NCAR/UNIDATA

CfRadial Data File Format

Proposed
CF-compliant netCDF Format
for Moments Data for RADAR and LIDAR
in Radial Coordinates

Version 1.3

Mike Dixon, Wen-Chau Lee

Bob Rilling , Chris Burghart, Joe Van Andel
EOL, NCAR*

2013-07-01

NCAR is supported by the US National Science Foundation.

1 Introduction

1.1 On-line location

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

http://www.ral.ucar.edu/projects/titan/docs/radial_formats

1.2 Summary of updates

Version 1.1 was the first operational release for CfRadial.

All changes made subsequent to this version will be backward-compatible.

A major change was made for version 1.1. Support for a variable number of gates per ray was added.

Version 1.2 made some minor changes/additions.

The changes/additions made to create this version, 1.3, are high-lighted in yellow.

See section 8 for details.

1.3 Purpose

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

The intention is that the format should, as far as possible, comply with the CF conventions for gridded data. However, the current CF 1.5 convention does not support radial radar/lidar data. Therefore, extensions to the conventions will be required.

The current CF conventions are documented at:

http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.5

1.4 Extensions to the CF convention

This convention introduces the following extensions to CF:

1. The following axis attribute types:
   - axis = "radial_azimuth_coordinate";
   - axis = "radial_elevation_coordinate";
   - axis = "radial_range_coordinate";
2. Additional standard units. The following need to be added:
   - 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.

CfRadial files will be CF compliant, with the above extensions.

1.5 Strict use of variable and attribute names

However, 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 **metadata variable names** exactly as specified in this document, in addition to their standard names. It is not practical to expect an application to search for standard names for metadata variables, since this makes the code unnecessarily complex.

Since this is a completely new format, there is no requirement to support legacy data sets which are less strictly defined.

Note that this strictness requirement only applies to **metadata** variables. The **moments data** fields will be handled as usual in CF, where the **standard name** is the **definitive guide** to the contents of the field. Suggested standard names for radar variables not yet supported by CF are listed in section 6.

One exception to this is the dimensions used to specify the length of string variables. String length dimensions may be added as needed. This document refers to the string length dimension as 'string_length', but any suitable dimension may be used in its place. See section 4.2.

1.6 _FillValue and missing_value attributes for data fields

Earlier version of the CF specification stated that the use of the **missing_value** attribute was deprecated and that only **_FillValue** should be used.

However, the CF 1.6 specification reverses this, and **missing_value** is no longer deprecated, with the following note: "Not allowed for coordinate data except in the case of auxiliary coordinate variables in discrete sampling geometries."

The use of **_FillValue** is recommended for CfRadial files, and its use is regarded as a 'best practice. However, the use of **missing_value** instead of **_FillValue** is permitted.Only one or the other should be specified.

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

1.7 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.

2 Data Content Overview

2.1 The nature of radar and lidar moments data

As a radar or lidar scans (or points) the data fields (or **moments**) are produced over an entity 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**.

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**.

In most cases, the spacing between the range gates is constant along the ray, although this is not necessarily the case. For example, some NOAA radars have gate spacings of 75m, 150m and 300m. Therefore, we need to be able to handle the cases for which the range gate spacing is **variable**. (This was not supported in version 1.0).

2.2 Geo-reference variables

A subset of the 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 4.4, 4.6 and 4.8 for details on these variables.

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

These are:

- heading
- roll
- pitch
- rotation
- tilt

See section 4.9 for details on these variables.

The mathematical procedures for computing data location relative to earth coordinates are described in detail in section 7.

2.3 Coordinate variables and storage of moments data

The moments data to be handled by this format is represented in 2 principal dimensions:

- **time**: rays have monotonically increasing time
- **range**: bins have monotonically increasing range

2.3.1 Regular 2-D storage – constant number of gates

If the rays at all times have the same number of gates, the data is stored in regular arrays, as shown below.

In this case the **time** dimension may be either **fixed** or **unlimited**.

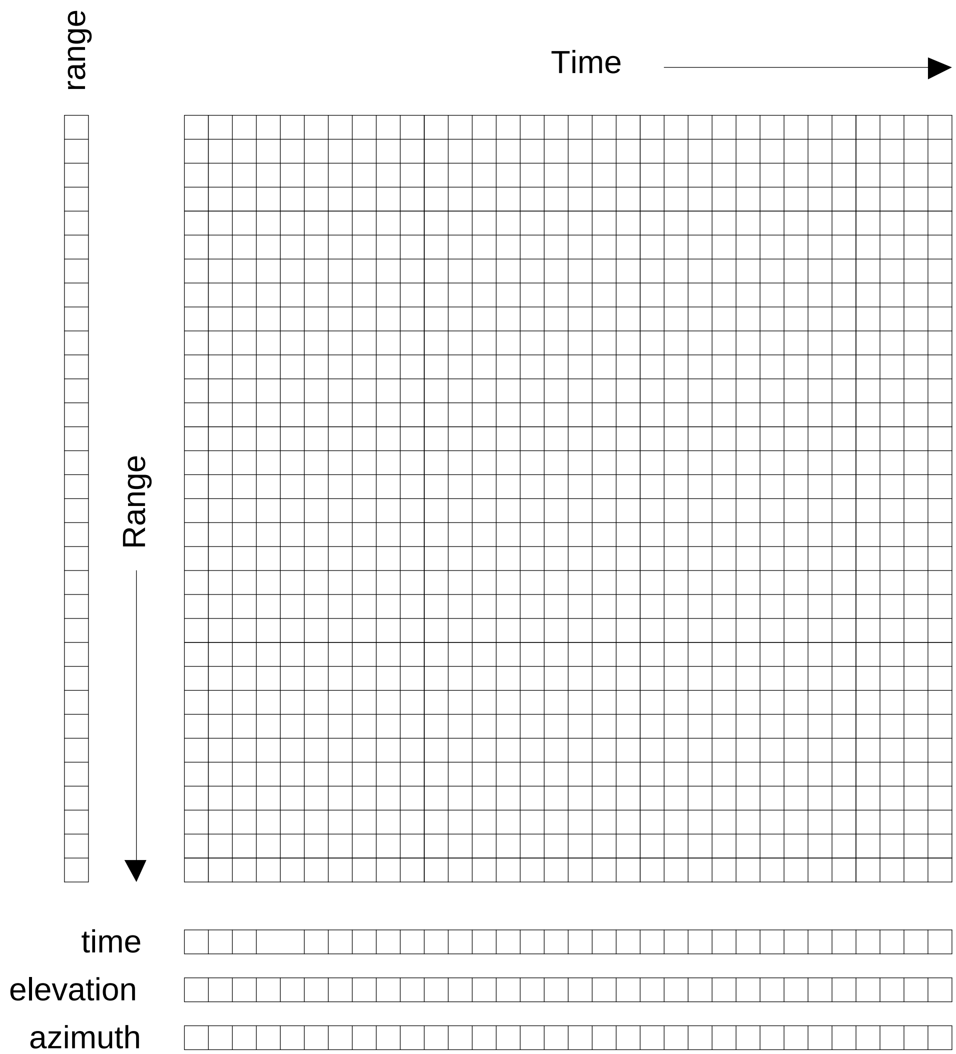

Figure 2.1 Data organization in time and range,
for a constant number of gates

2.3.2 Staggered 2-D storage – variable number of gates

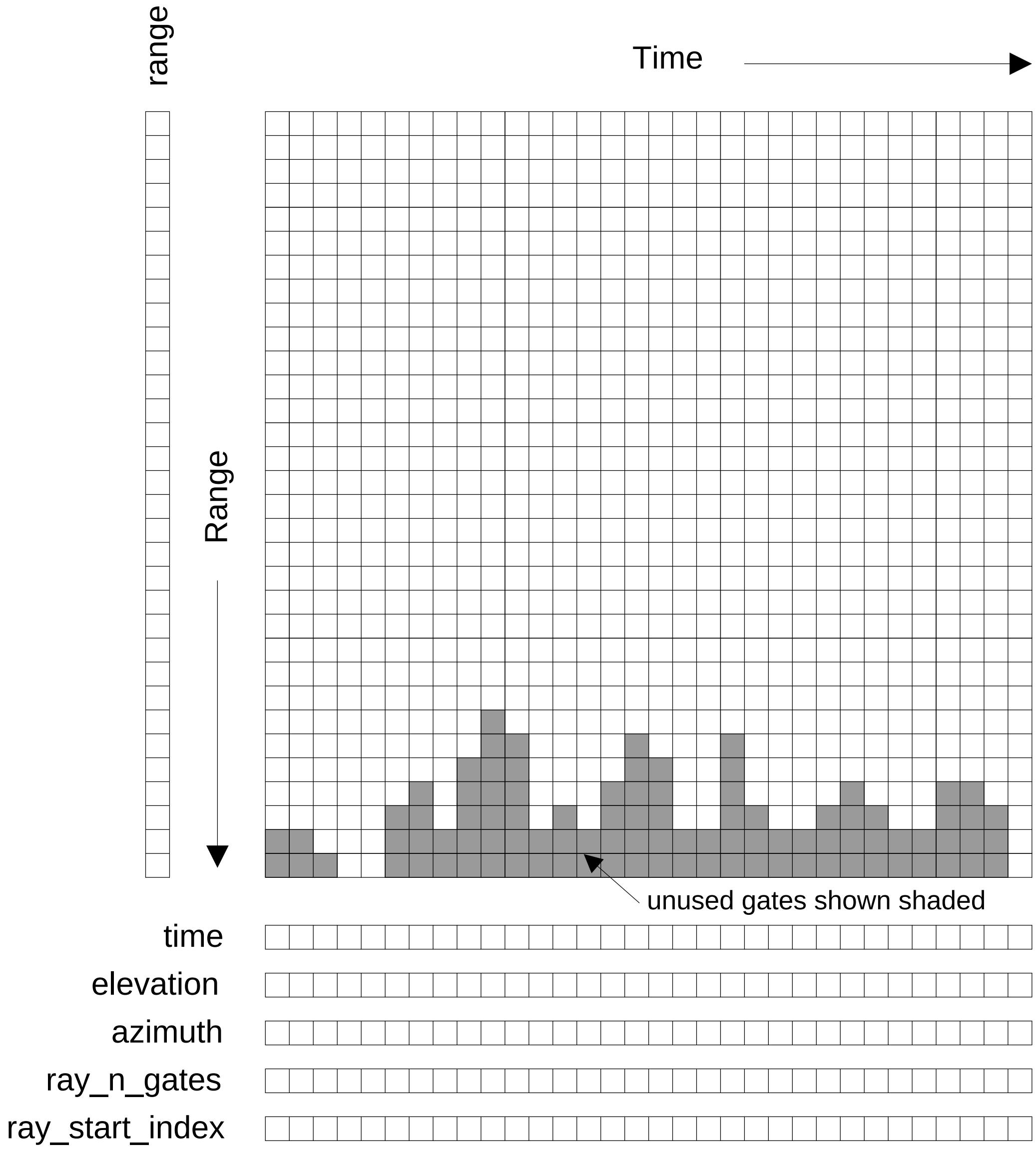

Figure 2.2 Data organization in time and range,
for a variable number of gates

2.3.3 Principal dimensions and variables for the case of a constant number of gates

Refer to figure 2.1.

The principal dimensions are **time** and **range**.

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

The **time** coordinate indicates the number of rays in the file. In the case of a constant number of gates, this may be either **fixed** or **unlimited**.

The **range** coordinate indicates the maximum number of gates for any ray in the file.

The **time(time)** coordinate variable stores the 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 **(time_reference)**.

The **range(range)** coordinate variable stores the range to the center of each gate. All rays in the volume must have the same range geometry, but not necessarily the same number of gates..

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)**.

2.3.1 Additional dimensions and variables for the case of a variable number of gates

Refer to figure 2.2.

For a variable number of gates per ray, and additional dimension, **n_points**, is introduced. The **time** coordinate in this case must be **fixed**.

The **ray_n_gates(time)** variable stores the number of gates in a ray.

The **ray_start_index(time)** variable stores the start index of the moments data for a ray, relative to the start of the moments array.

The **n_points** dimension indicates the total number of gates stored in all of the rays. It is equal to the **sum** of **ray_n_gates** over all rays.

The data fields are stored as 1-D arrays with dimension **(n_points).** The data from consecutive rays is concatenated to form a single array. The **ray_n_gates** and **ray_start_index** values are used to locate the data for a given time in this 1-D array.

2.4 Sweep indexes – a "pseudo" third dimension

A set of one or more related sweeps, typically a complete 3-D radar or lidar scan, is referred to as a **volume**.

A **volume scan** is comprised of one or more **sweeps**.

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

- horizontal scanning at fixed elevation (PPI mode)
- 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)

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

The **volume** may therefore be logically divided into **sweeps**. In CfRadial, we do not separate the sweeps in the stored field data arrays. Rather, we store arrays of **start** and **stop indexes**, which identify the rays that belong to each sweep. Some recorded rays may be in the transition region between defined sweeps, i.e. they may not belong to any sweep. For these rays we set the 'antenna_transition' flag to 1.

2.5 Constant start range and gate spacing per volume

The CF/radial convention supports a varying number of gates per ray.

However, the range to each gate cannot vary within a volume. The **range(range)** coordinate variable stores the range to the center of each gate, and these values are applicable to all rays.

If the raw data range geometry **changes over time** within a volume, the data to be represented must be re-sampled using a common **time-invariant** range geometry for the volume.

2.6 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

2.7 Calibration information

Radars must be calibrated to ensure that the moments data are accurate. Calibration for some types of lidar may also be appropriate.

A radar may have multiple sets of calibration parameters. Generally a separate calibration is performed for each transmit **pulse width.** Separate calibrations may be performed for other reasons as well. CfRadial supports storing multiple sets of calibration parameters, using the **radar_calibration** and **lidar_calibration** conventions.

The calibration applicable to a specific ray is indicated by the **calibration_index** variable.

2.8 Compression

The netCDF 4 library supports files in the following formats:

- classic
- 64bit offset
- netcdf4
- netcdf4 classic

The **netcdf4** format 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, combined with the use of the **netcdf4** format, provides efficient compression.

It is therefore recommended that the netcdf4 option be used whenever possible, to keep data sets as small as possible.

However, for importing data into 3rd party applications such as MatLab ©, it is wise (at this stage) to store data in the NetCDF-3 classic format, which is uncompressed, since support for the compressed format is not yet widespread.

3 Convention hierarchy

The CF/Radial convention comprises a **base** convention, along with a series of optional **sub-conventions** for specific purposes.

At the time of writing, the following conventions are supported:

Convention name	Type	Description
CF/Radial	Base	Radial data extension to the CF convention. Contains all necessary information for interpreting and displaying the data fields in a geo-referenced manner
instrument_parameters	Optional	Parameters common to both radar and lidar instruments
radar_parameters	Optional	Parameters specific to radars
lidar_parameters	Optional	Parameters specific to lidars
radar_calibration	Optional	Calibration values for radars
lidar_calibration	Optional	Calibration values for lidars
platform_velocity	Optional	Velocity of the platform, in multiple dimensions
geometry_correction	Optional	Corrections to the geometry of the data set

Note: items shown shaded are required, those not shaded are optional.

If a netCDF file conforms to a base convention and one or more sub-conventions, these are concatenated in the Conventions attribute as a space-delimited string.

The following are some examples:

- "CF/Radial instrument_parameters"
- "CF/Radial instrument_parameters radar_parameters radar_calibration"
- "CF/Radial lidar_parameters platform_velocity"

4 *CF/Radial* base convention

The base *CF/Radial* convention covers the minimum set of elements which are required to describe a radar/lidar data set sufficiently for basic display and plotting. *CF/Radial* is a specialization of *CF*.

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. The following abbreviations apply:

Units string in netCDF file	Abbreviation in tables
"degrees per second"	degrees/s
"meters per second"	m/s

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.

Attribute name	Type	Convention	Description
platform_is_mobile	string	CF/Radial	"true" or "false" Assumed "false" if missing.
n_gates_vary	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 false if the rays are not stored in time order.
field_names	string	CF/Radial	Comma-delimited list of field names included in this file.

Note: items shown shaded are required, those not shaded are optional.

4.2 Dimensions

Dimension name	Description
time	The number of rays. This dimension is optionally UNLIMITED
range	The number of range bins
n_points *	Total number of gates in file. Required for variable number of gates.
sweep	The number of sweeps
frequency	Number of frequencies used
string_length **	Length of char type variables.

Note 1: items shown shaded are required, those not shaded are optional.

*** Note2**: n_points is required if the number of gates varies by ray. It must not be included if the number of gates is fixed.

**** Note3**: any number of 'string_length' dimensions may be created and used. For example, you may declare the dimensions 'string_length", 'string_length_short' and 'string_length_long', and use them appropriately for strings of various lengths. These are only used to indicate the length of the strings actually stored, and have no effect on other parts of the format.

4.3 Global variables

Variable name	Dimension	Type	Comments
volume_number	none	int	Volume numbers are sequential, relative to some arbitrary start time, and may wrap.

Variable name	Dimension	Type	Comments
platform_type	(string_length)	char	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_length)	char	Options are: "radar", "lidar" Assumed "*radar*" if missing.
primary_axis	(string_length)	char	Options are: *"axis_z", "axis_y", "axis_x"* See section 7 for details. Assumed "*axis_z*" if missing.
time_coverage_start	(string_length)	char	UTC time of first ray in file. Resolution is integer seconds. The **time(time)** variable is computed relative to this time. Format is: yyyy-mm-ddThh:mm:ssZ
time_coverage_end	(string_length)	char	UTC time of last ray in file. Resolution is integer seconds. Format is: yyyy-mm-ddThh:mm:ssZ
time_reference	(string_length)	char	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: items shown shaded are required, those not shaded are optional.

4.4 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.

Variable name	Dimension	Type	Units	Comments
range	(range)	float	meters	Coordinate variable for range. Range to center of each bin.

Note: all items are required.

4.4.1 Attributes for time coordinate variable

Attribute name	Type	Value
standard_name	string	"time"
long_name	string	"time_in_seconds_since_volume_start"
units	string	"seconds since *yyyy-mm-dd*T*hh:mm:ssZ*", 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.
calendar	string	Defaults to "gregorian" or "standard" 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.

Note: All items are required.

4.4.2 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"

Note: items shown shaded are required, those not shaded are optional.

4.5 Ray dimension variables

Variable name	Dimension	Type	Comments
ray_n_gates	(time)	int	Number of gates in a ray.
ray_start_index	(time)	int	Index of start of moments data for a ray, relative to the start of the moments array

Note: required if n_gates_vary global attribute is true. Do not specify if n_gates_vary is false.

4.6 Location variables

Note: for *stationary* platforms, these are *scalars*, and for *moving* platforms they are *vectors* in the time dimension.

Variable name	Dimension	Type	Units	Comments
latitude	none or (time)	double	degrees_north	Latitude of instrument. For a stationary platform, this is a scalar. For a moving platform, this is a vector.
longitude	none or (time)	double	degrees_east	Longitude of instrument. For a stationary platform, this is a scalar. For a moving platform, this is a vector.
altitude	none or (time)	double	meters	Altitude of instrument, above mean sea level. For a stationary platform, this is a scalar. For a moving platform, this is a vector.
altitude_agl	none or (time)	double	meters	Altitude of instrument above ground level. For a stationary platform, this is a scalar. For a moving platform, this is a vector. Omit if not known.

Note: items shown shaded are required, those not shaded are optional.

4.7 Sweep variables

Variable name	Dimension	Type	Units	Comments
sweep_number	(sweep)	int		The number of the sweep, in the volume scan. 0-based.
sweep_mode	(sweep, string_length)	char		Options are: *"sector", "coplane",rhi", "vertical_pointing", "idle", "azimuth_surveillance", "elevation_surveillance", "sunscan", "pointing", "manual_ppi", "manual_rhi"*
fixed_angle	(sweep)	float	degrees	Target angle for the sweep. elevation in most modes azimuth in RHI mode
sweep_start_ray_index	(sweep)	int		Index of first ray in sweep, relative to start of volume. 0-based.
sweep_end_ray_index	(sweep)	int		Index of last ray in sweep, relative to start of volume. 0-based.
target_scan_rate	(sweep)	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	(sweep)	string		"true" or "false" Indicates whether or not the ray angles (elevation in RHI mode, azimuth in other modes) are indexed to a regular grid.
ray_angle_res	(sweep)	float	degrees	If rays_are_indexed is "true", this is the resolution of the angular grid – i.e. the delta angle between successive rays.

Note: items shown shaded are required, those not shaded are optional.

NOTE2: this section must always exist, even if a volume contains only 1 sweep. The reason for the inclusion is that the sweep_mode and sweep_fixed_angle are necessary for fully understanding the sweep strategy.

4.8 Sensor pointing variables

Variable name	Dimension	Type	Units	Comments
azimuth	(time)	float	degrees	Azimuth of antenna, relative to true north.
elevation	(time)	float	degrees	Elevation of antenna, relative to the horizontal plane.
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.
georefs_applied	(time)	byte		1 if georeference information for moving platforms has been applied to correct the azimuth and elevation. 0 otherwise. See section 4.9. Assumed 0 if missing.

Note: items shown shaded are required, those not shaded are optional.

4.8.1 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"

Note: All items are required.

4.8.2 Attributes for elevation(time) variable

Attribute name	Type	Value
standard_name	string	"ray_elevation_angle"
long_name	string	"elevation_angle_from_horizontal_plane"

Attribute name	Type	Value
units	string	“degrees”
axis	string	“radial_elevation_coordinate”

Note: All items are required.

4.9 Moving platform geo-reference variables

For *moving* platforms, the following additional variables will be included to allow geo-referencing of the platform in earth coordinates. Only include this section for moving platforms, omit completely for fixed platforms.

See section 7 for further details.

Variable name	Dimension	Type	Units	Comments
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 moving 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.

Note: if this block is included, all items are required.

4.10 Moments field data variables

If the number of gates per ray is fixed, the moments field variables will be 2-dimensional arrays, with the dimensions **time** and **range**.

If the number of gates per ray varies, the moments field variables will be 1-dimensional with the dimension **n_points.** The variables are stored as staggered arrays, using the auxiliary variables **ray_start_index(time)** and **ray_n_gates(time)** to locate the data for a ray.

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
long_name	string	CF	Long name describing the field. Any string is appropriate.
standard_name	string	CF	CF standard name for field. See section 6.2.
units	string	CF	Units for field
_FillValue	same type as field data	CF	Used if data are missing at this range bin
scale_factor	float	CF	Float value = (integer value) * scale_factor + add_offset Only applies to integer types.
add_offset	float	CF	
coordinates	string	CF	See note below
sampling_ratio	float	CF/Radial	Number of samples for this field divided by n_samples (see section 5.1). Assumed 1.0 if missing.

Attribute name	**Type**	**Convention**	**Description**
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.
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)

NOTES:

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

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 moving platforms, the coordinates attribute should be set to:

"*elevation azimuth range heading roll pitch rotation tilt*"

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>
```

5 Sub-conventions

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.

The following sub-conventions augment the base convention with additional information for various purposes.

5.1 The *instrument_parameters* sub-convention

This convention stores parameters relevant to both radars and lidars.

Variables in this convention will have the string attribute **meta_group**, set to the value "**instrument_parameters**".

Variable name	Dimension	Type	Units	Comments
frequency	(frequency)	float	s-1	List of operating frequencies, in Hertz. In most cases, only a single frequency is used.
follow_mode	(sweep, string_length)	char		options are: *"none", "sun", "vehicle", "aircraft", "target", "manual"* Assumed "*none*" if missing.
pulse_width	(time)	float	seconds	
prt_mode	(sweep, string_length)	char		Pulsing mode Options are: *"fixed", "staggered", "dual"* Assumed "*fixed*" if missing.
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.
polarization_mode	(sweep, string_length)	char		Options are: *"horizontal", "vertical", "hv_alt", "hv_sim", "circular"* Assumed "*horizontal*" if missing.
nyquist_velocity	(time)	float	m/s	Unambiguous velocity
unambiguous_range	(time)	float	meters	Unambiguous range
n_samples	(time)	int		Number of samples used to compute moments

Note: all items are optional.

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 4.10) is computed as the actual number of samples used for a given field, divided by n_samples. It would generally be 1.0.

5.2 The *radar_parameters* sub-convention

This convention handles parameters specific to radar platforms. Variables in this convention will have the string attribute **meta_group**, set to the value "**radar_parameters**".

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
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

Note: all items are optional.

5.3 The *lidar_parameters* sub-convention

This convention handles parameters specific to lidar platforms. Variables in this convention will have the string attribute **meta_group**, set to the value "**lidar_parameters**".

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	

Note: all items are optional.

5.4 The *radar_calibration* sub-convention

Variables in this convention will have the string attribute **meta_group**, set to the convention name "**radar_calibration"**.

5.4.1 Dimensions

Dimension name	Description
r_calib	The number of calibrations available

Note: required if any radar_calibration variables are included..

5.4.2 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.
r_calib_time	(r_calib, string_length)	char	UTC	e.g. 2008-09-25 T23:00:00Z
r_calib_pulse_width	(r_calib)	float	seconds	Pulse width for this calibration
r_calib_antenna_gain_h	(r_calib)	float	dB	Derived antenna gain H channel
r_calib_antenna_gain_v	(r_calib)	float	dB	ditto, V channel
r_calib_xmit_power_h	(r_calib)	float	dBm	Transmit power H channel
r_calib_xmit_power_v	(r_calib)	float	dBm	ditto, V channel

Variable name	Dimension	Type	Units	Comments
r_calib_two_way_waveguide_loss_h	(r_calib)	float	dB	2-way waveguide loss measurement plane to feed horn H channel
r_calib_two_way_waveguide_loss_v	(r_calib)	float	dB	ditto, V channel
r_calib_two_way_radome_loss_h	(r_calib)	float	dB	2-way radome loss H channel
r_calib_two_way_radome_loss_v	(r_calib)	float	dB	ditto, V channel
r_calib_receiver_mismatch_loss	(r_calib)	float	dB	Receiver filter bandwidth mismatch loss
r_calib_radar_constant_h	(r_calib)	float	m/mW dB units	Radar constant H channel
r_calib_radar_constant_v	(r_calib)	float	m/mW dB units	ditto, V channel
r_calib_noise_hc	(r_calib)	float	dBm	Measured noise level H co-pol channel
r_calib_noise_vc	(r_calib)	float	dBm	ditto, V co-pol channel
r_calib_noise_hx	(r_calib)	float	dBm	ditto, H cross-pol
r_calib_noise_vx	(r_calib)	float	dBm	ditto, V cross-pol
r_calib_receiver_gain_hc	(r_calib)	float	dB	Measured receiver gain H co-pol channel
r_calib_receiver_gain_vc	(r_calib)	float	dB	ditto, V co-pol channel
r_calib_receiver_gain_hx	(r_calib)	float	dB	ditto, H cross-pol
r_calib_receiver_gain_vx	(r_calib)	float	dB	ditto, V cross-pol
r_calib_base_1km_hc	(r_calib)	float	dBZ	reflectivity at 1km for SNR=0dB H co-pol channel
r_calib_base_1km_vc	(r_calib)	float	dBZ	ditto, V co-pol channel
r_calib_base_1km_hx	(r_calib)	float	dBZ	ditto, H cross-pol
r_calib_base_1km_vx	(r_calib)	float	dBZ	ditto, V cross-pol
r_calib_sun_power_hc	(r_calib)	float	dBm	Calibrate sun power H co-pol channel
r_calib_sun_power_vc	(r_calib)	float	dBm	ditto, V co-pol channel
r_calib_sun_power_hx	(r_calib)	float	dBm	ditto, H cross-pol
r_calib_sun_power_vx	(r_calib)	float	dBm	ditto, V cross-pol
r_calib_noise_source_power_h	(r_calib)	float	dBm	Noise source power H channel

Variable name	Dimension	Type	Units	Comments
r_calib_noise_source_power_v	(r_calib)	float	dBm	ditto, V channel
r_calib_power_measure_loss_h	(r_calib)	float	dB	Power measurement loss in coax and connectors H channel
r_calib_power_measure_loss_v	(r_calib)	float	dB	ditto, V channel
r_calib_coupler_forward_loss_h	(r_calib)	float	dB	Coupler loss into waveguide H channel
r_calib_coupler_forward_loss_v	(r_calib)	float	dB	ditto, V channel
r_calib_zdr_correction	(r_calib)	float	dB	corrected = measured + correction
r_calib_ldr_correction_h	(r_calib)	float	dB	corrected = measured + correction
r_calib_ldr_correction_v	(r_calib)	float	dB	corrected = measured + correction
r_calib_system_phidp	(r_calib)	float	degrees	System PhiDp, as seen in drizzle close to radar
r_calib_test_power_h	(r_calib)	float	dBm	Calibration test power H channel
r_calib_test_power_v	(r_calib)	float	dBm	ditto, V channel
r_calib_receiver_slope_hc	(r_calib)	float		Computed receiver slope, ideally 1.0 H co-pol channel
r_calib_receiver_slope_vc	(r_calib)	float		ditto, V co-pol channel
r_calib_receiver_slope_hx	(r_calib)	float		ditto, H cross-pol
r_calib_receiver_slope_vx	(r_calib)	float		ditto, V cross-pol

Note: all items are optional.

5.5 The *lidar_calibration* sub-convention

Variables in this convention will have the string attribute **meta_group**, set to the value **"lidar_calibration"**.

At the time of writing, this convention has not been defined.

5.6 The *platform_velocity* sub-convention

For *moving* platforms, include the following variables to indicate the velocity of the platform at each time. Omit entirely for fixed platforms.

Variables in this convention will have the string attribute **meta_group**, set to the value **"platform_velocity"**.

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.

Note: if this block is included, all items are required.

5.7 The *geometry_correction* sub-convention

The following additional variables are used to quantify errors in the georeference data for the platform. These are constant for a data set.

Variables in this convention will have the string attribute **meta_group**, set to the value **"geometry_correction"**.

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
elevation_correction	none	float	degrees	Correction to elevation values
range_correction	none	float	meters	Correction to range values

Variable name	Dimension	Type	Units	Comments
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

Note: none of these items is required. If missing, 0 will be assumed.

6 Standard names

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

6.1 Standard names for moments variables

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

This is an incomplete list. Please suggest additions as needed.

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_correlation_ratio_hv	RHOHV		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
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

Standard name	**Short name**	**Units**	**Already in CF?**
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_copolar_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

6.2 Proposed long names for metadata variables

Use of long names for metadata variables is optional, since the variable names themselves are reasonably self-explanatory. However, use of the long names does enhance clarity and makes the file more self-documenting.

Variable name ***Long name***	**Units**
altitude_agl *altitude_above_ground_level*	meters
altitude_correction *altitude_correction*	meters
altitude *altitude*	meters
antenna_transition *antenna_is_in_transition_between_sweeps*	unitless
azimuth_correction *azimuth_angle_correction*	degrees
azimuth *ray_azimuth_angle*	degrees
drift_correction *platform_drift_angle_correction*	degrees

Variable name ***Long name***	**Units**
drift *platform_drift_angle*	degrees
eastward_velocity_correction *platform_eastward_velocity_correction*	m/s
eastward_velocity *platform_eastward_velocity*	m/s
eastward_wind *eastward_wind*	m/s
elevation_correction *ray_elevation_angle_correction*	degrees
elevation *ray_elevation_angle*	degrees
time_coverage_end *data_volume_end_time_utc*	seconds
fixed_angle *target_fixed_angle*	degrees
follow_mode *follow_mode_for_scan_strategy*	unitless
frequency *radiation_frequency*	s-1
heading_change_rate *platform_heading_angle_rate_of_change*	degrees
heading_correction *platform_heading_angle_correction*	degrees
heading *platform_heading_angle*	degrees
instrument_name *name_of_instrument*	unitless
instrument_type *type_of_instrument*	unitless
latitude_correction *latitude_correction*	degrees
latitude *latitude*	degrees_east
lidar_aperture_diameter *lidar_aperture_diameter*	meters
lidar_aperture_efficiency *lidar_aperture_efficiency*	unitless
lidar_beam_divergence *lidar_beam_divergence*	radians
lidar_constant *lidar_calibration_constant*	unitless

Variable name ***Long name***	**Units**
lidar_field_of_view *lidar_field_of_view*	radians
lidar_peak_power *lidar_peak_power*	watts
lidar_pulse_energy *lidar_pulse_energy*	joules
longitude_correction *longitude_correction*	degrees
longitude *longitude*	degrees_east
northward_velocity_correction *platform_northward_velocity_correction*	m/s
northward_velocity *platform_northward_velocity*	m/s
northward_wind *northward_wind*	m/s
nyquist_velocity *unambiguous_doppler_velocity*	m/s
n_samples *number_of_samples_used_to_compute_moments*	unitless
pitch_change_rate *platform_pitch_angle_rate_of_change*	degrees
pitch_correction *platform_pitch_angle_correction*	degrees
pitch *platform_pitch_angle*	degrees
platform_is_mobile *platform_is_mobile*	unitless
platform_type *platform_type*	unitless
polarization_mode transmit_receive_*polarization_mode*	unitless
prt_mode *transmit_pulse_mode*	unitless
pressure_altitude_correction *pressure_altitude_correction*	meters
primary_axis *primary_axis_of_rotation*	unitless
prt *pulse_repetition_time*	seconds
prt_ratio *multiple_pulse_repetition_frequency_ratio*	

Variable name ***Long name***	**Units**
pulse_width *transmitter_pulse_width*	seconds
radar_antenna_gain_h *nominal_radar_antenna_gain_h_channel*	dB
radar_antenna_gain_v *nominal_radar_antenna_gain_v_channel*	dB
radar_beam_width_h *half_power_radar_beam_width_h_channel*	degrees
radar_beam_width_v *half_power_radar_beam_width_v_channel*	degrees
radar_receiver_bandwidth *radar_receiver_bandwidth*	s-1
radar_measured_transmit_power_h *radar_measured_transmit_power_h_channel*	dBm
radar_measured_transmit_power_v *radar_measured_transmit_power_v_channel*	dBm
range_correction *range_to_center_of_measurement_volume_correction*	meters
range *projection_range_coordinate*	meters
roll_correction *platform_roll_angle_correction*	degrees
roll *platform_roll_angle*	degrees
rotation_correction *ray_rotation_angle_relative_to_platform_correction*	degrees
rotation *ray_rotation_angle_relative_to_platform*	degrees
r_calib_antenna_gain_h *calibrated_radar_antenna_gain_h_channel*	dB
r_calib_antenna_gain_v *calibrated_radar_antenna_gain_v_channel*	dB
r_calib_base_dbz_1km_hc *radar_reflectivity_at_1km_at_zero_snr_h_co_polar_channel*	dBZ
r_calib_base_dbz_1km_hx *radar_reflectivity_at_1km_at_zero_snr_h_cross_polar_channel*	dBZ
r_calib_base_dbz_1km_vc *radar_reflectivity_at_1km_at_zero_snr_v_co_polar_channel*	dBZ
r_calib_base_dbz_1km_vx *radar_reflectivity_at_1km_at_zero_snr_v_cross_polar_channel*	dBZ
r_calib_coupler_forward_loss_h *radar_calibration_coupler_forward_loss_h_channel*	dB

Variable name ***Long name***	**Units**
r_calib_coupler_forward_loss_v *radar_calibration_coupler_forward_loss_v_channel*	dB
r_calib_index *calibration_data_array_index_per_ray*	unitless
r_calib_ldr_correction_h *calibrated_radar_ldr_correction_h_channel*	dB
r_calib_ldr_correction_v *calibrated_radar_ldr_correction_v_channel*	dB
r_calib_noise_hc *calibrated_radar_receiver_noise_h_co_polar_channel*	dBm
r_calib_noise_hx *calibrated_radar_receiver_noise_h_cross_polar_channel*	dBm
r_calib_noise_vc *calibrated_radar_receiver_noise_v_co_polar_channel*	dBm
r_calib_noise_vx *calibrated_radar_receiver_noise_v_cross_polar_channel*	dBm
r_calib_noise_source_power_h *radar_calibration_noise_source_power_h_channel*	dBm
r_calib_noise_source_power_v *radar_calibration_noise_source_power_v_channel*	dBm
r_calib_power_measure_loss_h *radar_calibration_power_measurement_loss_h_channel*	dB
r_calib_power_measure_loss_v *radar_calibration_power_measurement_loss_v_channel*	dB
r_calib_pulse_width *radar_calibration_pulse_width*	seconds
r_calib_radar_constant_h *calibrated_radar_constant_h_channel*	(m/mW)dB
r_calib_radar_constant_v *calibrated_radar_constant_v_channel*	(m/mW)dB
r_calib_receiver_gain_hc *calibrated_radar_receiver_gain_h_co_polar_channel*	dB
r_calib_receiver_gain_hx *calibrated_radar_receiver_gain_h_cross_polar_channel*	dB
r_calib_receiver_gain_vc *calibrated_radar_receiver_gain_v_co_polar_channel*	dB
r_calib_receiver_gain_vx *calibrated_radar_receiver_gain_v_cross_polar_channel*	dB
r_calib_receiver_mismatch_loss *radar_calibration_receiver_mismatch_loss*	dB
r_calib_receiver_slope_hc *calibrated_radar_receiver_slope_h_co_polar_channel*	unitless

Variable name ***Long name***	**Units**
r_calib_receiver_slope_hx *calibrated_radar_receiver_slope_h_cross_polar_channel*	unitless
r_calib_receiver_slope_vc *calibrated_radar_receiver_slope_v_co_polar_channel*	unitless
r_calib_receiver_slope_vx *calibrated_radar_receiver_slope_v_cross_polar_channel*	unitless
r_calib_sun_power_hc *calibrated_radar_sun_power_h_co_polar_channel*	dBm
r_calib_sun_power_hx *calibrated_radar_sun_power_h_cross_polar_channel*	dBm
r_calib_sun_power_vc *calibrated_radar_sun_power_v_co_polar_channel*	dBm
r_calib_sun_power_vx *calibrated_radar_sun_power_v_cross_polar_channel*	dBm
r_calib_system_phidp *calibrated_radar_system_phidp*	degrees
r_calib_test_power_h *radar_calibration_test_power_h_channel*	dBm
r_calib_test_power_v *radar_calibration_test_power_v_channel*	dBm
r_calib_time *radar_calibration_time_utc*	unitless
r_calib_two_way_radome_loss_h *radar_calibration_two_way_radome_loss_h_channel*	dB
r_calib_two_way_radome_loss_v *radar_calibration_two_way_radome_loss_v_channel*	dB
r_calib_two_way_waveguide_loss_h *radar_calibration_two_way_waveguide_loss_h_channel*	dB
r_calib_two_way_waveguide_loss_v *radar_calibration_two_way_waveguide_loss_v_channel*	dB
r_calib_xmit_power_h *calibrated_radar_xmit_power_h_channel*	dBm
r_calib_xmit_power_v *calibrated_radar_xmit_power_v_channel*	dBm
r_calib_zdr_correction *calibrated_radar_zdr_correction*	dB
scan_name *name_of_antenna_scan_strategy*	unitless
scan_rate *antenna_angle_scan_rate*	unitless
site_name *name_of_instrument_site*	unitless

Variable name ***Long name***	**Units**
spacing_is_constant *spacing_between_range_gates_is_constant*	unitless
sweep_end_ray_index *index_of_last_ray_in_sweep*	unitless
sweep_mode *scan_mode_for_sweep*	unitless
sweep_number *sweep_index_number_0_based*	unitless
sweep_start_ray_index *index_of_first_ray_in_sweep*	unitless
sweep_unambiguous_range *unambiguous_range_for_sweep*	meters
tilt_correction *ray_tilt_angle_relative_to_platform_correction*	degrees
tilt *ray_tilt_angle_relative_to_platform*	degrees
time *time*	seconds
time_coverage_start *data_volume_start_time_utc*	unitless
unambiguous_range *unambiguous_range*	meters
vertical_velocity_correction *platform_vertical_velocity_correction*	m/s
vertical_velocity *platform_vertical_velocity*	m/s
vertical_wind *upward_air_velocity*	m/s
volume_number *data_volume_index_number*	unitless

7 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 $\mathbf{X}$ (see Fig. 7.1) where Y is TN and X is East. Hence, a coordinate transformation between $\mathbf{X}_i$ (radar sampling space) and $\mathbf{X}$ (geo-reference space) is required. Based on the principal axes, most remote sensors can be classified into three right-hand types, X, Y, or Z type.

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

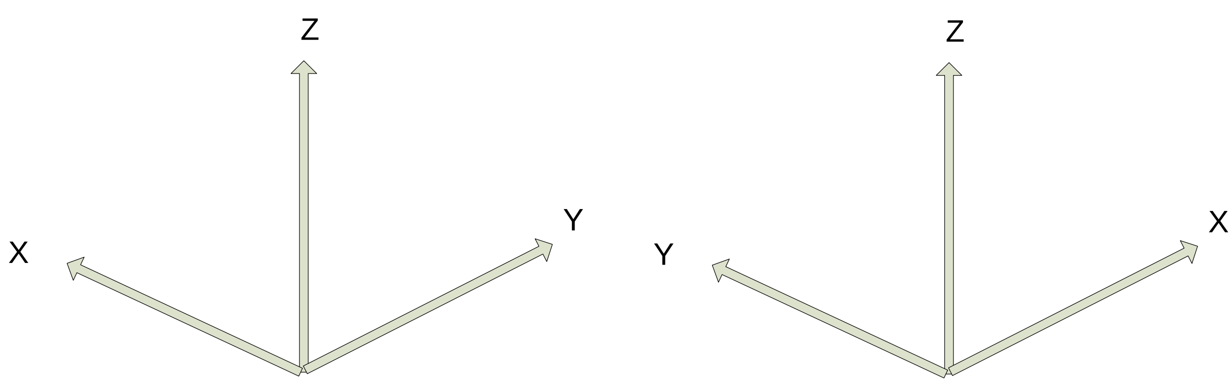

Figure 7.1: Left-handed XYZ coordinate system vs. Right-handed XYZ coordinate system.

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.

7.1 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°.

7.1.1 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.

7.1.2 RADARs

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

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

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

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

where $R' = \left(\frac{4}{3}\right) \bullet 6374 \ km$ 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.

7.2 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 section 7)
- 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.

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

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

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

7.3 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.

7.3.1 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 $\mathbf{X}_h$.

7.3.2 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.

7.3.3 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.

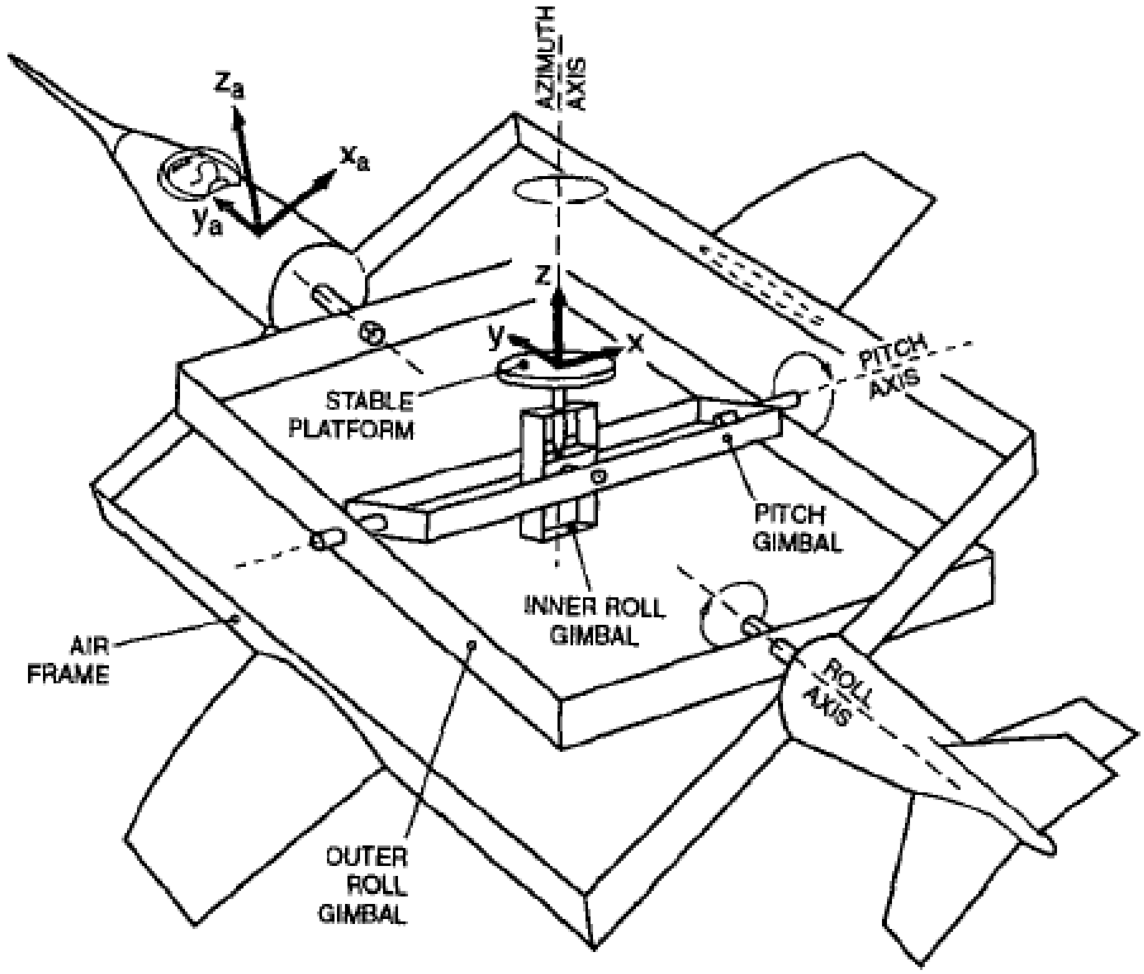

Figure 7.2 Moving platform axis definitions and reference frame (reproduced from Lee et al., 1994,originally from Axford, 1968) ©American Meteorological Society. Reprinted with permission.

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

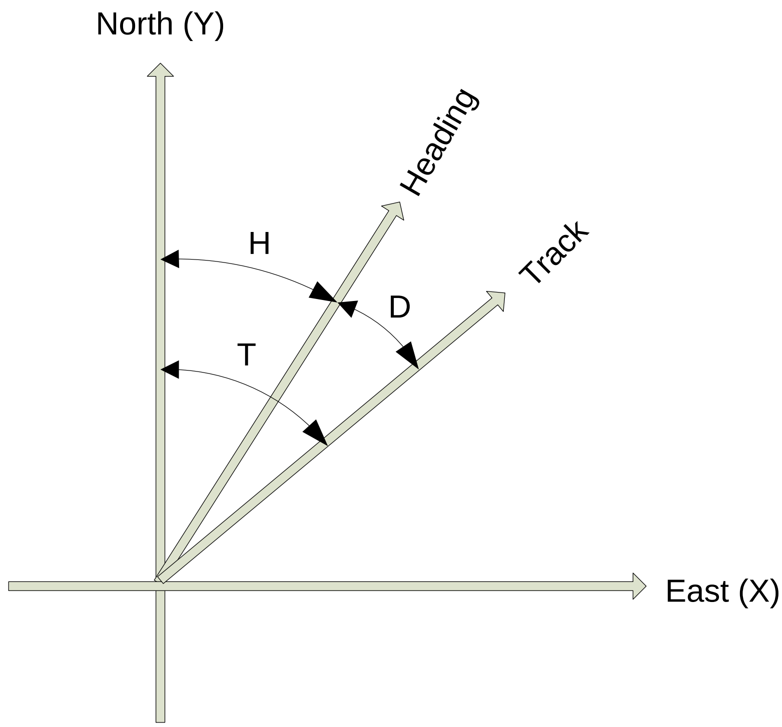

Figure 7.3(a): Definition of heading, drift and track.

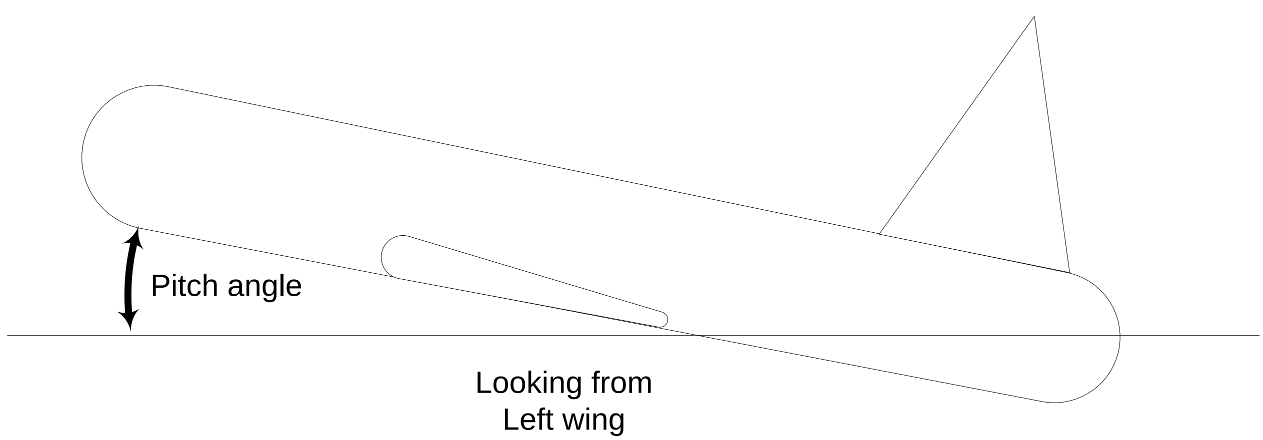

Figure 7.3(b): Definition of pitch

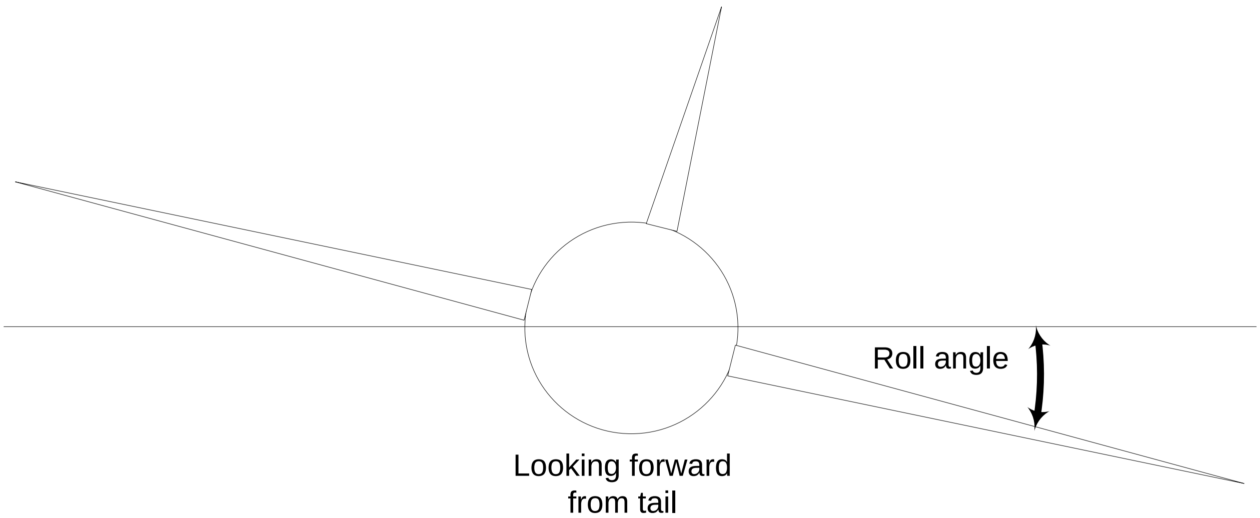

Figure 7.3(c): Definition of roll

7.3.4 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 7.1: 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

7.4 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 $\mathbf{X}$

7.4.1 Transformation from X_i to X_a

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

7.4.1.1 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:

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

7.4.1.2 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:

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

7.4.1.3 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:

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

7.4.1.4 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:

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

7.4.2 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:

$$X = M_H M_P M_R X_a$$

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

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

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

7.5 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:

7.5.1 For type Z radars:

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

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

7.5.2 For type Y radars:

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

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

7.5.3 For type Y-prime radars:

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

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

7.5.4 For type X radars:

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

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

7.5.5 Computing earth-relative azimuth and elevation

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

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

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

7.6 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 = $(4/3)6374km$

8 Change log

8.1 Version 1.3, released 2013-06-01

- Added **Y-prime** radar type, for ELDORA and NOAA tail-type radars.
- Added **field_names** global attribute, listing the fields in the file.
- Added **rays_are_indexed** and **ray_angle_res** sweep variables.
- Added **is_discrete**, **field_folds**, **field_limit_lower** and **field_limit_upper** field attributes.
- Version 1.3 changes are high-lighted in yellow.
- Moved section 6.2 to section 6.1. Added standard names to list.
- Moved section 6.1 to section 6.2 – and changed this section to specify suggested long names, rather than standard names.

8.2 Version 1.2, released 2011-06-07

- Formalized the concept of required vs. optional dimensions and variables. In this document, required variables are shown shaded, and footnotes were added to each table.
- Added scan_id global attribute.
- Added time_reference variable. If this exists, the time(time) variable is computed relative to this time rather than relative to time_coverage_start.
- Added radar_receiver_bandwidth variable.
- Fixed various errors.

8.3 Version 1.1, released 2011-02-01

- Version 1.1 is the first operational release for CfRadial.
- All changes made subsequent to this version must be backward-compatible.
- A major change was made for version 1.1 – changing the storage of moments variables from **regular** (time, range) arrays to **staggered** arrays. This change supports a variable number of gates per ray, which makes the storage of operational data more efficient. For example, the NEXRAD data format supports changing the number of gates for different sweeps.

9 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

10 Example ncdump of CfRadial file

The following is an example ncdump from a valid CfRadial file:

```
netcdf
cfrad.20010108_040012.730_to_20010108_040109.538_SPOL_v246_s1150_el14.60_S
UR {
dimensions:
    time = 372 ;
    range = 1176 ;
    sweep = 1 ;
    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:_fillValue = -9999 ;
    char platform_type(string_length_32) ;
        platform_type:long_name = "platform_type" ;
        platform_type:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
        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" ;
        status_xml:_fillValue = -9999 ;
    char instrument_type(string_length_32) ;
        instrument_type:long_name = "type_of_instrument" ;
        instrument_type:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
      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" ;
      time_coverage_end:_fillValue = -9999 ;
   float frequency(frequency) ;
      frequency:long_name = "radiation_frequency" ;
      frequency:units = "s-1" ;
      frequency:_fillValue = -9999.f ;
      frequency:meta_group = "instrument_parameters" ;
   float azimuth_correction ;
      azimuth_correction:long_name = "azimuth_angle_correction" ;
      azimuth_correction:units = "degrees" ;
      azimuth_correction:_fillValue = -9999.f ;
      azimuth_correction:meta_group = "geometry_correction" ;
   float elevation_correction ;
      elevation_correction:long_name = "ray_elevation_angle_correction" ;
      elevation_correction:units = "degrees" ;
      elevation_correction:_fillValue = -9999.f ;
      elevation_correction:meta_group = "geometry_correction" ;
   float range_correction ;
      range_correction:long_name =
"range_to_center_of_measurement_volume_correction" ;
      range_correction:units = "meters" ;
      range_correction:_fillValue = -9999.f ;
      range_correction:meta_group = "geometry_correction" ;
   float longitude_correction ;
      longitude_correction:long_name = "longitude_correction" ;
      longitude_correction:units = "degrees" ;
      longitude_correction:_fillValue = -9999.f ;
      longitude_correction:meta_group = "geometry_correction" ;
   float latitude_correction ;
      latitude_correction:long_name = "latitude_correction" ;
      latitude_correction:units = "degrees" ;
      latitude_correction:_fillValue = -9999.f ;
```

```
      latitude_correction:meta_group = "geometry_correction" ;
   float pressure_altitude_correction ;
      pressure_altitude_correction:long_name =
"pressure_altitude_correction" ;
      pressure_altitude_correction:units = "meters" ;
      pressure_altitude_correction:_fillValue = -9999.f ;
      pressure_altitude_correction:meta_group = "geometry_correction" ;
   float altitude_correction ;
      altitude_correction:long_name = "altitude_correction" ;
      altitude_correction:units = "meters" ;
      altitude_correction:_fillValue = -9999.f ;
      altitude_correction:meta_group = "geometry_correction" ;
   float eastward_velocity_correction ;
      eastward_velocity_correction:long_name =
"platform_eastward_velocity_correction" ;
      eastward_velocity_correction:units = "meters per second" ;
      eastward_velocity_correction:_fillValue = -9999.f ;
      eastward_velocity_correction:meta_group = "geometry_correction" ;
   float northward_velocity_correction ;
      northward_velocity_correction:long_name =
"platform_northward_velocity_correction" ;
      northward_velocity_correction:units = "meters per second" ;
      northward_velocity_correction:_fillValue = -9999.f ;
      northward_velocity_correction:meta_group = "geometry_correction" ;
   float vertical_velocity_correction ;
      vertical_velocity_correction:long_name =
"platform_vertical_velocity_correction" ;
      vertical_velocity_correction:units = "meters per second" ;
      vertical_velocity_correction:_fillValue = -9999.f ;
      vertical_velocity_correction:meta_group = "geometry_correction" ;
   float heading_correction ;
      heading_correction:long_name = "platform_heading_angle_correction" ;
      heading_correction:units = "degrees" ;
      heading_correction:_fillValue = -9999.f ;
      heading_correction:meta_group = "geometry_correction" ;
   float roll_correction ;
      roll_correction:long_name = "platform_roll_angle_correction" ;
      roll_correction:units = "degrees" ;
      roll_correction:_fillValue = -9999.f ;
      roll_correction:meta_group = "geometry_correction" ;
   float pitch_correction ;
      pitch_correction:long_name = "platform_pitch_angle_correction" ;
      pitch_correction:units = "degrees" ;
      pitch_correction:_fillValue = -9999.f ;
      pitch_correction:meta_group = "geometry_correction" ;
   float drift_correction ;
      drift_correction:long_name = "platform_drift_angle_correction" ;
      drift_correction:units = "degrees" ;
      drift_correction:_fillValue = -9999.f ;
      drift_correction:meta_group = "geometry_correction" ;
   float rotation_correction ;
      rotation_correction:long_name =
"ray_rotation_angle_relative_to_platform_correction" ;
      rotation_correction:units = "degrees" ;
```

```
        rotation_correction:_fillValue = -9999.f ;
        rotation_correction:meta_group = "geometry_correction" ;
    float tilt_correction ;
        tilt_correction:long_name =
"ray_tilt_angle_relative_to_platform_correction" ;
        tilt_correction:units = "degrees" ;
        tilt_correction:_fillValue = -9999.f ;
        tilt_correction:meta_group = "geometry_correction" ;
    int sweep_number(sweep) ;
        sweep_number:long_name = "sweep_index_number_0_based" ;
        sweep_number:_fillValue = -9999 ;
    char sweep_mode(sweep, string_length_32) ;
        sweep_mode:long_name = "scan_mode_for_sweep" ;
        sweep_mode:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
        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:_fillValue = -9999 ;
    int sweep_end_ray_index(sweep) ;
        sweep_end_ray_index:long_name = "index_of_last_ray_in_sweep" ;
        sweep_end_ray_index:_fillValue = -9999 ;
    char rays_are_indexed(sweep, string_length_8) ;
        rays_are_indexed:long_name = "flag_for_indexed_rays" ;
        rays_are_indexed:_fillValue = -9999 ;
    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:_fillValue = -9999 ;
      r_calib_time:meta_group = "radar_calibration" ;
   float r_calib_pulse_width(r_calib) ;
      r_calib_pulse_width:standard_name = "radar_calibration_pulse_width"
;
      r_calib_pulse_width:units = "seconds" ;
      r_calib_pulse_width:meta_group = "radar_calibration" ;
   float r_calib_xmit_power_h(r_calib) ;
      r_calib_xmit_power_h:standard_name =
"calibrated_radar_xmit_power_h_channel" ;
      r_calib_xmit_power_h:units = "dBm" ;
      r_calib_xmit_power_h:meta_group = "radar_calibration" ;
   float r_calib_xmit_power_v(r_calib) ;
      r_calib_xmit_power_v:standard_name =
"calibrated_radar_xmit_power_v_channel" ;
      r_calib_xmit_power_v:units = "dBm" ;
      r_calib_xmit_power_v:meta_group = "radar_calibration" ;
   float r_calib_two_way_waveguide_loss_h(r_calib) ;
      r_calib_two_way_waveguide_loss_h:standard_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" ;
   float r_calib_two_way_waveguide_loss_v(r_calib) ;
      r_calib_two_way_waveguide_loss_v:standard_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" ;
   float r_calib_two_way_radome_loss_h(r_calib) ;
      r_calib_two_way_radome_loss_h:standard_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" ;
   float r_calib_two_way_radome_loss_v(r_calib) ;
      r_calib_two_way_radome_loss_v:standard_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" ;
   float r_calib_receiver_mismatch_loss(r_calib) ;
      r_calib_receiver_mismatch_loss:standard_name =
"radar_calibration_receiver_mismatch_loss" ;
      r_calib_receiver_mismatch_loss:units = "dB" ;
      r_calib_receiver_mismatch_loss:meta_group = "radar_calibration" ;
   float r_calib_radar_constant_h(r_calib) ;
      r_calib_radar_constant_h:standard_name =
"calibrated_radar_constant_h_channel" ;
      r_calib_radar_constant_h:units = "dB" ;
      r_calib_radar_constant_h:meta_group = "radar_calibration" ;
   float r_calib_radar_constant_v(r_calib) ;
      r_calib_radar_constant_v:standard_name =
"calibrated_radar_constant_v_channel" ;
      r_calib_radar_constant_v:units = "dB" ;
```

```
      r_calib_radar_constant_v:meta_group = "radar_calibration" ;
   float r_calib_antenna_gain_h(r_calib) ;
      r_calib_antenna_gain_h:standard_name =
"calibrated_radar_antenna_gain_h_channel" ;
      r_calib_antenna_gain_h:units = "dB" ;
      r_calib_antenna_gain_h:meta_group = "radar_calibration" ;
   float r_calib_antenna_gain_v(r_calib) ;
      r_calib_antenna_gain_v:standard_name =
"calibrated_radar_antenna_gain_v_channel" ;
      r_calib_antenna_gain_v:units = "dB" ;
      r_calib_antenna_gain_v:meta_group = "radar_calibration" ;
   float r_calib_noise_hc(r_calib) ;
      r_calib_noise_hc:standard_name =
"calibrated_radar_receiver_noise_h_co_polar_channel" ;
      r_calib_noise_hc:units = "dBm" ;
      r_calib_noise_hc:meta_group = "radar_calibration" ;
   float r_calib_noise_vc(r_calib) ;
      r_calib_noise_vc:standard_name =
"calibrated_radar_receiver_noise_v_co_polar_channel" ;
      r_calib_noise_vc:units = "dBm" ;
      r_calib_noise_vc:meta_group = "radar_calibration" ;
   float r_calib_noise_hx(r_calib) ;
      r_calib_noise_hx:standard_name =
"calibrated_radar_receiver_noise_h_cross_polar_channel" ;
      r_calib_noise_hx:units = "dBm" ;
      r_calib_noise_hx:meta_group = "radar_calibration" ;
   float r_calib_noise_vx(r_calib) ;
      r_calib_noise_vx:standard_name =
"calibrated_radar_receiver_noise_v_cross_polar_channel" ;
      r_calib_noise_vx:units = "dBm" ;
      r_calib_noise_vx:meta_group = "radar_calibration" ;
   float r_calib_receiver_gain_hc(r_calib) ;
      r_calib_receiver_gain_hc:standard_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" ;
   float r_calib_receiver_gain_vc(r_calib) ;
      r_calib_receiver_gain_vc:standard_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" ;
   float r_calib_receiver_gain_hx(r_calib) ;
      r_calib_receiver_gain_hx:standard_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" ;
   float r_calib_receiver_gain_vx(r_calib) ;
      r_calib_receiver_gain_vx:standard_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" ;
   float r_calib_base_dbz_1km_hc(r_calib) ;
      r_calib_base_dbz_1km_hc:standard_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" ;
   float r_calib_base_dbz_1km_vc(r_calib) ;
      r_calib_base_dbz_1km_vc:standard_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" ;
   float r_calib_base_dbz_1km_hx(r_calib) ;
      r_calib_base_dbz_1km_hx:standard_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" ;
   float r_calib_base_dbz_1km_vx(r_calib) ;
      r_calib_base_dbz_1km_vx:standard_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" ;
   float r_calib_sun_power_hc(r_calib) ;
      r_calib_sun_power_hc:standard_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" ;
   float r_calib_sun_power_vc(r_calib) ;
      r_calib_sun_power_vc:standard_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" ;
   float r_calib_sun_power_hx(r_calib) ;
      r_calib_sun_power_hx:standard_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" ;
   float r_calib_sun_power_vx(r_calib) ;
      r_calib_sun_power_vx:standard_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" ;
   float r_calib_noise_source_power_h(r_calib) ;
      r_calib_noise_source_power_h:standard_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" ;
   float r_calib_noise_source_power_v(r_calib) ;
      r_calib_noise_source_power_v:standard_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" ;
   float r_calib_power_measure_loss_h(r_calib) ;
      r_calib_power_measure_loss_h:standard_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" ;
   float r_calib_power_measure_loss_v(r_calib) ;
```

```
      r_calib_power_measure_loss_v:standard_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" ;
   float r_calib_coupler_forward_loss_h(r_calib) ;
      r_calib_coupler_forward_loss_h:standard_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" ;
   float r_calib_coupler_forward_loss_v(r_calib) ;
      r_calib_coupler_forward_loss_v:standard_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" ;
   float r_calib_zdr_correction(r_calib) ;
      r_calib_zdr_correction:standard_name =
"calibrated_radar_zdr_correction" ;
      r_calib_zdr_correction:units = "dB" ;
      r_calib_zdr_correction:meta_group = "radar_calibration" ;
   float r_calib_ldr_correction_h(r_calib) ;
      r_calib_ldr_correction_h:standard_name =
"calibrated_radar_ldr_correction_h_channel" ;
      r_calib_ldr_correction_h:units = "dB" ;
      r_calib_ldr_correction_h:meta_group = "radar_calibration" ;
   float r_calib_ldr_correction_v(r_calib) ;
      r_calib_ldr_correction_v:standard_name =
"calibrated_radar_ldr_correction_v_channel" ;
      r_calib_ldr_correction_v:units = "dB" ;
      r_calib_ldr_correction_v:meta_group = "radar_calibration" ;
   float r_calib_system_phidp(r_calib) ;
      r_calib_system_phidp:standard_name = "calibrated_radar_system_phidp"
;
      r_calib_system_phidp:units = "degrees" ;
      r_calib_system_phidp:meta_group = "radar_calibration" ;
   float r_calib_test_power_h(r_calib) ;
      r_calib_test_power_h:standard_name =
"radar_calibration_test_power_h_channel" ;
      r_calib_test_power_h:units = "dBm" ;
      r_calib_test_power_h:meta_group = "radar_calibration" ;
   float r_calib_test_power_v(r_calib) ;
      r_calib_test_power_v:standard_name =
"radar_calibration_test_power_v_channel" ;
      r_calib_test_power_v:units = "dBm" ;
      r_calib_test_power_v:meta_group = "radar_calibration" ;
   float r_calib_receiver_slope_hc(r_calib) ;
      r_calib_receiver_slope_hc:standard_name =
"calibrated_radar_receiver_slope_h_co_polar_channel" ;
      r_calib_receiver_slope_hc:meta_group = "radar_calibration" ;
   float r_calib_receiver_slope_vc(r_calib) ;
      r_calib_receiver_slope_vc:standard_name =
"calibrated_radar_receiver_slope_v_co_polar_channel" ;
      r_calib_receiver_slope_vc:meta_group = "radar_calibration" ;
   float r_calib_receiver_slope_hx(r_calib) ;
```

```
        r_calib_receiver_slope_hx:standard_name =
"calibrated_radar_receiver_slope_h_cross_polar_channel" ;
        r_calib_receiver_slope_hx:meta_group = "radar_calibration" ;
    float r_calib_receiver_slope_vx(r_calib) ;
        r_calib_receiver_slope_vx:standard_name =
"calibrated_radar_receiver_slope_v_cross_polar_channel" ;
        r_calib_receiver_slope_vx:meta_group = "radar_calibration" ;
    double time(time) ;
        time:standard_name = "time" ;
        time:long_name = "time in seconds since volume start" ;
        time:calendar = "gregorian" ;
        time:units = "seconds since 2001-01-08T04:00:12Z" ;
        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 = 150.000005960464 ;
        range:meters_between_gates = 149.890631437302 ;
    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_frequency" ;
        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:_fillValue = 0b ;
      antenna_transition:comment = "1 if antenna is in transition, 0
otherwise" ;
   byte georefs_applied(time) ;
      georefs_applied:long_name = "georefs_have_been_applied_to_ray" ;
      georefs_applied:_fillValue = 0b ;
      georefs_applied:comment = "1 if georefs have been applied, 0
otherwise" ;
   int n_samples(time) ;
      n_samples:long_name = "number_of_samples_used_to_compute_moments" ;
      n_samples:_fillValue = -9999 ;
      n_samples:meta_group = "instrument_parameters" ;
   byte r_calib_index(time) ;
      r_calib_index:long_name = "calibration_data_array_index_per_ray" ;
      r_calib_index:_fillValue = 0b ;
      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" ;
   double latitude(time) ;
      latitude:long_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:_fillValue = -9999. ;
   double longitude(time) ;
      longitude:long_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:_fillValue = -9999. ;
   double altitude(time) ;
      altitude:long_name = "altitude" ;
      altitude:units = "meters" ;
      altitude:_fillValue = -9999. ;
      altitude:positive = "up" ;
   float altitude_agl(time) ;
      altitude_agl:long_name = "altitude_above_ground_level" ;
      altitude_agl:units = "meters" ;
      altitude_agl:_fillValue = -9999.f ;
```

```
float heading(time) ;
   heading:long_name = "platform_heading_angle" ;
   heading:units = "degrees" ;
   heading:_fillValue = -9999.f ;
float roll(time) ;
   roll:long_name = "platform_roll_angle" ;
   roll:units = "degrees" ;
   roll:_fillValue = -9999.f ;
float pitch(time) ;
   pitch:long_name = "platform_pitch_angle" ;
   pitch:units = "degrees" ;
   pitch:_fillValue = -9999.f ;
float drift(time) ;
   drift:long_name = "platform_drift_angle" ;
   drift:units = "degrees" ;
   drift:_fillValue = -9999.f ;
float rotation(time) ;
   rotation:long_name = "ray_rotation_angle_relative_to_platform" ;
   rotation:units = "degrees" ;
   rotation:_fillValue = -9999.f ;
float tilt(time) ;
   tilt:long_name = "ray_tilt_angle_relative_to_platform" ;
   tilt:units = "degrees" ;
   tilt:_fillValue = -9999.f ;
float eastward_velocity(time) ;
   eastward_velocity:long_name = "platform_eastward_velocity" ;
   eastward_velocity:units = "meters per second" ;
   eastward_velocity:_fillValue = -9999.f ;
   eastward_velocity:meta_group = "platform_velocity" ;
float northward_velocity(time) ;
   northward_velocity:long_name = "platform_northward_velocity" ;
   northward_velocity:units = "meters per second" ;
   northward_velocity:_fillValue = -9999.f ;
   northward_velocity:meta_group = "platform_velocity" ;
float vertical_velocity(time) ;
   vertical_velocity:long_name = "platform_vertical_velocity" ;
   vertical_velocity:units = "meters per second" ;
   vertical_velocity:_fillValue = -9999.f ;
   vertical_velocity:meta_group = "platform_velocity" ;
float eastward_wind(time) ;
   eastward_wind:long_name = "eastward_wind" ;
   eastward_wind:units = "meters per second" ;
   eastward_wind:_fillValue = -9999.f ;
   eastward_wind:meta_group = "platform_velocity" ;
float northward_wind(time) ;
   northward_wind:long_name = "northward_wind" ;
   northward_wind:units = "meters per second" ;
   northward_wind:_fillValue = -9999.f ;
   northward_wind:meta_group = "platform_velocity" ;
float vertical_wind(time) ;
   vertical_wind:long_name = "upward_air_velocity" ;
   vertical_wind:units = "meters per second" ;
   vertical_wind:_fillValue = -9999.f ;
   vertical_wind:meta_group = "platform_velocity" ;
```

```
   float heading_change_rate(time) ;
      heading_change_rate:long_name =
"platform_heading_angle_rate_of_change" ;
      heading_change_rate:units = "degrees per second" ;
      heading_change_rate:_fillValue = -9999.f ;
      heading_change_rate:meta_group = "platform_velocity" ;
   float pitch_change_rate(time) ;
      pitch_change_rate:long_name = "platform_pitch_angle_rate_of_change"
;
      pitch_change_rate:units = "degrees per second" ;
      pitch_change_rate:_fillValue = -9999.f ;
      pitch_change_rate:meta_group = "platform_velocity" ;
   short VEL(time, range) ;
      VEL:long_name = "Doppler radial velocity" ;
      VEL:units = "m/s" ;
      VEL:sampling_ratio = 1.f ;
      VEL:_FillValue = -32768s ;
      VEL:scale_factor = 0.01f ;
      VEL:add_offset = -0.f ;
      VEL:grid_mapping = "grid_mapping" ;
      VEL:coordinates = "time range" ;
   short WIDTH(time, range) ;
      WIDTH:long_name = "Spectrum width of VE" ;
      WIDTH:units = "m/s" ;
      WIDTH:sampling_ratio = 1.f ;
      WIDTH:_FillValue = -32768s ;
      WIDTH:scale_factor = 0.01f ;
      WIDTH:add_offset = -0.f ;
      WIDTH:grid_mapping = "grid_mapping" ;
      WIDTH:coordinates = "time range" ;
   short DBZ(time, range) ;
      DBZ:long_name = "Horizontal copolar reflectivity (HH)" ;
      DBZ:units = "dBz" ;
      DBZ:sampling_ratio = 1.f ;
      DBZ:_FillValue = -32768s ;
      DBZ:scale_factor = 0.01f ;
      DBZ:add_offset = -0.f ;
      DBZ:grid_mapping = "grid_mapping" ;
      DBZ:coordinates = "time range" ;
   short PHIDP(time, range) ;
      PHIDP:long_name = "Diff. propagation phase btwn HH and VV" ;
      PHIDP:units = "deg." ;
      PHIDP:sampling_ratio = 1.f ;
      PHIDP:_FillValue = -32768s ;
      PHIDP:scale_factor = 0.01f ;
      PHIDP:add_offset = -0.f ;
      PHIDP:grid_mapping = "grid_mapping" ;
      PHIDP:coordinates = "time range" ;

// global attributes:
      :Conventions = "CF/Radial instrument_parameters radar_parameters
radar_calibration geometry_correction" ;
      :version = "1.2" ;
      :title = "IMPROVE01" ;
```

```
        :institution =  ;
        :references =  ;
        :source = "NCAR/ATD" ;
        :history =  ;
        :comment =  ;
        :instrument_name = "SPOL" ;
        :site_name =  ;
        :scan_name =  ;
        :scan_id = 0 ;
        :platform_is_mobile = "false" ;
        :n_gates_vary = "false" ;
        :ray_times_increase = "true" ;
}
```