CF	Method of production of the original data
history	string	CF	List of modifications to the original data
comment	string	CF	Miscellaneous information
instrument_name	string	CF/Radial	Name of radar or lidar
site_name	string	CF/Radial	Name of site where data were gathered
scan_name	string	CF/Radial	Name of scan strategy used, if applicable
scan_id	int	CF/Radial	Scan strategy id, if applicable. Assumed 0 if missing.
platform_is_mobile	string	CF/Radial	“true” or “false” Assumed “false” if missing.
ray_times_increase	string	CF/Radial	“true” or “false” Assumed “true” if missing. This is set to true if ray times increase monotonically throughout all of the sweeps in the volume.

Attribute name	Type	Convention	Description
field_names	string[]	CF/Radial	Array of strings of field names present in this file.
time_coverage_start	string	CF/Radial	Copy of time_coverage_start global variable in 4.3.
time_coverage_end	string	CF/Radial	Copy of time_coverage_end global variable in 4.3.
simulated data	string	ODIM	“true” or “false” Assumed “false” if missing. Data in this file are simulated

## 4.2 Global Dimensions

---

Dimension name	Description
sweep	The number of sweeps in the dataset

## 4.3 Global variables

---

Variable name	Dimension	Type	Units	Comments
volume_number		int		Volume numbers are sequential, relative to some arbitrary start time, and may wrap.
platform_type		string		Options are: “fixed”, “vehicle”, “ship”, “aircraft”, “aircraft_fore”, “aircraft_aft”, “aircraft_tail”, “aircraft_belly”, “aircraft_roof”, “aircraft_nose”, “satellite_orbit”, “satellite_geostat” Assumed “fixed” if missing.
instrument_type		string		Options are: “radar”, “lidar” Assumed “radar” if missing.

Variable name	Dimension	Type	Units	Comments
primary_axis		string		Options are: <i>“axis_z”, “axis_y”, “axis_x”,  “axis_z_prime”, “axis_y_prime”,  “axis_x_prime”.</i> See section 9 for details. Assumed “axis_z” if missing.
time_coverage_start		double	ISO8601 string: “seconds since YYYY-MM-DD HH:MM:SS” or “seconds since YYYY-MM-DDTHH:MM:SS”	UTC time of first ray in file.
time_coverage_end		double	ISO8601 string: “seconds since YYYY-MM-DD HH:MM:SS” or “seconds since YYYY-MM-DDTHH:MM:SS”	UTC time of last ray in file.
latitude		double	degrees_north	Latitude of instrument, <b>using WGS84.</b> For a mobile platform, this is a latitude at the start of the volume.
longitude		double	degrees_east	Longitude of instrument, <b>using WGS84</b> For a mobile platform, this is the longitude at the start of the volume.
altitude		double	meters	Altitude of instrument, above mean sea level, <b>using WGS84 and EGM2008 geoid corrections.</b> For a scanning radar, this is the center of rotation of the antenna. For a mobile platform, this is the altitude at the start of the volume.
altitude_agl		double	meters	Altitude of instrument above ground level. Omit if not known.
sweep_group_name	(sweep)	string		Array of names for sweep groups. Allows the user to locate the sweep groups directly.
sweep_fixed_angle	(sweep)	float	degrees	Array of fixed angles for sweeps. This summarizes the fixed angles for all of the sweeps, so that a user does not need to read individual sweep groups to determine the fixed angles. The value should be copied from the sweep_fixed_angle attribute in the sweep groups.

Variable name	Dimension	Type	Units	Comments
status_str		string		General-purpose string for storing any information that is not included in other parts of the data structure. Format can be simple text, XML, JSon, etc.



## 5 Sweep groups

This section provides details of the information in each sweep group.

The name of the sweep groups is found in the *sweep\_group\_name* array variable in the root group.

NOTE that in the tables below, shading indicates required items. The non-shaded items are optional.

### 5.1 Sweep-specific Dimensions

Dimension name	Description
time	The number of rays.
range	The number of range bins
prt	Number of prts used in pulsing scheme. Optional for fixed, staggered or dual Required for more complicated schemes.
spectrum_groups	Number of spectrum groups in this sweep
frequency	Number of frequencies used

### 5.2 Sweep coordinate variables

Variable name	Dimension	Type	Units	Comments
time	(time)	double	ISO8601 string: "seconds since YYYY-MM-DD HH:MM:SS" or "seconds since YYYY-MM-DDTHH:MM:SS"	Coordinate variable for time. Time at center of each ray.
range	(range)	float	meters	Coordinate variable for range. Range to center of each bin.

#### 5.2.1 Attributes for time coordinate variable

Attribute name	Type	Value
standard_name	string	"time"
units	string	ISO8601 string "seconds since YYYY-MM-DDThh:mm:ssZ", where the actual reference time values are used.

Attribute name	Type	Value
calendar	string	Defaults to “gregorian” if missing. Options are: “gregorian” or “standard”, “proleptic_gregorian”, “no leap” or “365_day”, “all_leap” or “366_day”, “360_day”, “julian” See CF conventions for details.

### 5.2.2 Attributes for range coordinate variable

Attribute name	Type	Value
standard_name	string	“projection_range_coordinate”
long_name	string	“range to measurement volume”
units	string	“meters”
spacing_is_constant	string	“true” or “false”
meters_to_center_of_first_gate	float	Start range in meters.
meters_between_gates	float	Gate spacing in meters. Required if spacing_is_constant is “true”. Not applicable otherwise.
axis	string	“radial_range_coordinate”

### 5.3 Sweep variables

Variable name	Dimension	Type	Units	Comments
sweep_number		int		The number of the sweep, in the volume scan. 0-based.

Variable name	Dimension	Type	Units	Comments
sweep_mode		string		Options are: <i>“sector”, “coplane”, rhi”,  “vertical pointing”, “idle”,  “azimuth_surveillance”,  “elevation_surveillance”,  “sunscan”, “pointing”,  “manual ppi”, “manual rhi”,  “doppler_beam_swinging”,  “complex_trajectory”,  “electronic_steering”</i>
follow_mode		string		options are: <i>“none”, “sun”,  “vehicle”, “aircraft”, “target”,  “manual”</i> Assumed <i>“none”</i> if missing.
prt_mode		string		Pulsing mode Options are: <i>“fixed”,  “staggered”, “dual”,  “hybrid”</i> . Assumed <i>“fixed”</i> if missing. May also be more complicated pulsing schemes, such as HHVV, HHVVH etc.
frequency	(frequency)	float	s-1	List of operating frequencies, in Hertz. In most cases, only a single frequency is used.
polarization_mode		string		Options are: <i>“horizontal”,  “vertical”, “hv_alt”,  “hv_sim”, “circular”</i> Assumed <i>“horizontal”</i> if missing.
polarization_sequence	(prt)	string		This only applies if prt_mode is set to “hybrid”. As an example, the form of it would be [‘H’, ‘H’, ‘V’, ‘V’, ‘H’] for HHVVH pulsing.
sweep_fixed_angle		float	degrees	Target angle for the sweep. elevation in most modes azimuth in RHI mode

Variable name	Dimension	Type	Units	Comments
rays_are_indexed		string		“true” or “false” Indicates whether or not the ray angles (elevation in RHI mode, azimuth in other modes) are indexed to a regular grid.
ray_angle_resolution		float	degrees	If rays_are_indexed is “true”, this is the resolution of the angular grid – i.e. the delta angle between successive rays.
qc_procedures		string		Documents QC procedures per sweep.
target_scan_rate		float	degrees/s	Intended scan rate for this sweep. The actual scan rate is stored according to section 4.8. This variable is optional. Omit if not available.
scan_rate	(time)	float	degrees/s	Actual antenna scan rate. Set to negative if counter-clockwise in azimuth or decreasing in elevation. Positive otherwise.
azimuth	(time)	float	degrees	Azimuth of antenna, relative to true north. The azimuth should refer to the center of the dwell.
elevation	(time)	float	degrees	Elevation of antenna, relative to the horizontal plane. The elevation should refer to the center of the dwell.
antenna_transition	(time)	byte		1 if antenna is in transition, i.e. between sweeps, 0 if not. If variable is omitted, the transition will be assumed to be 0 everywhere. Assumed 0 if missing.
pulse_width	(time)	float	seconds	
calib_index	(time)	int		Index for the radar calibration that applies to this pulse width. See section 7.3.

Variable name	Dimension	Type	Units	Comments
rx_range_resolution	(time)	float	meters	Resolution of the raw receiver samples. If missing, assumed to be meters_between_gates (5.2.2). Raw data may be resampled before data storage.
prt	(time)	float	seconds	Pulse repetition time. For staggered prt, also see prt_ratio.
prt_ratio	(time)	float		Ratio of prt/prt2. For dual/staggered prt mode.
prt_sequence	(time, prt)	float	seconds	Sequence of prts used. Optional for fixed, staggered and dual, which can make use of 'prt' and 'prt_ratio'. Required for more complicated pulsing schemes.
nyquist_velocity	(time)	float	m/s	Unambiguous velocity. This is the effective nyquist velocity after unfolding. See also the field-specific attributes fold_limit_lower and fold_limit_upper, 5.6.
unambiguous_range	(time)	float	meters	Unambiguous range
n_samples	(time)	int		Number of samples used to compute moments
spectrum_group_names	(spectrum_group)	string		Array of names of spectrum groups.

The number of samples used to compute the moments may vary from field to field. In the table above, n\_samples refers to the maximum number of samples used for any field. The field attribute 'sampling\_ratio' (see 5.6) is the actual number of samples used for a given field, divided by n\_samples. It will generally be 1.0, the default.

### 5.3.1 Attributes for azimuth(time) variable

Attribute name	Type	Value
standard_name	string	"ray_azimuth_angle"
long_name	string	"azimuth angle from true north"
units	string	"degrees"

Attribute name	Type	Value
axis	string	“radial_azimuth_coordinate”

### 5.3.2 Attributes for elevation(time) variable

Attribute name	Type	Value
standard_name	string	“ray_elevation_angle”
long_name	string	“elevation angle from horizontal plane”
units	string	“degrees”
axis	string	“radial_elevation_coordinate”

## 5.4 The georeference sub-group

For mobile platforms, this sub-group will be included in each sweep group, to store the metadata for platform position, pointing and velocity.

This group will always be named ‘*georeference*’.

Variable name	Dimension	Type	Units	Comments
latitude	(time)	double	degrees_north	Latitude of instrument. WGS84.
longitude	(time)	double	degrees_east	Longitude of instrument. WGS84.
altitude	(time)	double	meters	Altitude of instrument, above mean sea level. WGS84 with EGM2008 geoid corections. For a scanning radar, this is the center of rotation of the antenna.
heading	(time)	float	degrees	Heading of the platform relative to true N, looking down from above.
roll	(time)	float	degrees	Roll about longitudinal axis of platform. Positive is left side up, looking forward.
pitch	(time)	float	degrees	Pitch about the lateral axis of the platform. Positive is up at the front.

Variable name	Dimension	Type	Units	Comments
drift	(time)	float	degrees	Difference between heading and track over the ground. Positive drift implies track is clockwise from heading, looking from above. NOTE: not applicable to land-based mobile platforms.
rotation	(time)	float	degrees	Angle between the radar beam and the vertical axis of the platform. Zero is along the vertical axis, positive is clockwise looking forward from behind the platform.
tilt	(time)	float	degrees	Angle between radar beam (when it is in a plane containing the longitudinal axis of the platform) and a line perpendicular to the longitudinal axis. Zero is perpendicular to the longitudinal axis, positive is towards the front of the platform.
eastward_velocity	(time)	float	m/s	EW velocity of the platform. Positive is eastwards.
northward_velocity	(time)	float	m/s	NS velocity of the platform. Positive is northwards.
vertical_velocity	(time)	float	m/s	Vertical velocity of the platform. Positive is up.
eastward_wind	(time)	float	m/s	EW wind at the platform location. Positive is eastwards.
northward_wind	(time)	float	m/s	NS wind at the platform location. Positive is northwards.
vertical_wind	(time)	float	m/s	Vertical wind at the platform location. Positive is up.
heading_rate	(time)	float	degrees/s	Rate of change of heading
roll_rate	(time)	float	degrees/s	Rate of change of roll of the platform
pitch_rate	(time)	float	degrees/s	Rate of change of pitch of the platform.

Variable name	Dimension	Type	Units	Comments
georefs_applied	(time)	byte		1 if georeference information for mobile platforms has been applied to correct the azimuth and elevation. 0 otherwise. Assumed 0 if missing.

## 5.5 The *monitoring* sub-groups

If monitoring data is available, this monitoring sub-groups will be included in each relevant sweep group, to store the monitoring variables.

The groups will be named '*radar\_monitoring*' and '*lidar\_monitoring*' depending on the instrument type. The variables included will be dependent on the particular instrument.

Variable name	Dimension	Type	Units	Comments
measured_transmit_power_h	(time)	float	dBm	Measured transmit power H polarization
measured_transmit_power_v	(time)	float	dBm	Measured transmit power V polarization
measured_sky_noise	(time)	float	dBm	Noise measured at the receiver when connected to the antenna with no noise source connected.
measured_cold_noise	(time)	float	dBm	Noise measured at the receiver when connected to the noise source, but it is not enabled.
measured_hot_noise	(time)	float	dBm	Noise measured at the receiver when it is connected to the noise source and the noise source is on.
phase_difference_transmit_hv	(time)	float	degrees	Phase difference between transmitted horizontally and vertically-polarized signals as determined from the first valid range bins
antenna_pointing_accuracy_elev	(time)	float	degrees	Antenna-pointing accuracy in elevation



Variable name	Dimension	Type	Units	Comments
antenna_pointing_accuracy_az	(time)	float	degrees	Antenna-pointing accuracy in azimuth
calibration_offset_h	(time)	float	dB	Calibration offset for the horizontal channel
calibration_offset_v	(time)	float	dB	Calibration offset for the vertical channel
zdr_offset	(time)	float	dB	ZDR offset (bias)

Radar monitoring variables.

## 5.6 Field data variables

The field variables will be 2-dimensional arrays, with the dimensions **time** and **range**.

The field data will be stored using one of the following:

NetCDF type	Byte width	Description
NC_UBYTE	1	scaled unsigned integer
NC_BYTE	1	scaled signed integer
NC_USHORT	2	scaled unsigned integer
NC_SHORT	2	scaled signed integer
NC_UINT	4	scaled unsigned integer
NC_INT	4	scaled signed integer
NC_UINT64	8	scaled unsigned integer
NC_INT64	8	scaled signed integer
NC_FLOAT	4	floating point
NC_DOUBLE	8	floating point

The NetCDF variable name is interpreted as the short name for the field.

Field data variables have the following attributes:

Attribute name	Type	Convention	Description
standard_name	string	CF	CF standard name for field. See section 8.

Attribute name	Type	Convention	Description
long_name	string	CF	Long name describing the field. Any string is appropriate. Although this is an optional attribute, its use is strongly encouraged.
Units	string	CF	Units for field
_FillValue (or missing_value)	same type as field data	CF	Indicates data are missing at this range bin. Use of _FillValue is preferred. Only use one or the other.
_Undetect	same as field data	ODIM	Indicates an area (range bin) that has been radiated but has not produced a valid echo
scale_factor	float	CF	Float value = (integer value) * scale_factor + add_offset Only applies to integer types. Section 5.6.1.
add_offset	float	CF	Float value = (integer value) * scale_factor + add_offset Only applies to integer types. Section 5.6.1.
coordinates	string	CF	Section 5.6.2.
sampling_ratio	float	CF/Radial	Number of samples for this field divided by n_samples (see section 5.3). Assumed 1.0 if missing.
is_discrete	string	CF/Radial	“true” or “false” If “true”, this indicates that the field takes on discrete values, rather than floating point values. For example, if a field is used to indicate the hydrometeor type, this would be a discrete field.

Attribute name	Type	Convention	Description
field_folds	string	CF/Radial	<p>“true” or “false”</p> <p>Used to indicate that a field is limited between a min and max value, and that it folds between the two extremes. This typically applies to such fields as radial velocity and PHIDP.</p>
fold_limit_lower	float	CF/Radial	If field_folds is “true”, this indicates the lower limit at which the field folds.
fold_limit_upper	float	CF/Radial	If field_folds is “true”, this indicates the upper limit at which the field folds.
is_quality_field	string	CF/Radial	<p>“true” or “false”</p> <p>“true” indicates this is a quality control field.</p> <p>If the attribute is not present, defaults to “false”.</p>
flag_values	array of same type as field data	CF	Array of flag values. These values have special meaning, as documented in flag_meanings.
flag_meanings	string[]	CF	Meaning of flag_values or flag_masks.
flag_masks	array of same type as field data	CF	<p>Valid bit-wise masks used in a flag field that is comprised of bit-wise combinations of mask values.</p> <p>See flag_meanings.</p>
qualified_variables	string[]	CF/Radial	<p>Applicable if is_quality_field is “true”.</p> <p>Array list of variables that this variable qualifies.</p> <p>Every field variable in this list should list this variable in its ancillary_variable attribute.</p>

Attribute name	Type	Convention	Description
ancillary_variables	string[]	CF	Array list of variables to which this variable is related. In particular, this is intended to list the variables that contain quality information about this field. In that case, the quality field will list this field in its qualified_variable attribute.
thresholding_xml	string	CF/Radial	Thresholding details. Supplied if thresholding has been applied to the field. This should be in self-descriptive XML. (See 5.6.6.)
legend_xml	string	CF/Radial	Legend details. Applies to discrete fields. Maps field values to the properties they represent. This should be in self-descriptive XML. (See 5.6.7.) Not sure what to do here.

### 5.6.1 Use of scale\_factor and add\_offset

---

scale\_factor and add\_offset are required for nbyte, short and int fields. They are not applicable to float and double fields.

$$\text{float\_value} = (\text{integer\_value} * \text{scale\_factor}) + \text{add\_offset}$$

### 5.6.2 Use of coordinates attribute

---

The “coordinates” attribute lists the variables needed to compute the location of a data point in space.

For stationary platforms, the coordinates attribute should be set to:

***“elevation azimuth range”***

For mobile platforms, the coordinates attribute should be set to:

***“elevation azimuth range heading roll pitch rotation tilt”***

### 5.6.3 Use of flag values - optional

---

For all data sets, the **\_FillValue** attribute has special meaning – see 3.5.

A field variable may make use of more than one reserved value, to indicate a variety of conditions. For example, with radar data, you may wish to indicate that the beam is blocked for a

given gate, and that no echo will ever be detected at that gate. That provides more information than just using **\_FillValue**.

The **flag\_values** and **flag\_meanings** attributes can be used in this case.

The **flag\_values** attribute is a list of values (other than **\_FillValue**) that have special meanings. It should have the same type as the variable.

The **flag\_meanings** string attribute is an array of strings that indicate the meanings of each of the **flag\_values**. If multi-word meanings are needed, use underscores to connect the words. For example you might use flag meanings of 'no\_coverage' and 'low\_snr' to distinguish between regions where the radar cannot see as opposed to regions where the signal is well below the noise.

#### 5.6.4 Flag mask fields - optional

---

An integer-type field variable may contain values that describe a number of independent Boolean conditions. The field is constructed using the bit-wise OR method to combine the conditions.

In this case, the **flag\_mask** and **flag\_meanings** attributes are used to indicate the valid values in the field, and the meanings.

The **flag\_masks** attribute is a list of integer values (other than **\_FillValue**) that are bit-wise combinations valid for the field variable. It should have the same type as the variable.

The **flag\_meanings** string attribute is an array of strings that indicate the meanings of each of the **flag\_masks**. If multi-word meanings are needed, use underscores to connect the words.

ODIM note - this mechanism can be used to indicate various conditions, such as no-echo etc. It is cleaner and more flexible than overloading the actual values in the data.

#### 5.6.5 Quality control fields - optional

---

Some field variables exist to provide quality information about another field variable. For example, one field may indicate the uncertainty associated with another field.

In this case, the field should have the **is\_quality** string attribute, with the value set to "true". If this attribute is missing, it is assumed to be "false".

In addition, the field should have the **qualified\_variables** string attribute. This is an array of field names that this field qualifies.

Each qualified field, in turn, should have the **ancillary\_variables** string attribute. This is an array of fields that qualify it.

#### 5.6.6 Thresholding XML

---

The **thresholding\_xml** should contain self-explanatory information about any thresholding that has been applied to the data field, as in the following example:

```
<thresholding field="DBZ">
  <field_used>
    <name>NCP</name>
    <min_val>0.15</min_val>
```

```
</field_used>
<field_used>
  <name>SNR</name>
  <min_val>-3.0</min_val>
</field_used>
<note>NCP only checked if DBZ > 40</note>
</thresholding>
```

### 5.6.7 Legend XML

---

The `legend_xml` should contain self-explanatory information about the categories for a discrete field, as in the following example for particle type:

```
<legend label="particle_id">
  <category>
    <value>1</value>
    <label>cloud</label>
  <category>
  <category>
    <value>2</value>
    <label>drizzle</label>
  <category>
    .....
    .....
  <category>
    <value>17</value>
    <label>ground_clutter</label>
  <category>
</legend>
```

## 6 Spectrum groups

---

Because spectra are potentially voluminous, they are stored in a sparse array, such that a spectrum need only be stored for gate locations of interest, rather than at all locations. (For example, we may not store spectra for gates with low SNR.) This is achieved through the use of an integer ‘spectrum index’ variable, which exists for every gate location. If the index is set to -1, no spectrum is stored for that gate. If the index is 0 or positive, it indicates the location of the spectrum in the spectrum data array.

### 6.1 Spectrum-specific information on sweep

---

If spectra are stored in a sweep, the sweep will define the following:

dimension *spectrum\_group* – the number of spectrum groups in the sweep;

variable *spectrum\_group\_names(spectrum\_group)* – the names of the spectrum groups.

This array will allow the user to locate the spectrum groups in the sweep.

### 6.2 *spectrum\_index* variable

---

Each spectrum group will contain a single *spectrum\_index* variable.

*spectrum\_index* is an integer variable, stored as a regular field data variable as specified in section 5.6 above. There is a spectrum index value for every gate in the sweep. The dimensions are:

*spectrum\_index*(time, range)

just as for all field variables.

The *\_FillValue* attribute is set to -1.

If the index is missing for a gate, this indicates that there is no spectrum stored for that gate.

If the index is not missing for a gate, it will be set to a value between 0 and (index – 1). This value indicates the location for the spectrum in the spectrum field data variable array.

### 6.3 Spectrum group dimensions

---

Dimension name	Description
index	Number of spectra stored in each spectrum variable in this group. This dimensions the number of indices used to locate the spectra, i.e. for the gates where the index is not -1.
sample	Number of samples in the spectra.

## 6.4 Spectrum field variables

---

A spectrum variable will have the dimensions: (index, sample).

As with moments field variables, a spectrum variable will be stored using one of the following:

NetCDF type	Byte width	Description
ncbyte	1	scaled signed integer
short	2	scaled signed integer
int	4	scaled signed integer
float	4	floating point
double	8	floating point

## 6.5 Spectrum field attributes

---

Spectrum fields have the following attributes:

Attribute name	Type	Convention	Description
long_name	string	CF	Long name describing the field. Any string is appropriate. Although this is an optional attribute, its use is strongly encouraged.
standard_name	string	CF	CF standard name for field, or from section 8.
units	string	CF	Units for field
_FillValue or missing_value	same type as field data	CF	Indicates data are missing for a given sample. _FillValue is preferred.
scale_factor	float	CF	See section 5.6. Float value = (integer value) * scale_factor + add_offset. Only applied to integer types.
add_offset	float	CF	See section 5.6. Float value = (integer value) * scale_factor + add_offset. Only applied to integer types.
coordinates	string	CF	See section 5.6.2.



Attribute name	Type	Convention	Description
fft_length	int	CF/Radial	Length used to compute this spectrum
block_avg_length	int	CF/Radial	Number of block spectra averaged for each output spectrum.

## 7 Root group metadata groups

---

The base *CF/Radial* convention, as described above, covers the minimum set of NetCDF elements which are required to locate radar/lidar data in time and space.

Additional groups may be included to provide metadata on other aspects of the system.

These groups reside in the root group, and are optional.

### 7.1 The *radar\_parameters* sub-group

---

This group holds radar parameters specific to a radar instrument:

Variable name	Dimension	Type	Units	Comments
antenna_gain_h	none	float	dB	Nominal antenna gain, H polarization
antenna_gain_v	none	float	dB	Nominal antenna gain, V polarization
beam_width_h	none	float	degrees	Antenna beam width H polarization
beam_width_v	none	float	degrees	Antenna beam width V polarization
receiver_bandwidth	none	float	s-1	Bandwidth of radar receiver

### 7.2 The *lidar\_parameters* sub-group

---

This group holds radar parameters specific to a lidar instrument:

Variable name	Dimension	Type	Units	Comments
beam_divergence	none	float	milliradians	Transmit side
field_of_view	none	float	milliradians	Receive side
aperture_diameter	none	float	cm	
aperture_efficiency	none	float	percent	
peak_power	none	float	watts	
pulse_energy	none	float	joules	

### 7.3 The *radar\_calibration* sub-group

---

For a radar, a different calibration is required for each pulse width. Therefore the calibration variables are arrays.

### 7.3.1 Dimensions

Dimension name	Description
calib	The number of radar calibrations available

### 7.3.2 Variables

The meaning of the designations used in the calibration variables are as follows for dual-polarization radars:

- 'h': horizontal channel
- 'v': vertical channel
- 'hc': horizontal co-polar (h transmit, h receive)
- 'hx' – horizontal cross-polar (v transmit, h receive)
- 'vc': vertical co-polar (v transmit, v receive)
- 'vx' – vertical cross-polar (h transmit, v receive)

For single polarization radars, the 'h' quantities should be used.

Variable name	Dimension	Type	Units	Comments
calib_index	(calib)	byte		Calibration index for each ray. Assumed 0 if missing.
time	(calib)	string	UTC	e.g. 2008-09-25 T23:00:00Z
pulse_width	(calib)	float	seconds	Pulse width for this calibration
antenna_gain_h	(calib)	float	dB	Derived antenna gain H channel
antenna_gain_v	(calib)	float	dB	Derived antenna gain V channel
xmit_power_h	(calib)	float	dBm	Transmit power H channel
xmit_power_v	(calib)	float	dBm	Transmit power V channel
two_way_waveguide_loss_h	(calib)	float	dB	2-way waveguide loss measurement plane to feed horn, H channel

Variable name	Dimension	Type	Units	Comments
two_way_waveguide_loss_v	(calib)	float	dB	2-way waveguide loss measurement plane to feed horn, V channel
two_way_radome_loss_h	(calib)	float	dB	2-way radome loss H channel
two_way_radome_loss_v	(calib)	float	dB	2-way radome loss V channel
receiver_mismatch_loss	(calib)	float	dB	Receiver filter bandwidth mismatch loss
receiver_mismatch_loss_h	(calib)	float	dB	Receiver filter bandwidth mismatch loss H channel
receiver_mismatch_loss_v	(calib)	float	dB	Receiver filter bandwidth mismatch loss V channel
radar_constant_h	(calib)	float	m/mW dB units	Radar constant H channel
radar_constant_v	(calib)	float	m/mW dB units	Radar constant V channel
probert_jones_correction	(calib)	float	dB	Probert Jones antenna correction factor.
dielectric_factor_used	(calib)	float		This is $ K^2 $ in the radar equation
noise_hc	(calib)	float	dBm	Measured noise level H co-pol channel
noise_vc	(calib)	float	dBm	Measured noise level V co-pol channel
noise_hx	(calib)	float	dBm	Measured noise level H cross-pol channel
noise_vx	(calib)	float	dBm	Measured noise level V cross-pol channel
receiver_gain_hc	(calib)	float	dB	Measured receiver gain H co-pol channel
receiver_gain_vc	(calib)	float	dB	Measured receiver gain V co-pol channel
receiver_gain_hx	(calib)	float	dB	Measured receiver gain H cross-pol channel
receiver_gain_vx	(calib)	float	dB	Measured receiver gain V cross-pol channel

Variable name	Dimension	Type	Units	Comments
base_1km_hc	(calib)	float	dBZ	reflectivity at 1km for SNR=0dB noise-corrected H co-pol channel
base_1km_vc	(calib)	float	dBZ	reflectivity at 1km for SNR=0dB noise-corrected V co-pol channel
base_1km_hx	(calib)	float	dBZ	reflectivity at 1km for SNR=0dB noise-corrected H cross-pol channel
base_1km_vx	(calib)	float	dBZ	reflectivity at 1km for SNR=0dB noise-corrected V cross-pol channel
sun_power_hc	(calib)	float	dBm	Calibrated sun power H co-pol channel
sun_power_vc	(calib)	float	dBm	Calibrated sun power V co-pol channel
sun_power_hx	(calib)	float	dBm	Calibrated sun power H cross-pol channel
sun_power_vx	(calib)	float	dBm	Calibrated sun power V cross-pol channel
noise_source_power_h	(calib)	float	dBm	Noise source power H channel
noise_source_power_v	(calib)	float	dBm	Noise source power V channel
power_measure_loss_h	(calib)	float	dB	Power measurement loss in coax and connectors H channel
power_measure_loss_v	(calib)	float	dB	Power measurement loss in coax and connectors V channel
coupler_forward_loss_h	(calib)	float	dB	Coupler loss into waveguide H channel
coupler_forward_loss_v	(calib)	float	dB	Coupler loss into waveguide V channel

Variable name	Dimension	Type	Units	Comments
zdr_correction	(calib)	float	dB	corrected = measured + correction
ldr_correction_h	(calib)	float	dB	corrected = measured + correction
ldr_correction_v	(calib)	float	dB	corrected = measured + correction
system_phidp	(calib)	float	degrees	System PhiDp, as seen in drizzle close to radar
test_power_h	(calib)	float	dBm	Calibration test power H channel
test_power_v	(calib)	float	dBm	Calibration test power V channel
receiver_slope_hc	(calib)	float		Computed receiver slope, ideally 1.0 H co-pol channel
receiver_slope_vc	(calib)	float		Computed receiver slope, ideally 1.0 V co-pol channel
receiver_slope_hx	(calib)	float		Computed receiver slope, ideally 1.0 H cross-pol channel
receiver_slope_vx	(calib)	float		Computed receiver slope, ideally 1.0 V cross-pol channel

#### 7.4 The *lidar\_calibration* sub-group

---

At the time of writing, the contents of this group have not been defined.

#### 7.5 The *georeference\_correction* sub-group

---

The following additional variables are used to quantify errors in the georeference data for moving platforms (see 5.4). These are constant for a volume.

If any item is omitted, the value is assumed to be 0.

Variable name	Dimension	Type	Units	Comments
azimuth_correction	none	float	degrees	Correction to azimuth values

Variable name	Dimension	Type	Units	Comments
elevation_correction	none	float	degrees	Correction to elevation values
range_correction	none	float	meters	Correction to range values
longitude_correction	none	float	degrees	Correction to longitude values
latitude_correction	none	float	degrees	Correction to latitude values
pressure_altitude_correction	none	float	meters	Correction to pressure altitude values
radar_altitude_correction	none	float	meters	Correction to radar altitude values
eastward_ground_speed_correction	none	float	m/s	Correction to EW ground speed values
northward_ground_speed_correction	none	float	m/s	Correction to NS ground speed values
vertical_velocity_correction	none	float	m/s	Correction to vertical velocity values
heading_correction	none	float	degrees	Correction to heading values
roll_correction	none	float	degrees	Correction to roll values
pitch_correction	none	float	degrees	Correction to pitch values
drift_correction	none	float	degrees	Correction to drift values
rotation_correction	none	float	degrees	Correction to rotation values
tilt_correction	none	float	degrees	Correction to tilt values

## 8 Standard names

---

To the extent possible, CfRadial uses standard names already defined by CF.

The `standard_name` entries not already accepted by CF have been requested for inclusion.

The relevant standard names for radar and lidar data will be specified in a separate document.



## 9 Computing the data location from geo-reference variables

Weather radars and lidars rotate primarily about a *principal axis* (e.g., “zenith” for plan-position-indicator mode in ground-based radar), slew about a secondary axis, orthogonal to the primary axis (e.g., range-height-indicator in ground-based radar), or slew on a plane by changing both primary and secondary axis (e.g., COPLANE in ground-based radar).

In the ground-based radar convention, a point in space relative to a radar is represented in a local spherical coordinate systems  $\mathbf{X}_i$  by three parameters, range ( $r$ ), azimuth ( $\lambda$ ), and elevation ( $\phi$ ). A ground-based radar is assumed “leveled” with positive (negative) elevation,  $\phi$ , above (below) a *reference plane* (a leveled plane orthogonal to the principal axis and containing the radar). The azimuth angle,  $\lambda$ , is the angle on the reference plane increases clockwise from the True North (TN) following the Meteorological coordinate convention (e.g., TN is  $0^\circ$  and East is  $90^\circ$ ).

Processing and manipulating radar data (e.g., interpolation, synthesis, etc.) typically are performed in a right-handed 3-D XYZ Cartesian geo-referenced coordinate system  $\mathbf{X}$  (see Fig. 7.1) where Y is TN and X is East. Hence, a coordinate transformation between  $\mathbf{X}_i$  (radar sampling space) and  $\mathbf{X}$  (geo-reference space) is required. Based on the principal axes, most remote sensors can be classified into three right-hand types, X, Y, or Z type.

The purpose of this chapter is two-fold: (1) to define a consistent terminology for the CfRadial format, and (2) to derive coordinate transformation matrices for each type of remote sensor. Many sensors (e.g. fixed ground radars) are of the Z-type, have a fixed location, are leveled and are aligned relative to True North (TN). Dealing with such sensors is much simpler than for those on moving platforms. Therefore, they will be dealt with first, and the more complicated treatment of all three types of remote sensor mounted on moving platforms will be covered in the later sections.

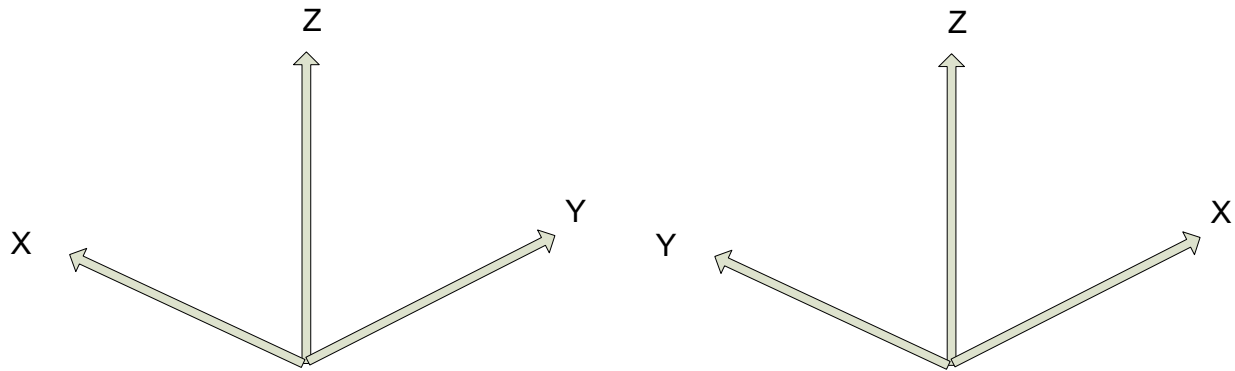


Figure 9.1: Left-handed XYZ coordinate system vs. Right-handed XYZ coordinate system.

In addition to the standard X, Y and Z right-hand types, specialized types such as the ELDORA and NOAA aircraft tail radars will be handled separately. The tail radars will be referred to as type Y-prime.

## 9.1 Special case – ground-based, stationary and leveled sensors

---

Ground-based sensors (radars and lidars) rotate primarily about the vertical (Z) axis (Z-Type), and the reference plane is a horizontal XY plane passing through the sensor. The Y-axis is aligned with TN, and the X-axis points East.

Azimuth angles ( $\lambda$ ) are positive clockwise looking from above (+Z), with 0 being TN.

Elevation angles ( $\phi$ ) are measured relative to the horizontal reference plane, positive above the plane and negative below it.

A ground-based, leveled vertical pointing sensor can be classified as a Z-Type with  $\phi=90^\circ$ .

### 9.1.1 LIDARs

---

For LIDARs, the assumption is generally made that propagation of the beam is along a straight line, emanating at the sensor. The coordinate transformation between  $\mathbf{X}_i(r, \lambda, \phi)$  and  $\mathbf{X}(x, y, z)$  is as follows:

$$x = x_0 + r \cos \phi \sin \lambda$$

$$y = y_0 + r \cos \phi \cos \lambda$$

$$z = z_0 + r \sin \phi$$

where

$x$  is positive east

$y$  is positive north

$(x_0, y_0, z_0)$  are the coordinates of the sensor relative to the Cartesian grid origin and the azimuth angle ( $\lambda$ ) is the angle clockwise from TN.

The sensor location is specified in longitude, latitude and altitude in the CfRadial format. Locations in the earth's geo-reference coordinate system are computed using the sensor location and the  $(x, y, z)$  from above, using normal spherical geometry.

### 9.1.2 RADARs

---

The propagation of radar microwave energy in a beam through the lower atmosphere is affected by the change of refractive index of the atmosphere with height. Under average conditions this causes the beam to be deflected downwards, in what is termed 'Standard Refraction'. For most purposes this is adequately modeled by assuming that the beam is in fact straight, relative to an earth which has a radius of 4/3 times the actual earth radius. (Rinehart 2004.)

For a stationary and leveled, ground-based radar, the equations are similar to those for the LIDAR case, except that we have one extra term, the height correction, which reflects the beam curvature relative to the earth.

The height  $h$  above the earth's surface for a given range is:

$$h = \sqrt{r^2 + R'^2 + 2rR' \sin(\phi)} - R' + h_0$$

where  $R' = \left(\frac{4}{3}\right) \cdot 6374 \text{ km}$  is the pseudo radius of earth. See Rinehart 2004, Chapter 3, for more details.

The  $(x,y)$  location for a given range is:

$$x = x_0 + r \cos \phi \sin \lambda$$

$$y = y_0 + r \cos \phi \cos \lambda$$

where  $x$  is positive east,  $y$  is positive north, and remembering that azimuth is the angle clockwise from true north.

## 9.2 Moving platforms

---

For moving platforms, the metadata for each beam will include:

- longitude of instrument
- latitude of instrument
- altitude of instrument
- rotation and tilt of the beam (see above)
- roll, pitch and heading of the platform
- platform motion ( $U_G$ ,  $V_G$ ,  $W_G$ )
- air motion ( $U_{\text{air}}$ ,  $V_{\text{air}}$ ,  $W_{\text{air}}$ )

For ground-based moving platforms (e.g., Doppler on Wheels), the earth-relative location of the observed point is:

$$x = x_0 + r \cos \phi \sin \lambda$$

$$y = y_0 + r \cos \phi \cos \lambda$$

$$h = \sqrt{r^2 + R'^2 + 2rR' \sin \phi} - R' + h_0$$

Note that for airborne radar platforms, correcting for refractive index does not apply. Therefore, for airborne radars, use the straight line equations for LIDARs.

Refer to the sections below for the computation of elevation ( $\phi$ ) and azimuth ( $\lambda$ ) relative to earth coordinates.

Then apply the following equations, as before, to compute the location of the observed point.

$$x = x_0 + r \cos \phi \sin \lambda$$

$$y = y_0 + r \cos \phi \cos \lambda$$

$$z = z_0 + r \sin \phi$$

### 9.3 Coordinate transformations for the general case

---

This section details the processing for the general case.

Sensors which do not fall under section 7.1 above must be handled as a general case.

#### 9.3.1 Coordinate systems

---

In addition to the previously-defined  $\mathbf{X}_i$  and  $\mathbf{X}$  coordinate systems, the following intermediate right-handed coordinate systems need to be defined to account for a moving, non-leveled platform:

- $\mathbf{X}_a$ : platform-relative coordinates, +Y points to heading, +X points to the right side (90° clockwise from +Y on the reference plane XY), +Z is orthogonal to the reference plane.
- $\mathbf{X}_h$ : leveled, platform heading-relative coordinates, +Y points heading, +X points 90° clockwise from heading, and Z points up (local zenith).

The goal here is to derive transformations from  $\mathbf{X}_i$  to  $\mathbf{X}$  via  $\mathbf{X}_a$  and  $\mathbf{X}_h$ .

#### 9.3.2 The earth-relative coordinate system

---

The earth-relative coordinate system,  $\mathbf{X}$ , is defined as follows, X is East, Y is North, and Z is zenith. Azimuth angle,  $\lambda$ , is defined as positive *clockwise* from TN (i.e., meteorological angle) while elevation angle,  $\phi$ , is defined positive/negative above/below the horizontal plane at the altitude ( $h_0$ ) of the remote sensor.

#### 9.3.3 The platform-relative coordinate system

---

The general form of the mathematic representation describes a remote sensing device mounted on a moving platform (e.g., an aircraft, see Figure 7.2). This figure depicts the theoretical reference frame for a moving platform. (We use the aircraft analogy here, but the discussion also applies to water-borne platforms and land-based moving platforms.)

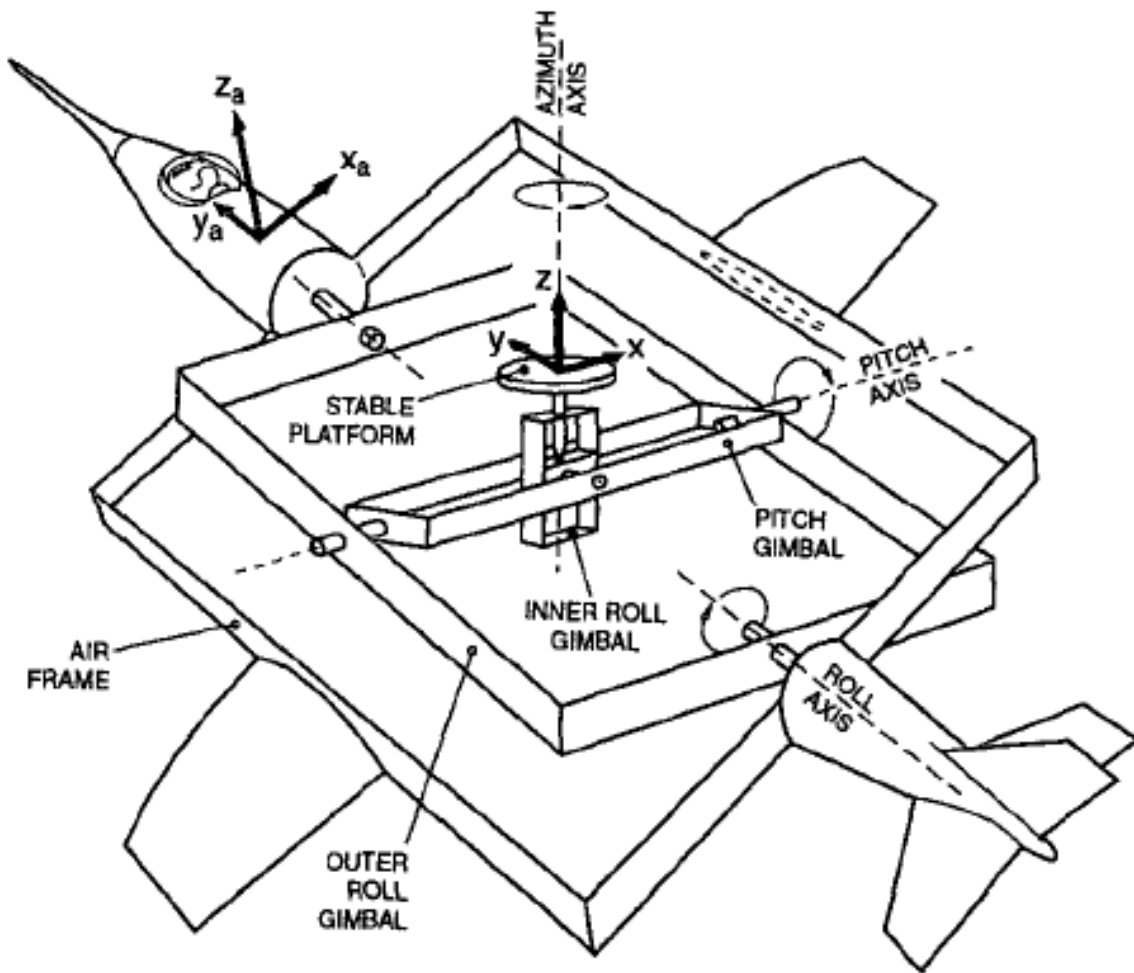
The platform-relative coordinate system of the platform,  $\mathbf{X}_a$ , is defined by the right side, ( $X_a$ ), the heading, ( $Y_a$ ), and the zenith, ( $Z_a$ ).

The origin of  $\mathbf{X}_a$  is defined as the location of the INS on a moving platform.

The platform-relative coordinate system is defined by 3 rotations in the following order: heading ( $H$ ), pitch ( $P$ ) and roll ( $R$ ) angles from  $\mathbf{X}$ . These angles are generally measured by an inertial navigation system (INS).

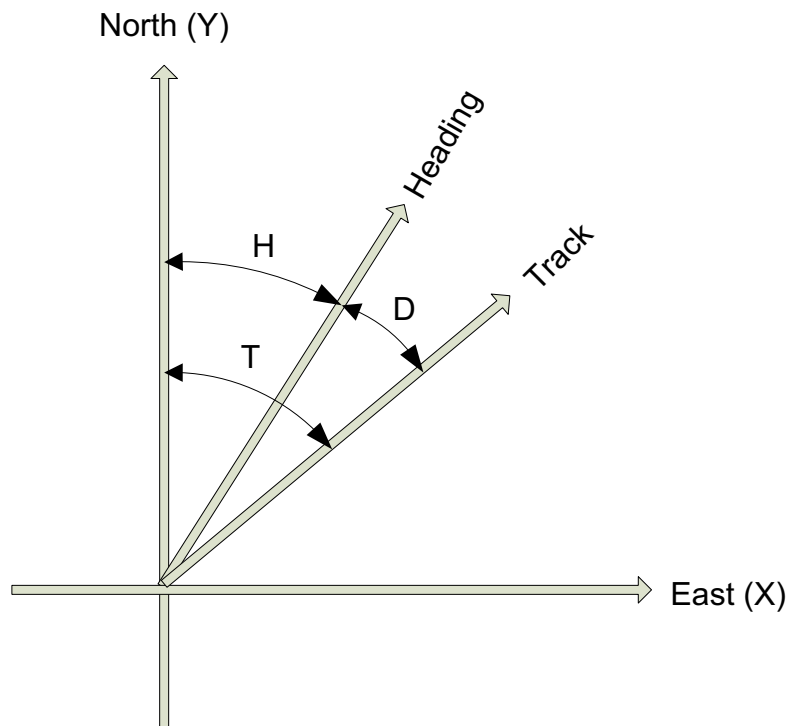
The platform moves relative to  $\mathbf{X}$ , based on its heading  $H$ , and the drift  $D$ , caused by wind or current. ( $D$  is 0 for land-based platforms). The track  $T$  is the line of the platform movement over the earth surface.

NOTE: -see Lee et al. (1994) for further background on this topic, and on the corrections to Doppler velocity for moving platforms. Usually, the platform INS and the sensor may not be collocated. The Doppler velocity needs to be compensated by the relative motion between these two.

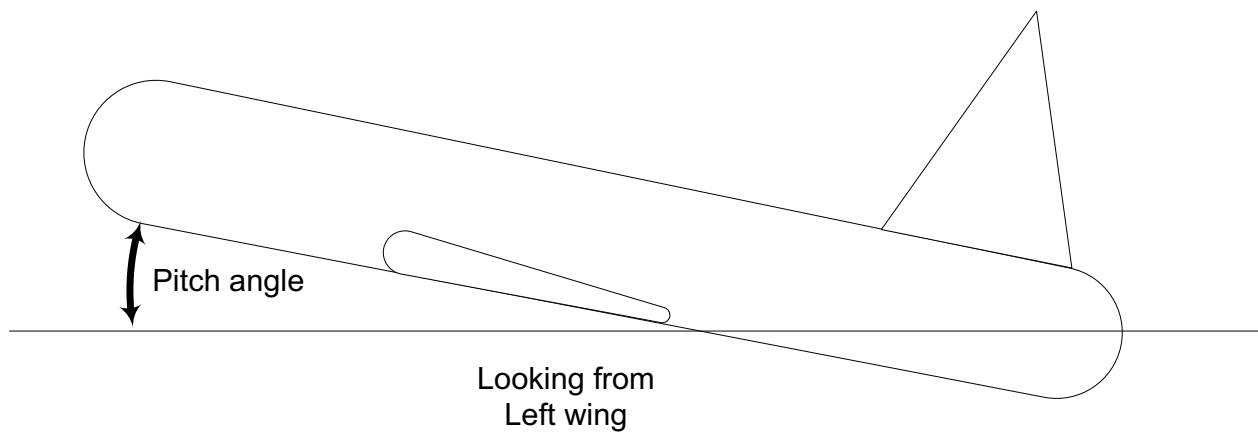


*Figure 9.2 Moving platform axis definitions and reference frame (reproduced from Lee et al., 1994, originally from Axford, 1968) ©American Meteorological Society. Reprinted with permission.*

Figures 7.3 a through c show the definitions of heading, drift, track, pitch and roll.



**Figure 9.3(a): Definition of heading, drift and track.**



**Figure 9.3(b): Definition of pitch**

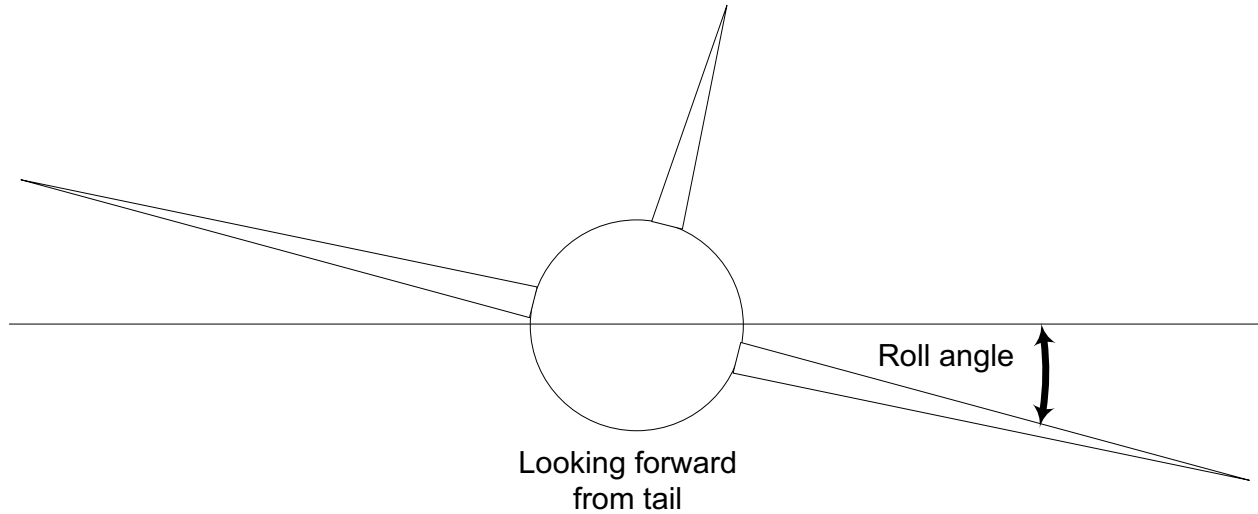


Figure 9.3(c): Definition of roll

### 9.3.4 The sensor coordinate system

In the sensor coordinate system,  $\mathbf{X}_i$ , each data location is characterized by a range,  $r$ , a rotation angle,  $\theta$ , and a tilt angle,  $\tau$ . Following the ground-based radar convention, the rotation angle,  $\theta$ , is the angle projected on the reference plane, positive *clockwise* from the third axis (counting from the principal axis in  $\mathbf{X}_a$ ) looking *towards the sensor* from the positive principal axis. The tilt angle,  $\tau$ , is the angle of the beam relative to the reference plane. A beam has a positive/negative  $\tau$  depending on whether it is on the positive/negative side of the reference plane, using the principal axis to determine the sign. Each gate location  $(r, \theta, \tau)$  in  $\mathbf{X}_i$  can be represented in  $(r, \lambda, \phi)$  in  $\mathbf{X}$ .

Table 9.1: Characteristics of 4 types of sensors.

Sensor Type	Type X	Type Y	Type Y-prime	Type Z
Principal Axis	$X_a$	$Y_a$	$Y_a$	$Z_a$
Reference Plane	$Y_a Z_a$	$Z_a X_a$	$Z_a X_a$	$X_a Y_a$
0° Rotation Angle	$+Z_a$	$+X_a$	$+Z_a$	$+Y_a$
90° Rotation Angle	$+Y_a$	$+Z_a$	$+X_a$	$+X_a$
Examples	EDOP, Wyoming Cloud Radar, Wind Profiler, downward scanning radar on Global Hawk		Tail Doppler radars on NOAA P3 and NSF/NCAR ELDORA	Ground-based radar/lidar, aircraft nose radar, NOAA P3 lower-fuselage radar, C-band scatterometer

## 9.4 Coordinate transformation sequence

---

The following transformations are carried out to transform the geometry from the instrument-based ( $\mathbf{X}_i$ ) to the earth-based coordinate system ( $\mathbf{X}$ ):

- translate from  $\mathbf{X}_i$  to  $\mathbf{X}_a$
- rotate from  $\mathbf{X}_a$  to  $\mathbf{X}$

### 9.4.1 Transformation from $\mathbf{X}_i$ to $\mathbf{X}_a$

---

The details of this step depend on the sensor type: Z, Y or X (Table 7.1)

#### 9.4.1.1 Type Z sensors

The characteristics are:

- the primary axis is  $Z_a$
- the reference plane is  $(X_a, Y_a)$
- the rotation angle  $\theta$  is 0 in the  $(Y_a, Z_a)$  plane, i.e. along the  $+Y$  axis. Rotation increases clockwise from  $+Y$ , when looking from above (i.e. from  $+Z$ )
- the tilt angle  $\tau$  is 0 in the  $(X_a, Y_a)$  plane, positive above it (for  $+Z_a$ ) and negative below it.

The transformation to  $\mathbf{X}_a$  coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin \theta \cos \tau \\ \cos \theta \cos \tau \\ \sin \tau \end{pmatrix}$$

#### 9.4.1.2 Type Y sensors

The characteristics are:

- the primary axis is  $Y_a$
- the reference plane is  $(Z_a, X_a)$
- the rotation angle  $\theta$  is 0 in the  $(Z_a, X_a)$  plane, i.e. along the  $+X_a$  axis. Rotation increases clockwise from  $+X$ , when looking from  $+Y$ .
- the tilt angle  $\tau$  is 0 in the  $(Z_a, X_a)$  plane, positive for  $+Y_a$ .

***Note that the definition of  $\theta$  is different from the convention defined in Lee et al. (1994)<sup>1</sup>. Let  $\theta'$  be the rotation angle defined in Lee et al. (1994),  $\theta = \text{mod}(450^\circ - \theta')$ .***

---

<sup>1</sup> The rotation angle,  $\theta'$ , defined in previous airborne tail Doppler radar convention (Lee et al. 1994) was positive clockwise looking from the tail toward the nose of an aircraft (i.e., looking from the  $-Y_a$ -axis) that has been the convention for airborne tail Doppler radars.  $\theta' = 0^\circ$  points to



The transformation to  $\mathbf{X}_a$  coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \cos \theta \cos \tau \\ \sin \tau \\ \sin \theta \cos \tau \end{pmatrix}$$

#### 9.4.1.3 Type Y-prime sensors

The characteristics are:

the primary axis is  $Y_a$

the reference plane is  $(Z_a, X_a)$

the rotation angle  $\theta$  is 0 in the  $(Y_a, Z_a)$  plane, i.e. along the  $+Z_a$  axis. Rotation increases clockwise from  $+Z$ , when looking from  $-Y$ .

the tilt angle  $\tau$  is 0 in the  $(Z_a, X_a)$  plane, positive for  $+Y_a$ .

*Note that the definition of  $\theta$  is the convention defined in Lee et al. (1994)*

The transformation to  $\mathbf{X}_a$  coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin \theta \cos \tau \\ \sin \tau \\ \cos \theta \cos \tau \end{pmatrix}$$

#### 9.4.1.4 Type X sensors

The characteristics are:

- the primary axis is  $X_a$
- the reference plane is  $(Y_a, Z_a)$
- the rotation angle  $\theta$  is 0 in the  $(Y_a, Z_a)$  plane, i.e. along the  $+Z_a$  axis. Rotation increases clockwise from  $+Z_a$ , when looking from  $+X_a$ .
- the tilt angle  $\tau$  is 0 in the  $(Y_a, Z_a)$  plane, positive for  $+X_a$ .

The transformation to  $\mathbf{X}_a$  coordinates is:

$$\begin{pmatrix} x_a \\ y_a \\ z_a \end{pmatrix} = r \begin{pmatrix} \sin \tau \\ \sin \theta \cos \tau \\ \cos \theta \cos \tau \end{pmatrix}$$

### 9.4.2 Rotating from $\mathbf{X}_a$ to $\mathbf{X}$

---

Rotating  $\mathbf{X}_a$  to  $\mathbf{X}$  requires the following 3 steps (in the reverse order of the rotation):

- remove the roll  $R$ , by rotating the x axis around the y axis by  $-R$ .
- 

$+Z$ . However, this convention is different from that used in the ground-based radars. The  $r$  and  $\tau$  were defined the same way in the current convention.

- remove the pitch  $P$ , by rotating the y axis around the x axis by  $-P$ .
- remove the heading  $H$ , by rotating the y axis around the z axis by  $+H$

The transformation matrix for removing the roll component is:

$$M_R = \begin{pmatrix} \cos R & 0 & \sin R \\ 0 & 1 & 0 \\ -\sin R & 0 & \cos R \end{pmatrix}$$

The transformation matrix for removing the pitch component is:

$$M_P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos P & -\sin P \\ 0 & \sin P & \cos P \end{pmatrix}$$

The transformation matrix for removing the heading component is:

$$M_H = \begin{pmatrix} \cos H & \sin H & 0 \\ -\sin H & \cos H & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We apply these transformations consecutively:

$$X = M_H M_P M_R X_a$$

$$\begin{aligned} M_H M_P M_R &= \begin{pmatrix} \cos H & \sin H & 0 \\ -\sin H & \cos H & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos P & -\sin P \\ 0 & \sin P & \cos P \end{pmatrix} \begin{pmatrix} \cos R & 0 & \sin R \\ 0 & 1 & 0 \\ -\sin R & 0 & \cos R \end{pmatrix} \\ &= \begin{pmatrix} \cos H \cos R + \sin H \sin P \sin R & \sin H \cos P & \cos H \sin R - \sin H \sin P \cos R \\ -\sin H \cos R + \cos H \sin P \sin R & \cos H \cos P & -\sin H \sin R - \cos H \sin P \cos R \\ -\cos P \sin R & \sin P & \cos P \cos R \end{pmatrix} \\ &= \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \end{aligned}$$

## 9.5 Summary of transforming from $X_i$ to $X$

---

We combine the above 2 main steps for transform all the way from the instrument coordinates to earth coordinates:

### 9.5.1 For type Z radars:

---

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin \theta \cos \tau \\ \cos \theta \cos \tau \\ \sin \tau \end{pmatrix}$$

$$= r \begin{pmatrix} m_{11} \sin \theta \cos \tau + m_{12} \cos \theta \cos \tau + m_{13} \sin \tau \\ m_{21} \sin \theta \cos \tau + m_{22} \cos \theta \cos \tau + m_{23} \sin \tau \\ m_{31} \sin \theta \cos \tau + m_{32} \cos \theta \cos \tau + m_{33} \sin \tau \end{pmatrix}$$

### 9.5.2 For type Y radars:

---

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \cos \theta \cos \tau \\ \sin \tau \\ \sin \theta \cos \tau \end{pmatrix}$$

$$= r \begin{pmatrix} m_{11} \cos \theta \cos \tau + m_{12} \sin \tau + m_{13} \sin \theta \cos \tau \\ m_{21} \cos \theta \cos \tau + m_{22} \sin \tau + m_{23} \sin \theta \cos \tau \\ m_{31} \cos \theta \cos \tau + m_{32} \sin \tau + m_{33} \sin \theta \cos \tau \end{pmatrix}$$

### 9.5.3 For type Y-prime radars:

---

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin \theta \cos \tau \\ \sin \tau \\ \cos \theta \cos \tau \end{pmatrix}$$

$$= r \begin{pmatrix} m_{11} \sin \theta \cos \tau + m_{12} \sin \tau + m_{13} \cos \theta \cos \tau \\ m_{21} \sin \theta \cos \tau + m_{22} \sin \tau + m_{23} \cos \theta \cos \tau \\ m_{31} \sin \theta \cos \tau + m_{32} \sin \tau + m_{33} \cos \theta \cos \tau \end{pmatrix}$$

#### 9.5.4 For type X radars:

---

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} r \begin{pmatrix} \sin \tau \\ \sin \theta \cos \tau \\ \cos \theta \cos \tau \end{pmatrix}$$

$$= r \begin{pmatrix} m_{11} \sin \tau + m_{12} \sin \theta \cos \tau + m_{13} \cos \theta \cos \tau \\ m_{21} \sin \tau + m_{22} \sin \theta \cos \tau + m_{23} \cos \theta \cos \tau \\ m_{31} \sin \tau + m_{32} \sin \theta \cos \tau + m_{33} \cos \theta \cos \tau \end{pmatrix}$$

#### 9.5.5 Computing earth-relative azimuth and elevation

---

We can then compute the earth-relative azimuth and elevation as follows:

$$\lambda = \tan^{-1}(x/y)$$

$$\phi = \sin^{-1}(z/r)$$

### 9.6 Summary of symbol definitions

---

$\mathbf{X}_i$ : instrument-relative coordinate system,  $(r, \theta, \tau)$  or  $(r, \lambda, \phi)$

$\mathbf{X}_a$ : platform-relative coordinate system  $(x_a, y_a, z_a)$  – see figure 7.2

$\mathbf{X}_h$ : coordinate system relative to level platform (no roll or pitch) with heading  $H$ .

$\mathbf{X}$ : earth-relative coordinate system  $(x, y, z)$ ,  $x$  is positive east,  $y$  is positive north,  $z$  is positive up.

$H$ : heading of platform (see figure 7.3)

$T$ : track of platform (see figure 7.3)

$D$ : drift angle (see figure 7.3)

$P$ : pitch angle (see figure 7.3)

$R$ : roll angle (see figure 7.3)

$\lambda$ : azimuth angle

$\phi$ : elevation angle

$\theta$ : rotation angle

$\tau$ : tilt angle

$r$ : range

$h$ : height

$h_0$ : height of the instrument

$R'$ : pseudo radius of earth =  $(4/3)6374km$

## 10 References

---

Axford, D. N., 1968: On the accuracy of wind measurements using an inertial platform in an aircraft, and an example of a measurement of the vertical structure of the atmosphere. *J. Appl. Meteor.*, 7, 645-666.

Doviak, R.J., and D.S. Zrnic, Doppler Radar and Weather Observations, 1984, *Academic Press*, 458pp.

Lee, W., P. Dodge, F. D. Marks Jr. and P. Hildebrand, 1994: Mapping of Airborne Doppler Radar Data. *Journal of Oceanic and Atmospheric Technology*, 11, 572 – 578.

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.

Rinehart, R. E., 2004: Radar for Meteorologists, Fourth Edition. *Rinehart Publications*. ISBN 0-9658002-1-0