NCAR/UNIDATA

CfRadial2 Data File Format

CF2-compliant NetCDF Format for Moments Data for RADAR and LIDAR in Radial Coordinates

Version 2.0 **DRAFT**

Mike Dixon

Wen-Chau Lee

EOL, NCAR*

2016-08-01

* NCAR is sponsored by the US National Science Foundation.

1 Introduction

1.1 Purpose

The purpose of this document is to specify CfRadial2, a CF2-compliant NetCDF format for radar and lidar moments data in radial (i.e. polar) coordinates.

1.2 Terminology – CfRadial1 and CfRadial2

We will refer to CfRadial 2.0, and future 2.x versions, collectively as CfRadial2.

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

1.3 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. 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).

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.
2016/08/01	2.0	Major revision – not backward compatible with CfRadial1 Uses NetCDF 4 and groups. Combines CfRadial 1.4 and ODIM-H5 version 2.2.

Table 1.1: History of CfRadial versions

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.

A WMO-sponsored meeting was held at NCAR in Boulder during July 2016, at 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 radar community ODIM-H5 (HDF5) format, 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 TT-WRDE.

1.4 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/cf-conventions/v1.6.0/cf-conventions.html

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.

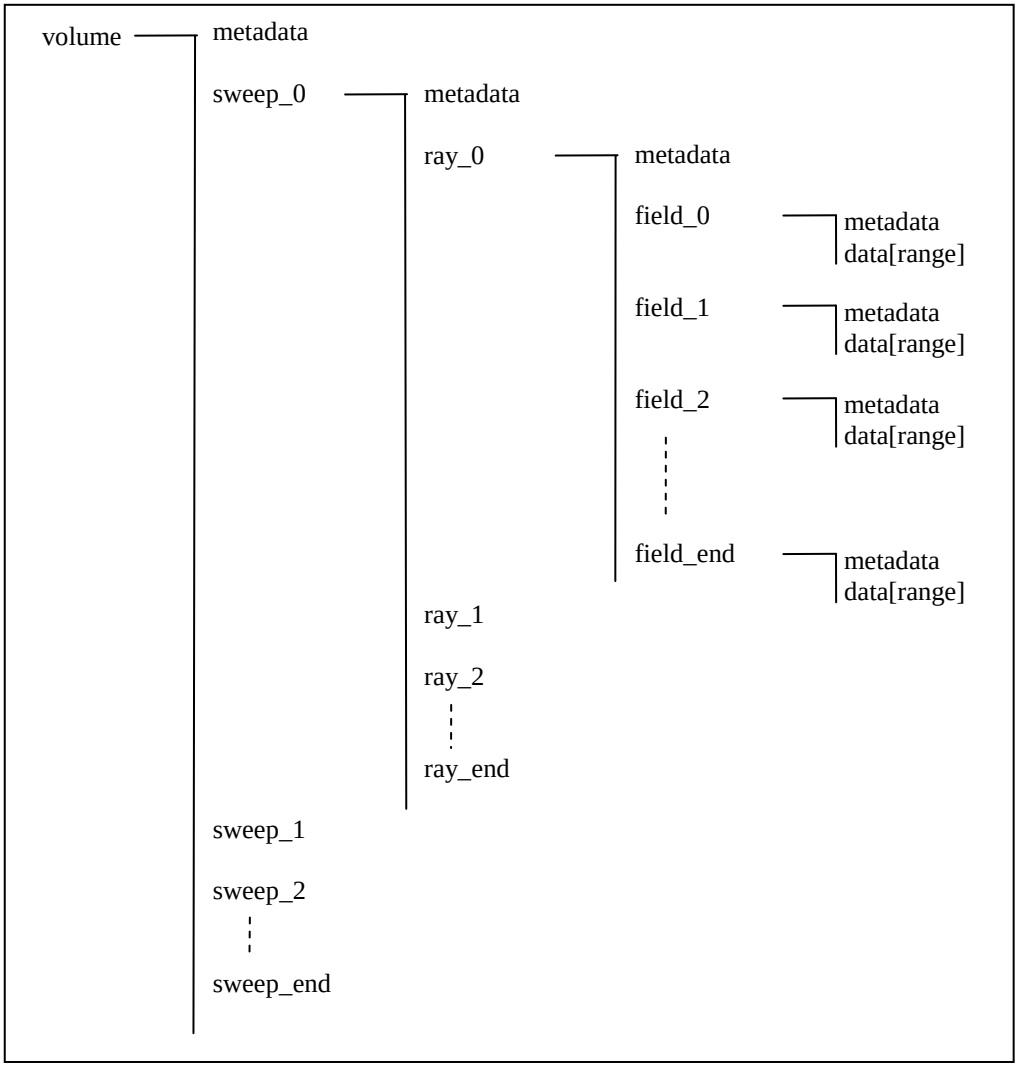

Figure 2.1: logical data structure for a volume scan, using sweep and ray abstractions and 1-D data field arrays

Figure 2.1 shows an idealized data model for a radar or lidar **volume**, expressing 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**:

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

For a given **ray**, the **field** 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

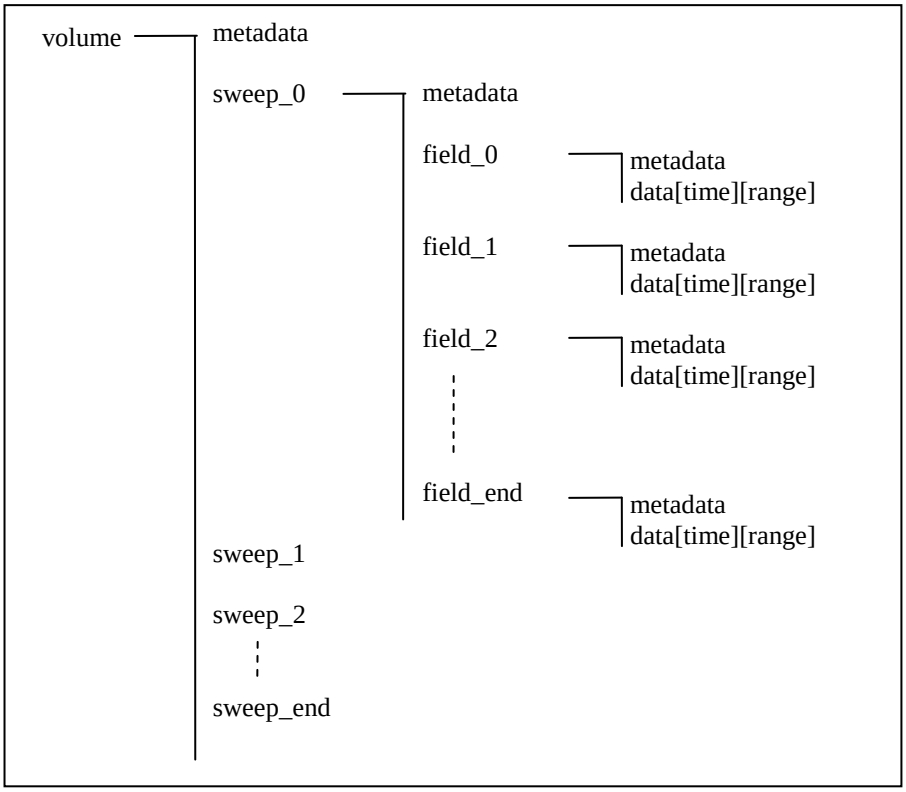

Figure 2.2: logical data structure for a volume scan, using the sweep abstraction with 2-D data fields

Figure 2.2 shows a modified data model, in which the **sweeps** contain the **field** data directly, stored as 2-D arrays of [time][range]. This requires that the number of gates in a sweep does not vary from ray to ray.

This version of the data model is used for CfRadial2 – see Figure 2.3 below.

This abstraction has the advantage that the 2-D arrays for sweeps simplify the data storage mechanism, and allow for more efficient data compression than 1-D arrays for rays.

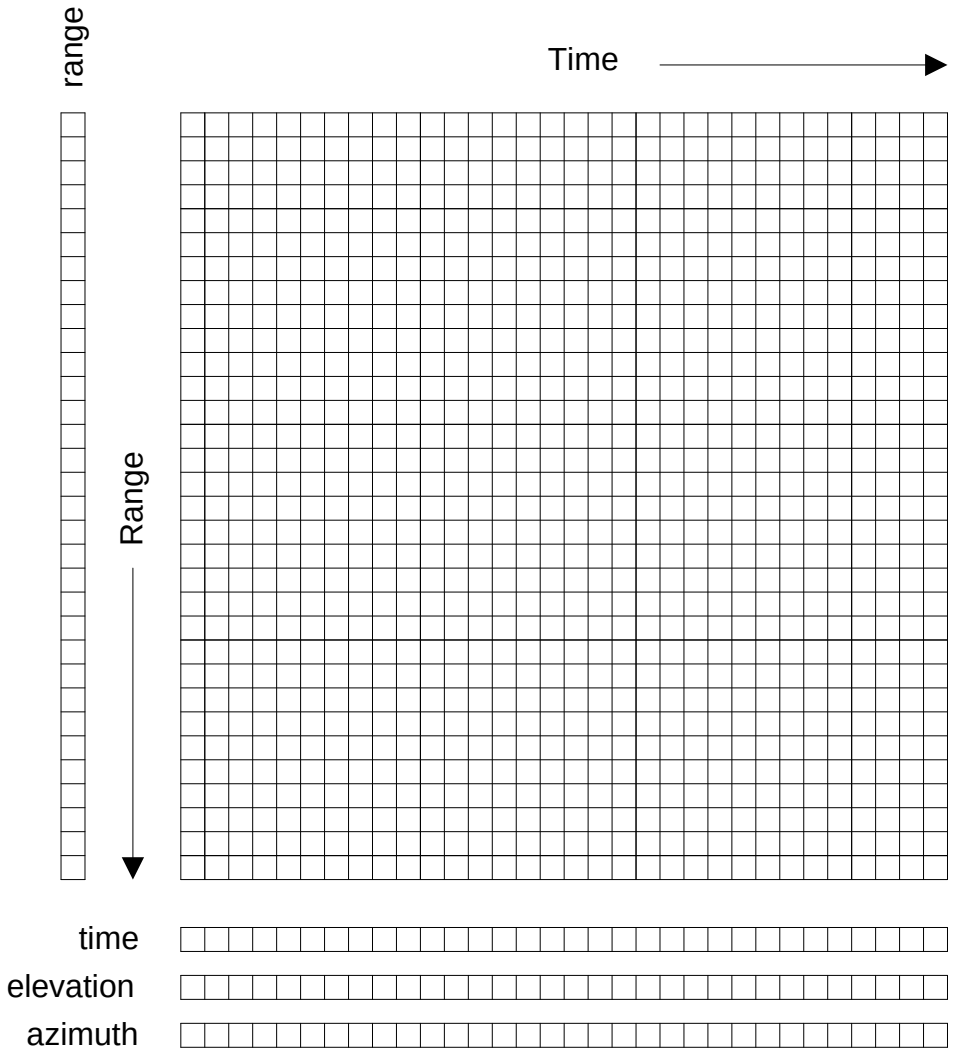

Figure 2.3 Data field represented in time and range, with a constant number of gates

2.3 Field data byte representation

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

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

Table 2.1: field data representation

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

data_value = (integer_value * scale_factor) + 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 constant azimuth (RHI mode)
- antenna not moving, i.e. constant elevation and azimuth (staring or pointing)
- aircraft radars which rotate around the longitudinal axis of the aircraft (e.g. ELDORA)
- sun scanning in either PPI or RHI mode
- vehicle following

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

- PPI mode: a sequence of rays at fixed elevation angle
- RHI mode: a sequence of rays at fixed azimuth angle
- pointing mode: a sequence of rays over some time period, at fixed azimuth and elevation
- ELDORA-type aircraft 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. Sometimes an **antenna_transition flag** is set for these transition rays to allow them to be optionally filtered out at a later 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, CfRadial1 uses the flat classic model of NetCDF3. A drawback of this approach is that the implementation can become complicated, since much of the metadata is placed at the top level in the data structure.

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 groups available in NetCDF4.

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

In CfRadial2 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_names', also at the top level. This allows the user to find 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 the 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 below.

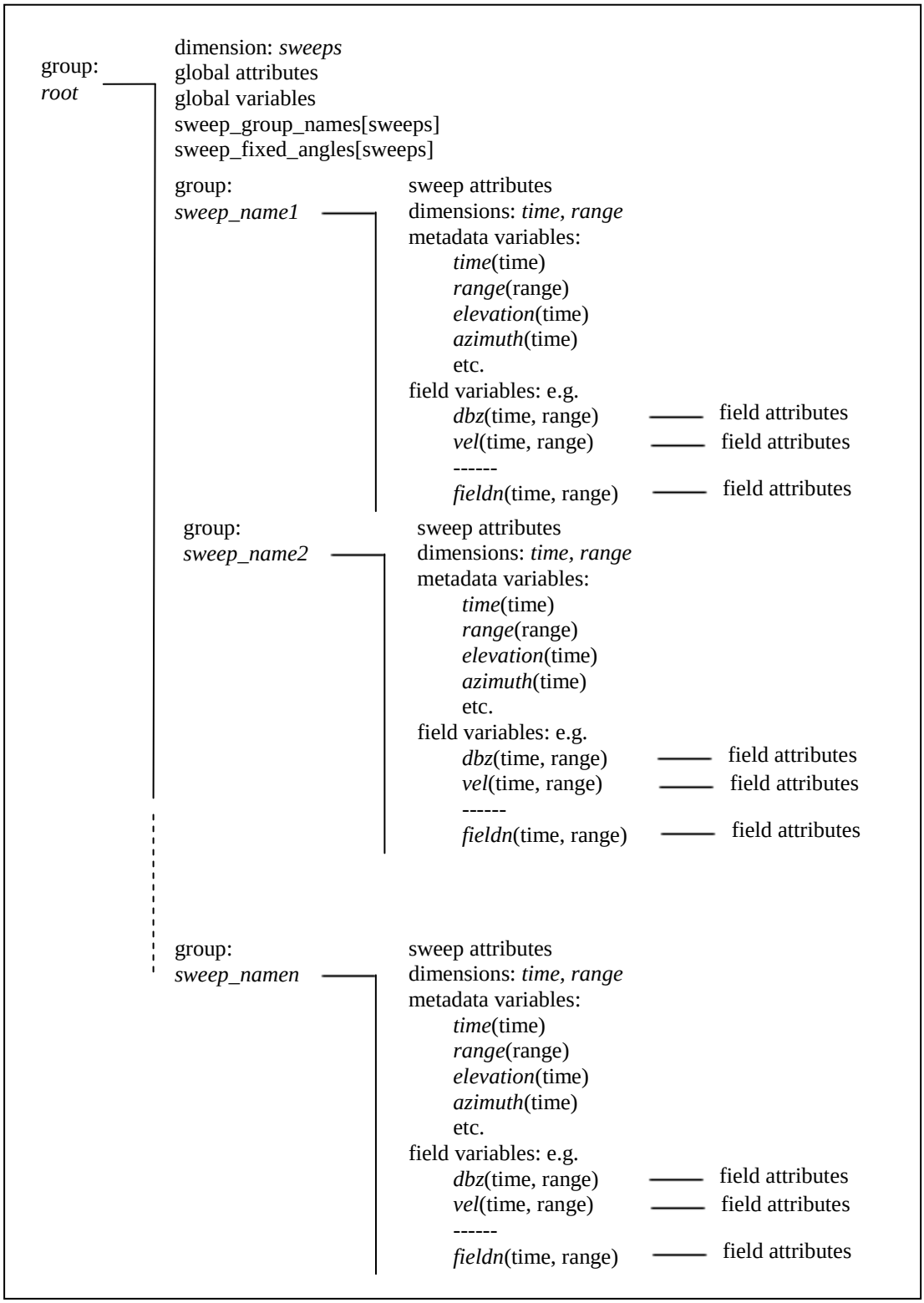

Figure 3.1: Group structure showing top-level dimensions, attributes, variables and sweep groups

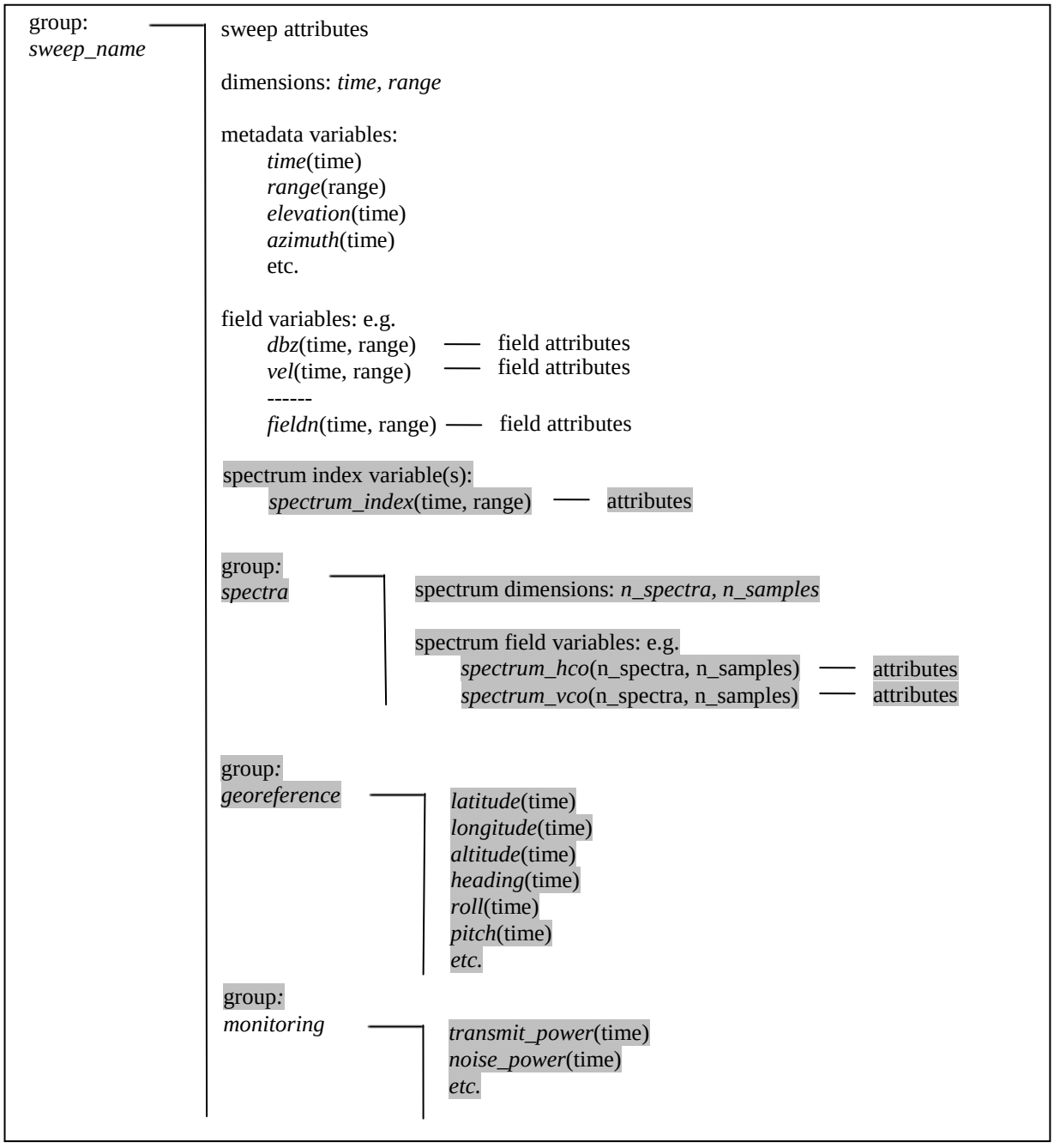

Figure 3.2: Sweep group structure in more detail, showing support for georeference metadata for moving platforms, spectra, and monitoring data.

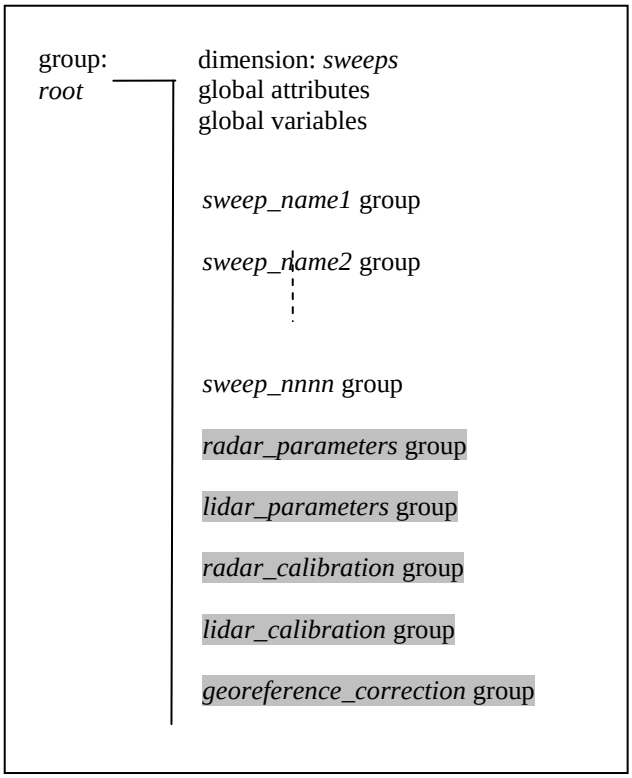

Figure 3.3: Optional metadata groups (highlighted in gray) in the root group

3.2 Principal dimensions and 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)** but may be a specified reference time variable **(time_reference**) if it exists.

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 Extensions to the CF convention

This convention requires the following extensions to CF:

1. The following axis attribute types:
   - axis = "radial_azimuth_coordinate";
   - axis = "radial_elevation_coordinate";
   - axis = "radial_range_coordinate";
2. For CfRadial to follow CF properly, support for the following must be added to the udunits package:
   - dB (ratio of two values, in log units. For example, ZDR).
   - dBm (power in milliwatts, in log units)
   - dBZ (radar reflectivity in log units)
3. 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

3.4 Strict use of variable and attribute names for non-field variables

In CfRadial a **field** variable stores such quantities as radar moments, derived quantities, data quality indicators etc. These are either measured by the radar, or derived from a field that is measured. These **field** variables require a **standard_name** attribute as specified in the CF conventions. For example the field variable 'radar reflectivity' would have the standard name '*equivalent_reflectivity_factor'*. These variables store the fundamental scientific data associated with the instrument.

By contrast, metadata variables store the dimensional information such as *time, range, azimuth* and *elevation,* and other metadata such as calibration and radar characteristics.

Because of the inherent complexity of radial radar and lidar data, the CfRadial format requires extra strictness, as compared to CF in general, in order to keep it manageable. There are so many metadata variables in CfRadial that it is essential to require **strict adherence** to the **dimension names** and **variable names** for the metadata variables, exactly as specified in this document. It is not practical to require a software application to search for standard names for metadata variables, since this makes the code unnecessarily complex and difficult to maintain.

To summarize, this strictness requirement only applies to **non-field metadata** variables. The **moments data** fields (i.e. the **field** variables) will be handled as usual in CF, where the **standard name** is the **definitive guide** to the contents of the field. The standard names for radar variables are listed in section 6.

3.5 _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** code. This procedure provides efficient compression.

3.6 Required vs. optional variables

Required variables are shown shaded in this document.

All other variables are optional.

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

3.7 No grid mapping variable

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.

A grid mapping type is not required, because the geo-reference variables contain all of the information required to locate the data in space.

For a *stationary* instrument, the following are stored as **scalar variables** (see section 4.6):

- latitude
- longitude
- altitude

Position and pointing references for *moving* platforms must take the following motions into account (see section 4.9):

- platform translation
- platform rotation

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, plus selected sub-conventions as applicable
version	string	CF/Radial	CF/Radial version number
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 the sweeps in the volume.

Attribute name	Type	Convention	Description
field_names	string	CF/Radial	Comma-delimited list of field names included in this file.

4.2 Global Dimensions

Dimension name	Description
sweep	The number of sweeps in the dataset
frequency	Number of frequencies used

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.
primary_axis		string		Options are: *"axis_z", "axis_y", "axis_x", "axis_z_prime", "axis_y_prime", "axis_x_prime".* See section 7 for details. Assumed "*axis_z*" if missing.

Variable name	Dimension	Type	Units	Comments
time_coverage_start		string		UTC time of first ray in file. Resolution is integer seconds. The **time(time)** variable is computed relative to this time. Format follows ISO 8601: yyyy-mm-ddThh:mm:ssZ NOTE: the T is optional, any single character may be used in this location.
time_coverage_end		string		UTC time of last ray in file. Resolution is integer seconds. Format is: yyyy-mm-ddThh:mm:ssZ NOTE: the T is optional, any single character may be used in this location.
time_reference		string		UTC time reference. Resolution is integer seconds. If defined, the **time(time)** variable is computed relative to this time instead of relative to **time_coverage_start**. Format is: yyyy-mm-ddThh:mm:ssZ NOTE: the T is optional, any single character may be used in this location.
latitude		double	degrees_north	Latitude of instrument. For a mobile platform, this is a latitude at the start of the volume.
longitude		double	degrees_east	Longitude of instrument. For a mobile platform, this is the longitude at the start of the volume.
altitude		double	meters	Altitude of instrument, above mean sea level. 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.

Variable name	Dimension	Type	Units	Comments
altitude_agl		double	meters	Altitude of instrument above ground level. Omit if not known. Omit for mobile platform.
sweep_group_names	(sweep)	string		Array of names for sweep groups. Allows the user to locate the sweep groups directly.
sweep_fixed_angles	(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. This is a repeat of the data in the sweep groups.
frequency	(frequency)	float	s-1	List of operating frequencies, in Hertz. In most cases, only a single frequency is used.
field_names		string		Comma-delimited list of field names included in this file.
status_xml		string		General-purpose XML string for storing any information that is not included in other parts of the data structure.

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_names* 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
n_prts	Number of prts used in pulsing scheme. Optional for fixed, staggered or dual Required for more complicated schemes.

5.2 Sweep coordinate variables

Variable name	Dimension	Type	Units	Comments
time	(time)	double	seconds	Coordinate variable for time. Time at center of each ray, in fractional seconds since time_coverage_start, or since time_reference if it exists.
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”
long_name	string	“time_in_seconds_since_volume_start”

Attribute name	Type	Value
units	string	"seconds since *yyyy-mm-dd*T*hh:mm:ss*Z", where the actual reference time values are used. This unit string is **very important** and must be correct. It should either match time_reference(if it exists) or time_coverage_start. NOTE: the T is optional, any single character may be used in this location.
calendar	string	Defaults to "gregorian" if missing. Options are: "gregorian" or "standard", "proleptic_gregorian", "noleap" 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: *"sector","coplane",rhi", "vertical_pointing","idle", "azimuth_surveillance", "elevation_surveillance", "sunscan","pointing", "manual_ppi","manual_rhi"*
follow_mode		string		options are: *"none", "sun", "vehicle", "aircraft", "target", "manual"* Assumed "*none*" if missing.
prt_mode		string		Pulsing mode Options are: *"fixed", "staggered", "dual", "hybrid"*. Assumed "*fixed*" if missing. May also be more complicated pulsing schemes, such as HHVV, HHVVH etc.
polarization_mode		string		Options are: *"horizontal", "vertical", "hv_alt", "hv_sim", "circular"* Assumed "*horizontal*" if missing.
polarization_sequence	(n_prts)	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.
fixed_angle		float	degrees	Target angle for the sweep. elevation in most modes azimuth in RHI mode
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.
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.

Variable name	Dimension	Type	Units	Comments
ray_angle_res		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.
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.
scan_rate	(time)	float	degrees/s	Actual antenna scan rate. Set to negative if counter-clockwise in azimuth or decreasing in elevation. Positive otherwise.
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	
r_calib_index	(time)	int		Index for the radar calibration that applies to this pulse width. See section 7.3.
rx_range_resolution	(time)	float	meters	Resolution of the raw receiver samples. If missing, assumed to be meters_between_gates (4.4.2). 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.

Variable name	Dimension	Type	Units	Comments
prt_sequence	(time, n_prts)	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, 4.10.
unambiguous_range	(time)	float	meters	Unambiguous range
n_samples	(time)	int		Number of samples used to compute moments

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 computed as 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"
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.
longitude	(time)	double	degrees_east	Longitude of instrument.
altitude	(time)	double	meters	Altitude of instrument, above mean sea level. 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.
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.

Variable name	Dimension	Type	Units	Comments
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.
georefs_applied	(time)	byte		1 if georeference information for mobile platforms has been applied to correct the azimuth and elevation. 0 otherwise. See section 4.9. Assumed 0 if missing.

5.5 The *monitoring* sub-group

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

The group will always be named '*monitoring'*.

Variable name	Dimension	Type	Units	Comments
radar_measured_transmit_power_h	(time)	float	dBm	Measured transmit power H polarization
radar_measured_transmit_power_v	(time)	float	dBm	Measured transmit power V polarization
radar_measured_sky_noise	(time)	float	dBm	Noise measured at the receiver when connected to the antenna with no noise source connected.

Variable name	Dimension	Type	Units	Comments
radar_measured_cold_noise	(time)	float	dBm	Noise measured at the receiver when connected to the noise source, but it is not enabled.
radar_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.

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
ncbyte	1	scaled signed integer
short	2	scaled signed integer
int	4	scaled signed integer
float	4	floating point
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 6.2.
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

Attribute name	Type	Convention	Description
_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.
scale_factor	float	CF	Float value = (integer value) * scale_factor + add_offset Only applies to integer types.
add_offset	float	CF	Float value = (integer value) * scale_factor + add_offset Only applies to integer types.
coordinates	string	CF	See note below
sampling_ratio	float	CF/Radial	Number of samples for this field divided by n_samples (see section 6.3). 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.
field_folds	string	CF/Radial	"true" or "false" 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.
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.

Attribute name	Type	Convention	Description
is_quality	string	CF/Radial	"true" or "false" "true" indicates this is a quality control field. If the attribute is not present, defaults to "false".
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. Space-separated list in string.
flag_masks	array of same type as field data	CF	Valid bit-wise masks used in a flag field that is comprised of bit-wise combinations of mask values. See flag_meanings.
qualified_variables	string	CF/Radial	Applicable if is_quality is "true". Space-separated list of variables that this variable qualifies. Every field variable in this list should list this variable in its ancillary_variable attribute.
ancillary_variables	string	CF	Space-separated 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 below for example)
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 below for example)

5.6.1 Use of scale_factor and add_offset

scale_factor and add_offset are required for ncbyte, short and int fields. They are not applicable to float and double fields.

float_value = (integer_value * scale_factor) + 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 1.6 above.

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 a space-delimited list of strings that indicate the meanings of each of the **flag_values**. If multi-word meanings are needed, use underscores to connect the words.

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 a space-delimited list of strings that indicate the meanings of each of the **flag_masks**. If multi-word meanings are needed, use underscores to connect the words.

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 a space-separated list of field names that this field qualifies.

Each qualified field, in turn, should have the **ancillary_variables** string attribute. This is a space-separated list of fields that qualify it.

5.6.6 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>
```

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>
```

6 Spectra 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 index variables

Spectrum index variables reside in the sweep group. There may be 0, 1 or more than 1.

The 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 (time, range), as for all field variables.

The _FillValue attribute for the index 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 (n_spectra – 1). This value indicates the location for the spectrum in the spectrum field data variable array.

A spectrum index variable must have the following attributes:

Attribute name	Type	Value
is_spectrum_index	string	"true"
spectra_group_name	string	name of sub-group containing the spectrum data

The 'is_spectrum_index' allows you to determine that a field is a spectrum index field.

The 'spectra_group_name' allows you to find the sub-group (within the sweep group) that holds the spectra data fields.

6.2 Spectra group dimensions

Dimension name	Description
n_spectra	Number of stored spectra in data set. These are for the gates where the index is not -1. This dimension name is not fixed. Any suitable name can be used. There will be multiple dimensions if different spectral data sets have different shapes.

Dimension name	Description
n_samples	Number of samples in the spectra. This dimension name is not fixed. Any suitable name can be used. There will be multiple dimensions if different spectral data sets have a different number of samples.

6.3 Spectrum fields

As indicated above, a spectrum variable will have the dimensions:

(n_spectra, n_samples).

As mentioned above, the dimension names are not fixed. Any suitable names can be used. There will be multiple dimensions if different spectral data sets have different shapes.

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

6.4 Spectrum field attributes

Spectrum fields have the following attributes:

Attribute name	Type	Convention	Description
is_spectrum	string	CF/Radial	Set to “true” to indicate that this is a spectrum variable.
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.
units	string	CF	Units for field

Attribute name	Type	Convention	Description
_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 6.6.. Float value = (integer value) * scale_factor + add_offset. Only applied to integer types.
add_offset	float	CF	See section 6.6.. Float value = (integer value) * scale_factor + add_offset. Only applied to integer types.
coordinates	string	CF	See section 5.6.2.
index_var_name	string	CF/Radial	Name of index variable for this field. NOTE: a single index variable may be used for multiple spectrum fields.
fft_length	int	CF/Radial	Length of fft used to compute this spectrum
block_avg_length	int	CF/Radial	Length of averaging block used to compute this 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
radar_antenna_gain_h	none	float	dB	Nominal antenna gain, H polarization
radar_antenna_gain_v	none	float	dB	Nominal antenna gain, V polarization
radar_beam_width_h	none	float	degrees	Antenna beam width H polarization
radar_beam_width_v	none	float	degrees	Antenna beam width V polarization
radar_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
lidar_beam_divergence	none	float	milliradians	Transmit side
lidar_field_of_view	none	float	milliradians	Receive side
lidar_aperture_diameter	none	float	cm	
lidar_aperture_efficiency	none	float	percent	
lidar_peak_power	none	float	watts	
lidar_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
r_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
r_calib_index	(time)	byte		Index for the calibration that applies to each ray. Assumed 0 if missing.
time	(r_calib, string_length)	string	UTC	e.g. 2008-09-25 T23:00:00Z
pulse_width	(r_calib)	float	seconds	Pulse width for this calibration
antenna_gain_h	(r_calib)	float	dB	Derived antenna gain H channel
antenna_gain_v	(r_calib)	float	dB	Derived antenna gain V channel
xmit_power_h	(r_calib)	float	dBm	Transmit power H channel

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

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

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

The following additional variables are used to quantify errors in the georeference data for moving platforms. 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.

8.1 Standard names for moments variables

This section lists the proposed standard names for moments data and other fields derived from the raw radar data.

Standard name	Short name	Units	Already in CF?
equivalent_reflectivity_factor	DBZ	dBZ	yes
linear_equivalent_reflectivity_factor	Z	Z	no
radial_velocity_of_scatterers_ away_from_instrument	VEL	m/s	yes
doppler_spectrum_width	WIDTH	m/s	no
log_differential_reflectivity_hv	ZDR	dB	no
log_linear_depolarization_ratio_hv	LDR	dB	no
log_linear_depolarization_ratio_h	LDRH	dB	no
log_linear_depolarization_ratio_v	LDRV	dB	no
differential_phase_hv	PHIDP	degrees	no
specific_differential_phase_hv	KDP	degrees/km	no
cross_polar_differential_phase	PHIHX	degrees	no
cross_correlation_ratio_hv	RHOHV		no
co_to_cross_polar_correlation_ratio_h	RHOHX		no
co_to_cross_polar_correlation_ratio_v	RHOXV		no
log_power	DBM	dBm	no
log_power_co_polar_h	DBMHC	dBm	no
log_power_cross_polar_h	DBMHX	dBm	no
log_power_co_polar_v	DBMVC	dBm	no
log_power_cross_polar_v	DBMVX	dBm	no
linear_power	PWR	mW	no
linear_power_co_polar_h	PWRHC	mW	no
linear_power_cross_polar_h	PWRHX	mW	no
linear_power_co_polar_v	PWRVC	mW	no

Standard name	Short name	Units	Already in CF?
linear_power_cross_polar_v	PWRVX	mW	no
signal_to_noise_ratio	SNR	dB	no
signal_to_noise_ratio_co_polar_h	SNRHC	dB	no
signal_to_noise_ratio_cross_polar_h	SNRHX	dB	no
signal_to_noise_ratio_co_polar_v	SNRVC	dB	no
signal_to_noise_ratio_cross_polar_v	SNRVX	dB	no
normalized_coherent_power (Note: this is also known as signal-quality-index)	NCP (SQI)		no
corrected_equivalent_reflectivity_factor	DBZc	dBZ	no
corrected_radial_velocity_of_scatterers_ away_from_instrument	VELc	m/s	no
corrected_log_differential_reflectivity_hv	ZDRc	dB	no
radar_estimated_rain_rate	RRR	mm/hr	no
rain_rate	RR	kg/m2/s	yes
radar_echo_classification (should be used for PID, HCA, HID etc)	REC	legend	no

8.2 Standard names for spectra variables

This section lists the proposed standard names for spectra field variables.

Standard name	Suggested short name	Units	Already in CF?
spectrum_copolar_horizontal	SPEC_HH_HH*		no
spectrum_copolar_vertical	SPEC_VV_VV*		no
spectrum_crosspolar_horizontal	SPEC_VH_VH*		no
spectrum_crosspolar_vertical	SPEC_HV_HV*		no
cross_spectrum_of_copolar_horizontal	XSPEC_HH_VV*		no
cross_spectrum_of_copolar_verticall	XSPEC_VV_HH*		no
cross_spectrum_of_crosspolar_horizontal	XSPEC_HH_VH*		no
cross_spectrum_of_crosspolar_verticall	XSPEC_VV_HV*		no

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 (λ), and elevation (ϕ). A ground-based radar is assumed "leveled" with positive (negative) elevation, ϕ, above (below) a *reference plane* (a leveled plane orthogonal to the principal axis and containing the radar). The azimuth angle, λ, is the angle on the reference plane increases clockwise from the True North (TN) following the Meteorological coordinate convention (e.g., TN is 0° and East is 90°).

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 **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 **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.

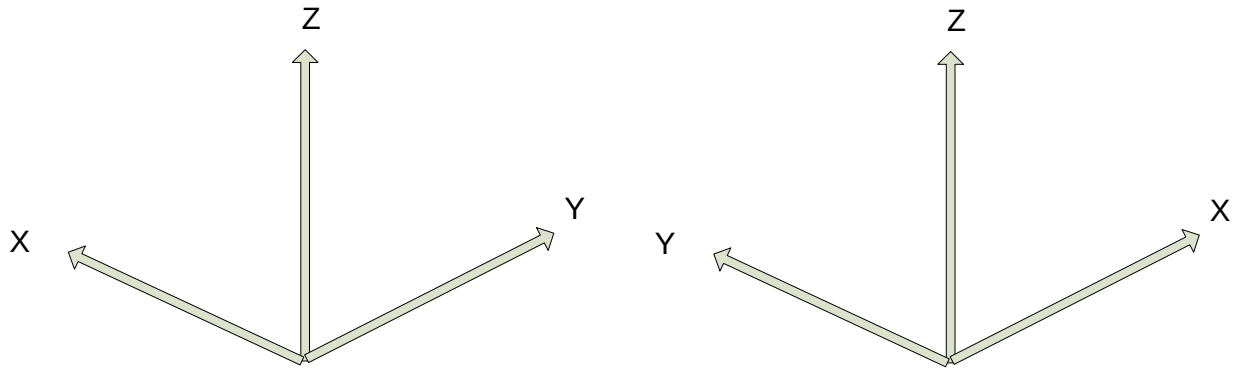

Figure 9.1: Left-handed XYZ coordinate system vs. Right-handed XYZ coordinate system.

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 (λ) are positive clockwise looking from above (+Z), with 0 being TN.

Elevation angles (ϕ) 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 ϕ=90°.

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, λ, ϕ) 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 (λ) 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 $R' = \left(4/3\right) \bullet 6374 km$ 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_{air}, V_{air}, W_{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 (ϕ) and azimuth (λ) 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, **X**, is defined as follows, X is East, Y is North, and Z is zenith. Azimuth angle, λ, is defined as positive *clockwise* from TN (i.e., meteorological angle) while elevation angle, ϕ, 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 **X**. These angles are generally measured by an inertial navigation system (INS).

The platform moves relative to **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.

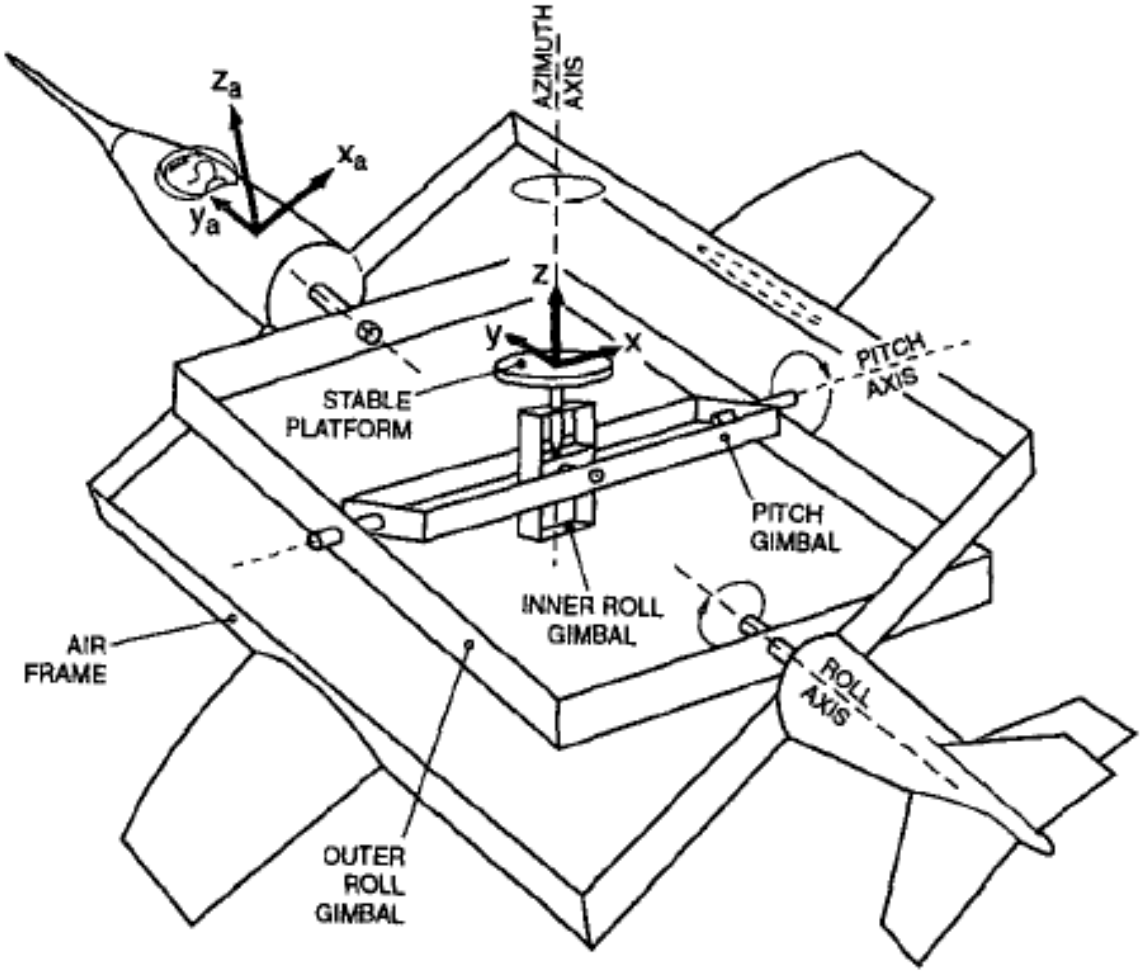

Figure 9.2 Moving platform axis definitions and reference frame (reproduced from Lee et al., 1994,originally from Axford, 1968) ©American Meteorological Society. Reprinted with permission.

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

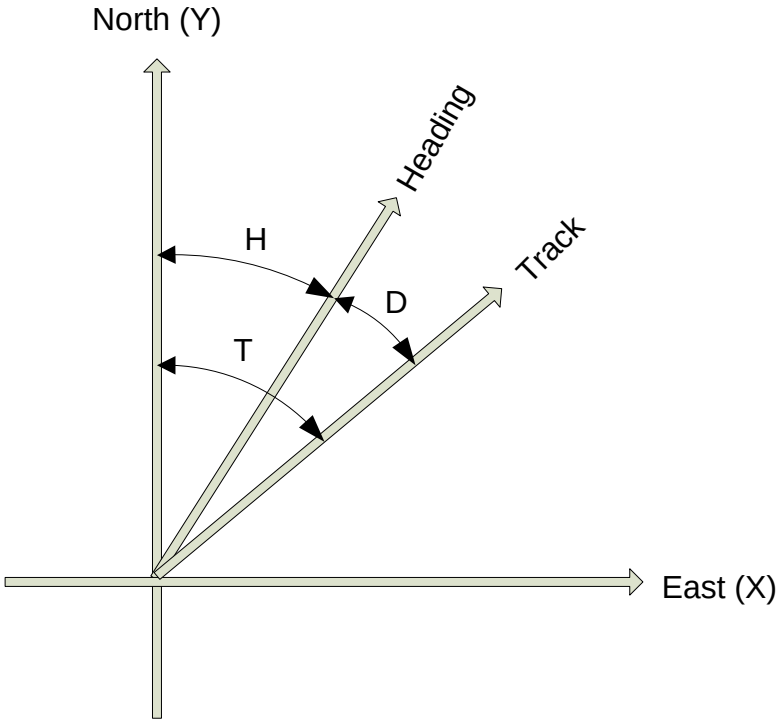

Figure 7.3(a): Definition of heading, drift and track.

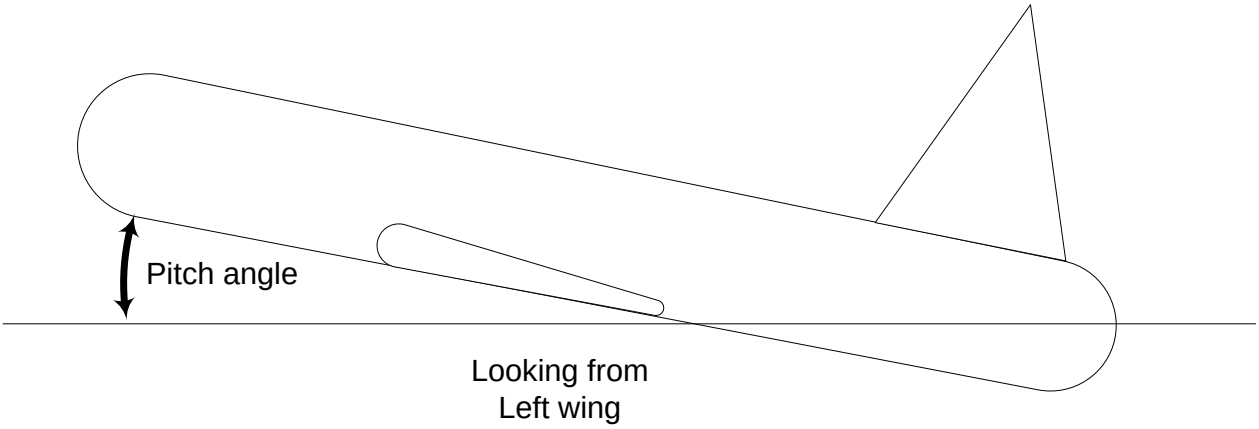

Figure 7.3(b): Definition of pitch

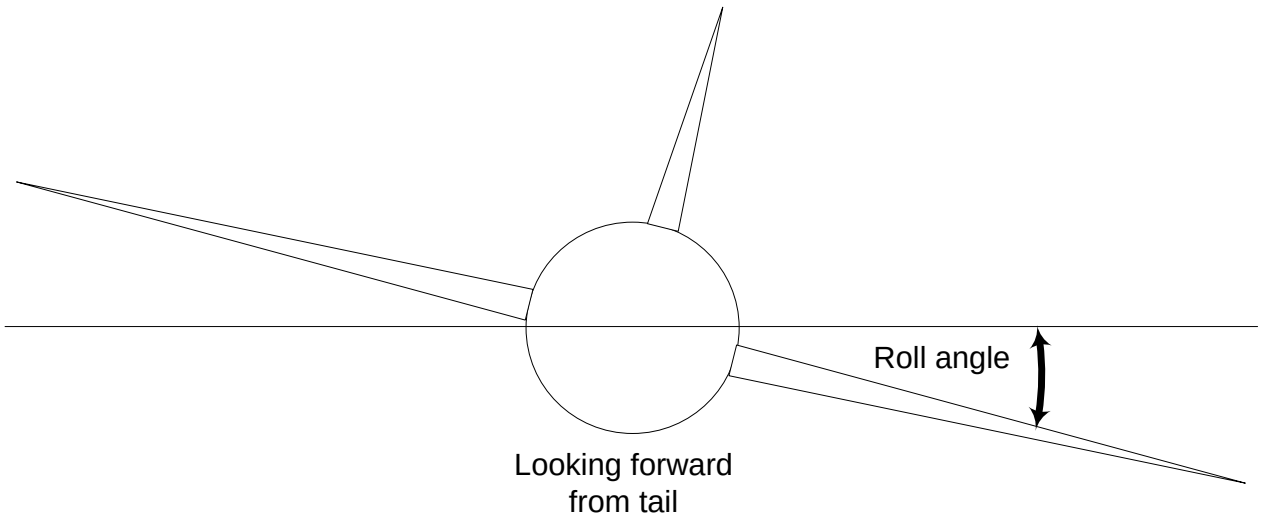

Figure 7.3(c): Definition of roll

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, θ, and a tilt angle, τ. Following the ground-based radar convention, the rotation angle, θ, 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, τ, is the angle of the beam relative to the reference plane. A beam has a positive/negative τ 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, θ, τ) in $\mathbf{X}_i$ can be represented in (r, λ, ϕ) 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	Ya	Z_a
Reference Plane	Y_aZ_a	Z_aX_a	ZaXa	X_aY_a
0° Rotation Angle	$+Z_a$	$+X_a$	+Za	$+Y_a$
90° Rotation Angle	$+Y_a$	$+Z_a$	+Xa	$+X_a$
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 X_i to 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 θ 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 τ is 0 in the (X_a, Y_a) plane, positive above it (for $+Z_a$) and negative below it.

The transformation to 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 θ 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 τ is 0 in the (Z_a, X_a) plane, positive for $+Y_a$.

Note that the definition of θ is different from the convention defined in Lee et al. (1994)[1]. Let θ' be the rotation angle defined in Lee et al. (1994), θ=mod(450°- θ').

[1] The rotation angle, θ', 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. θ'=0° 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 θ is 0 in the (Ya, Za) plane, i.e. along the $+Z_a$ axis. Rotation increases clockwise from +Z, when looking from -Y.

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

Note that the definition of θ 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 θ 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 τ is 0 in the (Y_a, Z_a) plane, positive for $+X_a$.

The transformation to 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 X

Rotating $\mathbf{X}_a$ to **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 τ 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:

$$X = M_H M_P M_R X_a$$

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

$$= \begin{pmatrix} \cos H \cos R + \sin H \sin P \sin R & \sin H \cos P & \cos H \sin R - \sin H \sin P \cos R \\ -\sin H \cos R + \cos H \sin P \sin R & \cos H \cos P & -\sin H \sin R - \cos H \sin P \cos R \\ -\cos P \sin R & \sin P & \cos P \cos R \end{pmatrix}$$

$$= \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

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, θ, τ) or (r, λ, ϕ)

$\mathbf{X}_a$: 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)

λ: azimuth angle

ϕ: elevation angle

θ: rotation angle

τ: 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.

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.

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

11 Example ncdump of CfRadial file

The following is an example ncdump from a valid CfRadial file, for the KDDC NEXRAD radar, at 12:04:15 on 2015/06/26:

```
netcdf
cfrad.20150626_120415.982_to_20150626_120831.578_KDDC_Surveillance_SUR {
dimensions:
    time = 4200 ;
    range = 1832 ;
    n_points = 6087840 ;
    sweep = 9 ;
    string_length_8 = 8 ;
    string_length_32 = 32 ;
    status_xml_length = 1 ;
    r_calib = 1 ;
    frequency = 1 ;
variables:
    int volume_number ;
       volume_number:long_name = "data_volume_index_number" ;
       volume_number:units = "" ;
       volume_number:_FillValue = -9999 ;
    char platform_type(string_length_32) ;
       platform_type:long_name = "platform_type" ;
       platform_type:options = "fixed, vehicle, ship, aircraft_fore,
aircraft_aft, aircraft_tail, aircraft_belly, aircraft_roof, aircraft_nose,
satellite_orbit, satellite_geostat" ;
    char primary_axis(string_length_32) ;
       primary_axis:long_name = "primary_axis_of_rotation" ;
       primary_axis:options = "axis_z, axis_y, axis_x, axis_z_prime,
axis_y_prime, axis_x_prime" ;
    char status_xml(status_xml_length) ;
       status_xml:long_name = "status_of_instrument" ;
    char instrument_type(string_length_32) ;
       instrument_type:long_name = "type_of_instrument" ;
       instrument_type:options = "radar, lidar" ;
       instrument_type:meta_group = "instrument_parameters" ;
    float radar_antenna_gain_h ;
       radar_antenna_gain_h:long_name =
"nominal_radar_antenna_gain_h_channel" ;
       radar_antenna_gain_h:units = "db" ;
       radar_antenna_gain_h:_FillValue = -9999.f ;
       radar_antenna_gain_h:meta_group = "radar_parameters" ;
    float radar_antenna_gain_v ;
       radar_antenna_gain_v:long_name =
"nominal_radar_antenna_gain_v_channel" ;
       radar_antenna_gain_v:units = "db" ;
       radar_antenna_gain_v:_FillValue = -9999.f ;
       radar_antenna_gain_v:meta_group = "radar_parameters" ;
    float radar_beam_width_h ;
       radar_beam_width_h:long_name =
"half_power_radar_beam_width_h_channel" ;
```

```
      radar_beam_width_h:units = "degrees" ;
      radar_beam_width_h:_FillValue = -9999.f ;
      radar_beam_width_h:meta_group = "radar_parameters" ;
   float radar_beam_width_v ;
      radar_beam_width_v:long_name =
"half_power_radar_beam_width_v_channel" ;
      radar_beam_width_v:units = "degrees" ;
      radar_beam_width_v:_FillValue = -9999.f ;
      radar_beam_width_v:meta_group = "radar_parameters" ;
   float radar_rx_bandwidth ;
      radar_rx_bandwidth:long_name = "radar_receiver_bandwidth" ;
      radar_rx_bandwidth:units = "s-1" ;
      radar_rx_bandwidth:_FillValue = -9999.f ;
      radar_rx_bandwidth:meta_group = "radar_parameters" ;
   char time_coverage_start(string_length_32) ;
      time_coverage_start:long_name = "data_volume_start_time_utc" ;
      time_coverage_start:comment = "ray times are relative to start time
in secs" ;
   char time_coverage_end(string_length_32) ;
      time_coverage_end:long_name = "data_volume_end_time_utc" ;
   float frequency(frequency) ;
      frequency:long_name = "transmission_frequency" ;
      frequency:units = "s-1" ;
      frequency:_FillValue = -9999.f ;
      frequency:meta_group = "instrument_parameters" ;
   int grid_mapping ;
      grid_mapping:grid_mapping_name = "radar_lidar_radial_scan" ;
      grid_mapping:longitude_of_projection_origin = -99.9688873291016 ;
      grid_mapping:latitude_of_projection_origin = 37.7608337402344 ;
      grid_mapping:altitude_of_projection_origin = 813. ;
      grid_mapping:false_northing = 0. ;
      grid_mapping:false_easting = 0. ;
   double latitude ;
      latitude:long_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:_FillValue = -9999. ;
   double longitude ;
      longitude:long_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:_FillValue = -9999. ;
   double altitude ;
      altitude:long_name = "altitude" ;
      altitude:units = "meters" ;
      altitude:_FillValue = -9999. ;
      altitude:positive = "up" ;
   double altitude_agl ;
      altitude_agl:long_name = "altitude_above_ground_level" ;
      altitude_agl:units = "meters" ;
      altitude_agl:_FillValue = -9999. ;
      altitude_agl:positive = "up" ;
   int sweep_number(sweep) ;
      sweep_number:long_name = "sweep_index_number_0_based" ;
      sweep_number:units = "" ;
      sweep_number:_FillValue = -9999 ;
```

```
   char sweep_mode(sweep, string_length_32) ;
      sweep_mode:long_name = "scan_mode_for_sweep" ;
      sweep_mode:options = "sector, coplane, rhi, vertical_pointing, idle,
azimuth_surveillance, elevation_surveillance, sunscan, pointing,
calibration, manual_ppi, manual_rhi, sunscan_rhi" ;
   char polarization_mode(sweep, string_length_32) ;
      polarization_mode:long_name = "polarization_mode_for_sweep" ;
      polarization_mode:options = "horizontal, vertical, hv_alt, hv_sim,
circular" ;
      polarization_mode:meta_group = "radar_parameters" ;
   char prt_mode(sweep, string_length_32) ;
      prt_mode:long_name = "transmit_pulse_mode" ;
      prt_mode:options = "fixed, staggered, dual" ;
      prt_mode:meta_group = "radar_parameters" ;
   char follow_mode(sweep, string_length_32) ;
      follow_mode:long_name = "follow_mode_for_scan_strategy" ;
      follow_mode:options = "none, sun, vehicle, aircraft, target, manual"
;
      follow_mode:meta_group = "instrument_parameters" ;
   float fixed_angle(sweep) ;
      fixed_angle:long_name = "ray_target_fixed_angle" ;
      fixed_angle:units = "degrees" ;
      fixed_angle:_FillValue = -9999.f ;
   float target_scan_rate(sweep) ;
      target_scan_rate:long_name = "target_scan_rate_for_sweep" ;
      target_scan_rate:units = "degrees per second" ;
      target_scan_rate:_FillValue = -9999.f ;
   int sweep_start_ray_index(sweep) ;
      sweep_start_ray_index:long_name = "index_of_first_ray_in_sweep" ;
      sweep_start_ray_index:units = "" ;
      sweep_start_ray_index:_FillValue = -9999 ;
   int sweep_end_ray_index(sweep) ;
      sweep_end_ray_index:long_name = "index_of_last_ray_in_sweep" ;
      sweep_end_ray_index:units = "" ;
      sweep_end_ray_index:_FillValue = -9999 ;
   char rays_are_indexed(sweep, string_length_8) ;
      rays_are_indexed:long_name = "flag_for_indexed_rays" ;
   float ray_angle_res(sweep) ;
      ray_angle_res:long_name = "angular_resolution_between_rays" ;
      ray_angle_res:units = "degrees" ;
      ray_angle_res:_FillValue = -9999.f ;
   char r_calib_time(r_calib, string_length_32) ;
      r_calib_time:long_name = "radar_calibration_time_utc" ;
      r_calib_time:meta_group = "radar_calibration" ;
   float r_calib_pulse_width(r_calib) ;
      r_calib_pulse_width:long_name = "radar_calibration_pulse_width" ;
      r_calib_pulse_width:units = "seconds" ;
      r_calib_pulse_width:meta_group = "radar_calibration" ;
      r_calib_pulse_width:_FillValue = -9999.f ;
   float r_calib_xmit_power_h(r_calib) ;
      r_calib_xmit_power_h:long_name =
"calibrated_radar_xmit_power_h_channel" ;
      r_calib_xmit_power_h:units = "dBm" ;
      r_calib_xmit_power_h:meta_group = "radar_calibration" ;
```

```
      r_calib_xmit_power_h:_FillValue = -9999.f ;
   float r_calib_xmit_power_v(r_calib) ;
      r_calib_xmit_power_v:long_name =
"calibrated_radar_xmit_power_v_channel" ;
      r_calib_xmit_power_v:units = "dBm" ;
      r_calib_xmit_power_v:meta_group = "radar_calibration" ;
      r_calib_xmit_power_v:_FillValue = -9999.f ;
   float r_calib_two_way_waveguide_loss_h(r_calib) ;
      r_calib_two_way_waveguide_loss_h:long_name =
"radar_calibration_two_way_waveguide_loss_h_channel" ;
      r_calib_two_way_waveguide_loss_h:units = "db" ;
      r_calib_two_way_waveguide_loss_h:meta_group = "radar_calibration" ;
      r_calib_two_way_waveguide_loss_h:_FillValue = -9999.f ;
   float r_calib_two_way_waveguide_loss_v(r_calib) ;
      r_calib_two_way_waveguide_loss_v:long_name =
"radar_calibration_two_way_waveguide_loss_v_channel" ;
      r_calib_two_way_waveguide_loss_v:units = "db" ;
      r_calib_two_way_waveguide_loss_v:meta_group = "radar_calibration" ;
      r_calib_two_way_waveguide_loss_v:_FillValue = -9999.f ;
   float r_calib_two_way_radome_loss_h(r_calib) ;
      r_calib_two_way_radome_loss_h:long_name =
"radar_calibration_two_way_radome_loss_h_channel" ;
      r_calib_two_way_radome_loss_h:units = "db" ;
      r_calib_two_way_radome_loss_h:meta_group = "radar_calibration" ;
      r_calib_two_way_radome_loss_h:_FillValue = -9999.f ;
   float r_calib_two_way_radome_loss_v(r_calib) ;
      r_calib_two_way_radome_loss_v:long_name =
"radar_calibration_two_way_radome_loss_v_channel" ;
      r_calib_two_way_radome_loss_v:units = "db" ;
      r_calib_two_way_radome_loss_v:meta_group = "radar_calibration" ;
      r_calib_two_way_radome_loss_v:_FillValue = -9999.f ;
   float r_calib_receiver_mismatch_loss(r_calib) ;
      r_calib_receiver_mismatch_loss:long_name =
"radar_calibration_receiver_mismatch_loss" ;
      r_calib_receiver_mismatch_loss:units = "db" ;
      r_calib_receiver_mismatch_loss:meta_group = "radar_calibration" ;
      r_calib_receiver_mismatch_loss:_FillValue = -9999.f ;
   float r_calib_radar_constant_h(r_calib) ;
      r_calib_radar_constant_h:long_name =
"calibrated_radar_constant_h_channel" ;
      r_calib_radar_constant_h:units = "db" ;
      r_calib_radar_constant_h:meta_group = "radar_calibration" ;
      r_calib_radar_constant_h:_FillValue = -9999.f ;
   float r_calib_radar_constant_v(r_calib) ;
      r_calib_radar_constant_v:long_name =
"calibrated_radar_constant_v_channel" ;
      r_calib_radar_constant_v:units = "db" ;
      r_calib_radar_constant_v:meta_group = "radar_calibration" ;
      r_calib_radar_constant_v:_FillValue = -9999.f ;
   float r_calib_antenna_gain_h(r_calib) ;
      r_calib_antenna_gain_h:long_name =
"calibrated_radar_antenna_gain_h_channel" ;
      r_calib_antenna_gain_h:units = "db" ;
      r_calib_antenna_gain_h:meta_group = "radar_calibration" ;
```

```
      r_calib_antenna_gain_h:_FillValue = -9999.f ;
   float r_calib_antenna_gain_v(r_calib) ;
      r_calib_antenna_gain_v:long_name =
"calibrated_radar_antenna_gain_v_channel" ;
      r_calib_antenna_gain_v:units = "db" ;
      r_calib_antenna_gain_v:meta_group = "radar_calibration" ;
      r_calib_antenna_gain_v:_FillValue = -9999.f ;
   float r_calib_noise_hc(r_calib) ;
      r_calib_noise_hc:long_name =
"calibrated_radar_receiver_noise_h_co_polar_channel" ;
      r_calib_noise_hc:units = "dBm" ;
      r_calib_noise_hc:meta_group = "radar_calibration" ;
      r_calib_noise_hc:_FillValue = -9999.f ;
   float r_calib_noise_vc(r_calib) ;
      r_calib_noise_vc:long_name =
"calibrated_radar_receiver_noise_v_co_polar_channel" ;
      r_calib_noise_vc:units = "dBm" ;
      r_calib_noise_vc:meta_group = "radar_calibration" ;
      r_calib_noise_vc:_FillValue = -9999.f ;
   float r_calib_noise_hx(r_calib) ;
      r_calib_noise_hx:long_name =
"calibrated_radar_receiver_noise_h_cross_polar_channel" ;
      r_calib_noise_hx:units = "dBm" ;
      r_calib_noise_hx:meta_group = "radar_calibration" ;
      r_calib_noise_hx:_FillValue = -9999.f ;
   float r_calib_noise_vx(r_calib) ;
      r_calib_noise_vx:long_name =
"calibrated_radar_receiver_noise_v_cross_polar_channel" ;
      r_calib_noise_vx:units = "dBm" ;
      r_calib_noise_vx:meta_group = "radar_calibration" ;
      r_calib_noise_vx:_FillValue = -9999.f ;
   float r_calib_receiver_gain_hc(r_calib) ;
      r_calib_receiver_gain_hc:long_name =
"calibrated_radar_receiver_gain_h_co_polar_channel" ;
      r_calib_receiver_gain_hc:units = "db" ;
      r_calib_receiver_gain_hc:meta_group = "radar_calibration" ;
      r_calib_receiver_gain_hc:_FillValue = -9999.f ;
   float r_calib_receiver_gain_vc(r_calib) ;
      r_calib_receiver_gain_vc:long_name =
"calibrated_radar_receiver_gain_v_co_polar_channel" ;
      r_calib_receiver_gain_vc:units = "db" ;
      r_calib_receiver_gain_vc:meta_group = "radar_calibration" ;
      r_calib_receiver_gain_vc:_FillValue = -9999.f ;
   float r_calib_receiver_gain_hx(r_calib) ;
      r_calib_receiver_gain_hx:long_name =
"calibrated_radar_receiver_gain_h_cross_polar_channel" ;
      r_calib_receiver_gain_hx:units = "db" ;
      r_calib_receiver_gain_hx:meta_group = "radar_calibration" ;
      r_calib_receiver_gain_hx:_FillValue = -9999.f ;
   float r_calib_receiver_gain_vx(r_calib) ;
      r_calib_receiver_gain_vx:long_name =
"calibrated_radar_receiver_gain_v_cross_polar_channel" ;
      r_calib_receiver_gain_vx:units = "db" ;
      r_calib_receiver_gain_vx:meta_group = "radar_calibration" ;
```

```
      r_calib_receiver_gain_vx:_FillValue = -9999.f ;
   float r_calib_base_dbz_1km_hc(r_calib) ;
      r_calib_base_dbz_1km_hc:long_name =
"radar_reflectivity_at_1km_at_zero_snr_h_co_polar_channel" ;
      r_calib_base_dbz_1km_hc:units = "dBZ" ;
      r_calib_base_dbz_1km_hc:meta_group = "radar_calibration" ;
      r_calib_base_dbz_1km_hc:_FillValue = -9999.f ;
   float r_calib_base_dbz_1km_vc(r_calib) ;
      r_calib_base_dbz_1km_vc:long_name =
"radar_reflectivity_at_1km_at_zero_snr_v_co_polar_channel" ;
      r_calib_base_dbz_1km_vc:units = "dBZ" ;
      r_calib_base_dbz_1km_vc:meta_group = "radar_calibration" ;
      r_calib_base_dbz_1km_vc:_FillValue = -9999.f ;
   float r_calib_base_dbz_1km_hx(r_calib) ;
      r_calib_base_dbz_1km_hx:long_name =
"radar_reflectivity_at_1km_at_zero_snr_h_cross_polar_channel" ;
      r_calib_base_dbz_1km_hx:units = "dBZ" ;
      r_calib_base_dbz_1km_hx:meta_group = "radar_calibration" ;
      r_calib_base_dbz_1km_hx:_FillValue = -9999.f ;
   float r_calib_base_dbz_1km_vx(r_calib) ;
      r_calib_base_dbz_1km_vx:long_name =
"radar_reflectivity_at_1km_at_zero_snr_v_cross_polar_channel" ;
      r_calib_base_dbz_1km_vx:units = "dBZ" ;
      r_calib_base_dbz_1km_vx:meta_group = "radar_calibration" ;
      r_calib_base_dbz_1km_vx:_FillValue = -9999.f ;
   float r_calib_sun_power_hc(r_calib) ;
      r_calib_sun_power_hc:long_name =
"calibrated_radar_sun_power_h_co_polar_channel" ;
      r_calib_sun_power_hc:units = "dBm" ;
      r_calib_sun_power_hc:meta_group = "radar_calibration" ;
      r_calib_sun_power_hc:_FillValue = -9999.f ;
   float r_calib_sun_power_vc(r_calib) ;
      r_calib_sun_power_vc:long_name =
"calibrated_radar_sun_power_v_co_polar_channel" ;
      r_calib_sun_power_vc:units = "dBm" ;
      r_calib_sun_power_vc:meta_group = "radar_calibration" ;
      r_calib_sun_power_vc:_FillValue = -9999.f ;
   float r_calib_sun_power_hx(r_calib) ;
      r_calib_sun_power_hx:long_name =
"calibrated_radar_sun_power_h_cross_polar_channel" ;
      r_calib_sun_power_hx:units = "dBm" ;
      r_calib_sun_power_hx:meta_group = "radar_calibration" ;
      r_calib_sun_power_hx:_FillValue = -9999.f ;
   float r_calib_sun_power_vx(r_calib) ;
      r_calib_sun_power_vx:long_name =
"calibrated_radar_sun_power_v_cross_polar_channel" ;
      r_calib_sun_power_vx:units = "dBm" ;
      r_calib_sun_power_vx:meta_group = "radar_calibration" ;
      r_calib_sun_power_vx:_FillValue = -9999.f ;
   float r_calib_noise_source_power_h(r_calib) ;
      r_calib_noise_source_power_h:long_name =
"radar_calibration_noise_source_power_h_channel" ;
      r_calib_noise_source_power_h:units = "dBm" ;
      r_calib_noise_source_power_h:meta_group = "radar_calibration" ;
```

```
      r_calib_noise_source_power_h:_FillValue = -9999.f ;
   float r_calib_noise_source_power_v(r_calib) ;
      r_calib_noise_source_power_v:long_name =
"radar_calibration_noise_source_power_v_channel" ;
      r_calib_noise_source_power_v:units = "dBm" ;
      r_calib_noise_source_power_v:meta_group = "radar_calibration" ;
      r_calib_noise_source_power_v:_FillValue = -9999.f ;
   float r_calib_power_measure_loss_h(r_calib) ;
      r_calib_power_measure_loss_h:long_name =
"radar_calibration_power_measurement_loss_h_channel" ;
      r_calib_power_measure_loss_h:units = "db" ;
      r_calib_power_measure_loss_h:meta_group = "radar_calibration" ;
      r_calib_power_measure_loss_h:_FillValue = -9999.f ;
   float r_calib_power_measure_loss_v(r_calib) ;
      r_calib_power_measure_loss_v:long_name =
"radar_calibration_power_measurement_loss_v_channel" ;
      r_calib_power_measure_loss_v:units = "db" ;
      r_calib_power_measure_loss_v:meta_group = "radar_calibration" ;
      r_calib_power_measure_loss_v:_FillValue = -9999.f ;
   float r_calib_coupler_forward_loss_h(r_calib) ;
      r_calib_coupler_forward_loss_h:long_name =
"radar_calibration_coupler_forward_loss_h_channel" ;
      r_calib_coupler_forward_loss_h:units = "db" ;
      r_calib_coupler_forward_loss_h:meta_group = "radar_calibration" ;
      r_calib_coupler_forward_loss_h:_FillValue = -9999.f ;
   float r_calib_coupler_forward_loss_v(r_calib) ;
      r_calib_coupler_forward_loss_v:long_name =
"radar_calibration_coupler_forward_loss_v_channel" ;
      r_calib_coupler_forward_loss_v:units = "db" ;
      r_calib_coupler_forward_loss_v:meta_group = "radar_calibration" ;
      r_calib_coupler_forward_loss_v:_FillValue = -9999.f ;
   float r_calib_dbz_correction(r_calib) ;
      r_calib_dbz_correction:long_name = "calibrated_radar_dbz_correction"
;
      r_calib_dbz_correction:units = "db" ;
      r_calib_dbz_correction:meta_group = "radar_calibration" ;
      r_calib_dbz_correction:_FillValue = -9999.f ;
   float r_calib_zdr_correction(r_calib) ;
      r_calib_zdr_correction:long_name = "calibrated_radar_zdr_correction"
;
      r_calib_zdr_correction:units = "db" ;
      r_calib_zdr_correction:meta_group = "radar_calibration" ;
      r_calib_zdr_correction:_FillValue = -9999.f ;
   float r_calib_ldr_correction_h(r_calib) ;
      r_calib_ldr_correction_h:long_name =
"calibrated_radar_ldr_correction_h_channel" ;
      r_calib_ldr_correction_h:units = "db" ;
      r_calib_ldr_correction_h:meta_group = "radar_calibration" ;
      r_calib_ldr_correction_h:_FillValue = -9999.f ;
   float r_calib_ldr_correction_v(r_calib) ;
      r_calib_ldr_correction_v:long_name =
"calibrated_radar_ldr_correction_v_channel" ;
      r_calib_ldr_correction_v:units = "db" ;
      r_calib_ldr_correction_v:meta_group = "radar_calibration" ;
```

```
      r_calib_ldr_correction_v:_FillValue = -9999.f ;
   float r_calib_system_phidp(r_calib) ;
      r_calib_system_phidp:long_name = "calibrated_radar_system_phidp" ;
      r_calib_system_phidp:units = "degrees" ;
      r_calib_system_phidp:meta_group = "radar_calibration" ;
      r_calib_system_phidp:_FillValue = -9999.f ;
   float r_calib_test_power_h(r_calib) ;
      r_calib_test_power_h:long_name =
"radar_calibration_test_power_h_channel" ;
      r_calib_test_power_h:units = "dBm" ;
      r_calib_test_power_h:meta_group = "radar_calibration" ;
      r_calib_test_power_h:_FillValue = -9999.f ;
   float r_calib_test_power_v(r_calib) ;
      r_calib_test_power_v:long_name =
"radar_calibration_test_power_v_channel" ;
      r_calib_test_power_v:units = "dBm" ;
      r_calib_test_power_v:meta_group = "radar_calibration" ;
      r_calib_test_power_v:_FillValue = -9999.f ;
   float r_calib_receiver_slope_hc(r_calib) ;
      r_calib_receiver_slope_hc:long_name =
"calibrated_radar_receiver_slope_h_co_polar_channel" ;
      r_calib_receiver_slope_hc:units = "" ;
      r_calib_receiver_slope_hc:meta_group = "radar_calibration" ;
      r_calib_receiver_slope_hc:_FillValue = -9999.f ;
   float r_calib_receiver_slope_vc(r_calib) ;
      r_calib_receiver_slope_vc:long_name =
"calibrated_radar_receiver_slope_v_co_polar_channel" ;
      r_calib_receiver_slope_vc:units = "" ;
      r_calib_receiver_slope_vc:meta_group = "radar_calibration" ;
      r_calib_receiver_slope_vc:_FillValue = -9999.f ;
   float r_calib_receiver_slope_hx(r_calib) ;
      r_calib_receiver_slope_hx:long_name =
"calibrated_radar_receiver_slope_h_cross_polar_channel" ;
      r_calib_receiver_slope_hx:units = "" ;
      r_calib_receiver_slope_hx:meta_group = "radar_calibration" ;
      r_calib_receiver_slope_hx:_FillValue = -9999.f ;
   float r_calib_receiver_slope_vx(r_calib) ;
      r_calib_receiver_slope_vx:long_name =
"calibrated_radar_receiver_slope_v_cross_polar_channel" ;
      r_calib_receiver_slope_vx:units = "" ;
      r_calib_receiver_slope_vx:meta_group = "radar_calibration" ;
      r_calib_receiver_slope_vx:_FillValue = -9999.f ;
   double time(time) ;
      time:standard_name = "time" ;
      time:long_name = "time in seconds since volume start" ;
      time:calendar = "gregorian" ;
      time:units = "seconds since 2015-06-26T12:04:15Z" ;
      time:comment = "times are relative to the volume start_time" ;
   float range(range) ;
      range:long_name = "Range from instrument to center of gate" ;
      range:units = "meters" ;
      range:spacing_is_constant = "true" ;
      range:meters_to_center_of_first_gate = 2125. ;
      range:meters_between_gates = 250. ;
```

```
   int ray_n_gates(time) ;
      ray_n_gates:long_name = "number_of_gates" ;
      ray_n_gates:units = "" ;
      ray_n_gates:_FillValue = -9999 ;
   int ray_start_index(time) ;
      ray_start_index:long_name = "array_index_to_start_of_ray" ;
      ray_start_index:units = "" ;
      ray_start_index:_FillValue = -9999 ;
   float ray_start_range(time) ;
      ray_start_range:long_name = "start_range_for_ray" ;
      ray_start_range:units = "meters" ;
      ray_start_range:_FillValue = -9999.f ;
   float ray_gate_spacing(time) ;
      ray_gate_spacing:long_name = "gate_spacing_for_ray" ;
      ray_gate_spacing:units = "meters" ;
      ray_gate_spacing:_FillValue = -9999.f ;
   float azimuth(time) ;
      azimuth:long_name = "ray_azimuth_angle" ;
      azimuth:units = "degrees" ;
      azimuth:_FillValue = -9999.f ;
   float elevation(time) ;
      elevation:long_name = "ray_elevation_angle" ;
      elevation:units = "degrees" ;
      elevation:_FillValue = -9999.f ;
      elevation:positive = "up" ;
   float pulse_width(time) ;
      pulse_width:long_name = "transmitter_pulse_width" ;
      pulse_width:units = "seconds" ;
      pulse_width:_FillValue = -9999.f ;
      pulse_width:meta_group = "instrument_parameters" ;
   float prt(time) ;
      prt:long_name = "pulse_repetition_time" ;
      prt:units = "seconds" ;
      prt:_FillValue = -9999.f ;
      prt:meta_group = "instrument_parameters" ;
   float prt_ratio(time) ;
      prt_ratio:long_name = "pulse_repetition_frequency_ratio" ;
      prt_ratio:units = "seconds" ;
      prt_ratio:_FillValue = -9999.f ;
      prt_ratio:meta_group = "instrument_parameters" ;
   float nyquist_velocity(time) ;
      nyquist_velocity:long_name = "unambiguous_doppler_velocity" ;
      nyquist_velocity:units = "meters per second" ;
      nyquist_velocity:_FillValue = -9999.f ;
      nyquist_velocity:meta_group = "instrument_parameters" ;
   float unambiguous_range(time) ;
      unambiguous_range:long_name = "unambiguous_range" ;
      unambiguous_range:units = "meters" ;
      unambiguous_range:_FillValue = -9999.f ;
      unambiguous_range:meta_group = "instrument_parameters" ;
   byte antenna_transition(time) ;
      antenna_transition:long_name =
"antenna_is_in_transition_between_sweeps" ;
      antenna_transition:units = "" ;
```

```
      antenna_transition:_FillValue = -128b ;
      antenna_transition:comment = "1 if antenna is in transition, 0
otherwise" ;
   int n_samples(time) ;
      n_samples:long_name = "number_of_samples_used_to_compute_moments" ;
      n_samples:units = "" ;
      n_samples:_FillValue = -9999 ;
      n_samples:meta_group = "instrument_parameters" ;
   int r_calib_index(time) ;
      r_calib_index:long_name = "calibration_data_array_index_per_ray" ;
      r_calib_index:units = "" ;
      r_calib_index:_FillValue = -9999 ;
      r_calib_index:meta_group = "radar_calibration" ;
      r_calib_index:comment = "This is the index for the calibration which
applies to this ray" ;
   float measured_transmit_power_h(time) ;
      measured_transmit_power_h:long_name =
"measured_radar_transmit_power_h_channel" ;
      measured_transmit_power_h:units = "dBm" ;
      measured_transmit_power_h:_FillValue = -9999.f ;
      measured_transmit_power_h:meta_group = "radar_parameters" ;
   float measured_transmit_power_v(time) ;
      measured_transmit_power_v:long_name =
"measured_radar_transmit_power_v_channel" ;
      measured_transmit_power_v:units = "dBm" ;
      measured_transmit_power_v:_FillValue = -9999.f ;
      measured_transmit_power_v:meta_group = "radar_parameters" ;
   float scan_rate(time) ;
      scan_rate:long_name = "antenna_angle_scan_rate" ;
      scan_rate:units = "degrees per second" ;
      scan_rate:_FillValue = -9999.f ;
      scan_rate:meta_group = "instrument_parameters" ;
   short DBZ(n_points) ;
      DBZ:long_name = "radar_reflectivity" ;
      DBZ:standard_name = "equivalent_reflectivity_factor" ;
      DBZ:units = "dBZ" ;
      DBZ:sampling_ratio = 1.f ;
      DBZ:_FillValue = -32768s ;
      DBZ:scale_factor = 0.001411481f ;
      DBZ:add_offset = 17.25f ;
      DBZ:grid_mapping = "grid_mapping" ;
      DBZ:coordinates = "time range" ;
   short VEL(n_points) ;
      VEL:long_name = "radial_velocity" ;
      VEL:standard_name =
"radial_velocity_of_scatterers_away_from_instrument" ;
      VEL:units = "m/s" ;
      VEL:sampling_ratio = 1.f ;
      VEL:_FillValue = -32768s ;
      VEL:scale_factor = 0.0009842219f ;
      VEL:add_offset = -0.25f ;
      VEL:grid_mapping = "grid_mapping" ;
      VEL:coordinates = "time range" ;
   short WIDTH(n_points) ;
```

```
      WIDTH:long_name = "spectrum_width" ;
      WIDTH:standard_name = "doppler_spectrum_width" ;
      WIDTH:units = "m/s" ;
      WIDTH:sampling_ratio = 1.f ;
      WIDTH:_FillValue = -32768s ;
      WIDTH:scale_factor = 0.0002899258f ;
      WIDTH:add_offset = 9.5f ;
      WIDTH:grid_mapping = "grid_mapping" ;
      WIDTH:coordinates = "time range" ;
   short ZDR(n_points) ;
      ZDR:long_name = "differential_reflectivity" ;
      ZDR:standard_name = "log_differential_reflectivity_hv" ;
      ZDR:units = "dB" ;
      ZDR:sampling_ratio = 1.f ;
      ZDR:_FillValue = -32768s ;
      ZDR:scale_factor = 0.000241287f ;
      ZDR:add_offset = 0.03125f ;
      ZDR:grid_mapping = "grid_mapping" ;
      ZDR:coordinates = "time range" ;
   short PHIDP(n_points) ;
      PHIDP:long_name = "differential_phase" ;
      PHIDP:standard_name = "differential_phase_hv" ;
      PHIDP:units = "deg" ;
      PHIDP:sampling_ratio = 1.f ;
      PHIDP:_FillValue = -32768s ;
      PHIDP:scale_factor = 0.3525968f ;
      PHIDP:add_offset = 11553.19f ;
      PHIDP:grid_mapping = "grid_mapping" ;
      PHIDP:coordinates = "time range" ;
   short RHOHV(n_points) ;
      RHOHV:long_name = "cross_correlation" ;
      RHOHV:standard_name = "cross_correlation_ratio_hv" ;
      RHOHV:units = "" ;
      RHOHV:sampling_ratio = 1.f ;
      RHOHV:_FillValue = -32768s ;
      RHOHV:scale_factor = 1.286864e-05f ;
      RHOHV:add_offset = 0.63f ;
      RHOHV:grid_mapping = "grid_mapping" ;
      RHOHV:coordinates = "time range" ;

// global attributes:
      :Conventions = "CF-1.6" ;
      :Sub_conventions = "CF-Radial instrument_parameters radar_parameters
radar_calibration" ;
      :version = "CF-Radial-1.3" ;
      :title = "" ;
      :institution = "" ;
      :references = "" ;
      :source = "ARCHIVE 2 data" ;
      :history = "" ;
      :comment = "" ;
      :author = "" ;
      :original_format = "NEXRAD" ;
      :driver = "RadxConvert(NCAR)" ;
```

```
        :created = "2016/05/22 22:50:15.274" ;
        :start_datetime = "2015-06-26T12:04:15Z" ;
        :time_coverage_start = "2015-06-26T12:04:15Z" ;
        :end_datetime = "2015-06-26T12:08:31Z" ;
        :time_coverage_end = "2015-06-26T12:08:31Z" ;
        :instrument_name = "KDDC" ;
        :site_name = "" ;
        :scan_name = "Surveillance" ;
        :scan_id = 212 ;
        :platform_is_mobile = "false" ;
        :n_gates_vary = "true" ;
        :ray_times_increase = "true" ;
}
```