NCAR/UNIDATA

CfRadial Data File Format

**CF-compliant netCDF Format**

**for Moments Data for RADAR and LIDAR**

**in Radial Coordinates**

Version 1.5

Mike Dixon  
Wen-Chau Lee

EOL, NCAR\*

2021-12-01

\* NCAR is sponsored by the US National Science Foundation.

[1 Introduction 5](#_Toc88657613)

[1.1 On-line location 5](#_Toc88657614)

[1.2 Summary of updates 5](#_Toc88657615)

[1.2.1 Version 1.5 5](#_Toc88657616)

[1.2.2 Version 1.4, released 2016-08-01 5](#_Toc88657617)

[1.2.3 Version 1.3, released 2013-06-01 6](#_Toc88657618)

[1.2.4 Version 1.2, released 2011-06-07 6](#_Toc88657619)

[1.2.5 Version 1.1, released 2011-02-01 6](#_Toc88657620)

[1.3 Purpose 7](#_Toc88657621)

[1.4 Extensions to the CF convention 7](#_Toc88657622)

[1.5 Strict use of variable and attribute names for non-field variables 7](#_Toc88657623)

[1.6 \_FillValue and missing\_value attributes for data fields 8](#_Toc88657624)

[1.7 Required vs. optional variables 8](#_Toc88657625)

[1.8 String length variables 8](#_Toc88657626)

[2 Data Content Overview 9](#_Toc88657627)

[2.1 The nature of radar and lidar moments data 9](#_Toc88657628)

[2.2 Geo-reference variables 9](#_Toc88657629)

[2.3 Coordinate variables and storage of moments data 10](#_Toc88657630)

[2.3.1 Regular 2-D storage – constant number of gates 10](#_Toc88657631)

[2.3.2 Staggered 2-D storage – variable number of gates 11](#_Toc88657632)

[2.3.3 Principal dimensions and variables for the case of a constant number of gates 11](#_Toc88657633)

[2.3.1 Additional dimensions and variables for the case of a variable number of gates 12](#_Toc88657634)

[2.4 Sweep indexes – a "pseudo" third dimension 12](#_Toc88657635)

[2.5 Constant start range and gate spacing per sweep 13](#_Toc88657636)

[2.6 No grid mapping variable 13](#_Toc88657637)

[2.7 Calibration information 14](#_Toc88657638)

[2.8 Compression 14](#_Toc88657639)

[3 Convention hierarchy 15](#_Toc88657640)

[4 *CF/Radial* base convention 16](#_Toc88657641)

[4.1 Global attributes 16](#_Toc88657642)

[4.2 Dimensions 17](#_Toc88657643)

[4.3 Global variables 18](#_Toc88657644)

[4.4 Coordinate variables 19](#_Toc88657645)

[4.4.1 Attributes for time coordinate variable 19](#_Toc88657646)

[4.4.2 Attributes for range coordinate variable 20](#_Toc88657647)

[4.5 Ray dimension variables 20](#_Toc88657648)

[4.6 Location variables 21](#_Toc88657649)

[4.7 Sweep variables 22](#_Toc88657650)

[4.8 Sensor pointing variables 23](#_Toc88657651)

[4.8.1 Attributes for azimuth(time) variable 23](#_Toc88657652)

[4.8.2 Attributes for elevation(time) variable 24](#_Toc88657653)

[4.9 Moving platform geo-reference variables 24](#_Toc88657654)

[4.10 Moments field data variables 25](#_Toc88657655)

[4.10.1 Use of scale\_factor and add\_offset 27](#_Toc88657656)

[4.10.2 Use of coordinates attribute 28](#_Toc88657657)

[4.10.3 Use of flag values - optional 28](#_Toc88657658)

[4.10.4 Flag mask fields - optional 28](#_Toc88657659)

[4.10.5 Quality control fields - optional 28](#_Toc88657660)

[4.10.6 Legend XML 29](#_Toc88657661)

[4.11 Ray qualifier fields 29](#_Toc88657662)

[4.12 Spectra field variables 31](#_Toc88657663)

[4.12.1 Spectrum index variables 32](#_Toc88657664)

[4.12.2 Spectrum fields 32](#_Toc88657665)

[4.12.3 Spectrum field attributes 32](#_Toc88657666)

[5 Sub-conventions 34](#_Toc88657667)

[5.1 The *instrument\_parameters* sub-convention 34](#_Toc88657668)

[5.2 The *radar\_parameters* sub-convention 36](#_Toc88657669)

[5.3 The *lidar\_parameters* sub-convention 36](#_Toc88657670)

[5.4 The *radar\_calibration* sub-convention 36](#_Toc88657671)

[5.4.1 Dimensions 37](#_Toc88657672)

[5.4.2 Variables 37](#_Toc88657673)

[5.5 The *lidar\_calibration* sub-convention 40](#_Toc88657674)

[5.6 The *platform\_velocity* sub-convention 40](#_Toc88657675)

[5.7 The *geometry\_correction* sub-convention 41](#_Toc88657676)

[6 Standard names 43](#_Toc88657677)

[6.1 Standard names for moments variables 43](#_Toc88657678)

[6.2 Standard names for spectra variables 44](#_Toc88657679)

[6.3 Long names for metadata variables 45](#_Toc88657680)

[7 Computing the data location from geo-reference variables 52](#_Toc88657681)

[7.1 Special case – ground-based, stationary and leveled sensors 53](#_Toc88657682)

[7.1.1 LIDARs 53](#_Toc88657683)

[7.1.2 RADARs 53](#_Toc88657684)

[7.2 Moving platforms 54](#_Toc88657685)

[7.3 Coordinate transformations for the general case 55](#_Toc88657686)

[7.3.1 Coordinate systems 55](#_Toc88657687)

[7.3.2 The earth-relative coordinate system 55](#_Toc88657688)

[7.3.3 The platform-relative coordinate system 55](#_Toc88657689)

[7.3.4 The sensor coordinate system 58](#_Toc88657690)

[7.4 Coordinate transformation sequence 59](#_Toc88657691)

[7.4.1 Transformation from Xi to Xa 59](#_Toc88657692)

[7.4.2 Rotating from Xa to X 60](#_Toc88657693)

[7.5 Summary of transforming from Xi to X 62](#_Toc88657694)

[7.5.1 For type Z radars: 62](#_Toc88657695)

[7.5.2 For type Y radars: 62](#_Toc88657696)

[7.5.3 For type Y-prime radars: 62](#_Toc88657697)

[7.5.4 For type X radars: 63](#_Toc88657698)

[7.5.5 Computing earth-relative azimuth and elevation 63](#_Toc88657699)

[7.6 Summary of symbol definitions 63](#_Toc88657700)

[8 References 64](#_Toc88657701)

[9 Example ncdump of CfRadial file 65](#_Toc88657702)

# Introduction

## On-line location

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

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

## Summary of updates

|  |  |  |
| --- | --- | --- |
| Date | Version | Remarks |
| 2011/02/01 | 1.1 | First operational version. |
| 2011/06/07 | 1.2 | Minor changes / additions. |
| 2013/07/01 | 1.3 | Major changes / additions |
| 2016/08/01 | 1.4 | Major additions – data quality, spectra. |
| 2021/12/01 | 1.5 | Added ray qualifier fields |

All changes made subsequent to version 1.1 are backward-compatible.

The changes/additions between this version, 1.4, and the previous version, 1.3, are highlighted in yellow in this document.

### Version 1.5

* Added ray qualifier fields, Section 4.11.

### Version 1.4, released 2016-08-01

* Clarifies the preferential use of **\_FillValue**. Section 1.6.
* Relax requirement for gate geometry to be constant for entire volume. Changed to support gate geometry changes between sweeps. Sections 2.5, 4.4.
* Added support for multiple prts. Sections 4.2, 5.1.
* Added support for spectra. Sections 4.2, 4.11.
* ‘T’ optional in time strings. Section 4.3.
* Added ‘**qc\_procedures’** as optional sweep variable to document quality control procedures per sweep. Section 4.7.
* Added optional use of **flag\_values**, **flag\_masks** and **flag\_meanings** for fields. Section 4.10.
* Added optional use of **is\_quality**, **qualified\_variables** and **ancillary\_variables** for quality control fields. Section 4.10.
* Added optional **rx\_range\_resolution** – raw receiver range resolution. Section 5.1.

### 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.

### 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.

### 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.

See section 8 for details.

## 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 1.6 convention does not support radial radar/lidar data. Therefore, extensions to the conventions will be required.

The current CF conventions are documented at:

<http://cfconventions.org/>

<http://cfconventions.org/cf-conventions/v1.6.0/cf-conventions.html>

## 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";

1. Additional standard units. For CfRadial to follow CF properly, the following need to be added to udunits:

* dB (ratio of two values, in log units. For example, ZDR).
* dBm (power in milliwatts, in log units)
* dBZ (radar reflectivity in log units)

1. Additional standard names.

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

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

In CfRadial a ‘field’ variable stores such quantities as radar moments, derived quantities, qc measures etc. These are either measured by the radar, or derived from a field that is measured. These ‘field’ variables require a standard\_name as specified in the CF conventions. For example the field variable ‘radar reflectivity’ would have the standard name ‘*equivalent\_reflectivity\_factor’*. These variables store the fundamental scientific data associated with the instrument.

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

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

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

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

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

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

For CfRadial, either **\_FillValue** or **missing\_value** may be used to indicate missing values. **\_FillValue** is preferred.

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

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

## 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.

## String length variables

An extra note is required with respect to 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.

# Data Content Overview

## 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).

## 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.

## 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

### 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

### Staggered 2-D storage – variable number of gates



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

### 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)**.

### 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.

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

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.

## Constant start range and gate spacing per sweep

CF/Radial allows the number of gates stored per ray to vary.

However, the range geometry cannot vary within a sweep. This means that the range to the first gate, and the spacing between gates, must be constant for a sweep. For many datasets, the range geometry will be constant for the entire volume.

If the gate geometry is constant for the entire volume, then the **range(range)** coordinate variable is 1-Dimensional, and stores the range to the center of each gate, and these values are applicable to all rays.

If the gate geometry is **not** constant for the entire volume, then the **range(sweep, range)** coordinate variable is 2-Dimensional and stores the range to the center of each gate. These values apply to all rays within the specified sweep.

If the raw data range geometry **changes over time** (i.e. from ray to ray) within a sweep, the data to be represented in that sweep must be re-sampled using a common **time-invariant** range geometry for the sweep.

## 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

## 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.

## 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** (or **missing\_value**)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.

# 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”

# *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 |

## Global attributes

| Attribute name | Type | Convention | Description |
| --- | --- | --- | --- |
| Conventions | string | CF | Conventions string will specify Cf/Radial, plus selected sub-conventions as applicable |
| version | string | CF/Radial | CF/Radial version number |
| title | string | CF | Short description of file contents |
| institution | string | CF | Where the original data were produced |
| references | string | CF | References that describe the data or the methods used to produce it |
| source | string | CF | Method of production of the original data |
| history | string | CF | List of modifications to the original data |
| comment | string | CF | Miscellaneous information |
| instrument\_name | string | CF/Radial | Name of radar or lidar |
| site\_name | string | CF/Radial | Name of site where data were gathered |
| scan\_name | string | CF/Radial | Name of scan strategy used, if applicable |
| scan\_id | int | CF/Radial | Scan strategy id, if applicable. Assumed 0 if missing. |
| platform\_is\_mobile | string | CF/Radial | “true” or “false” Assumed “false” if missing. |
| 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.

## 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. |
| n\_prts | Number of prts used in pulsing scheme. Optional for fixed, staggered or dual Required for more complicated schemes. |
| n\_spectra | Number of spectra in data set. This dimension name is not fixed. Any suitable name can be used. There will be multiple dimensions if different spectral data sets have different shapes. |
| spectrum\_n\_samples | Number of samples per spectrum. This dimension name is not fixed. Any suitable name can be used. There will be multiple dimensions if different spectral data sets have different shapes. |

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.

## Global variables

| Variable name | Dimension | Type | Comments |
| --- | --- | --- | --- |
| volume\_number | none | int | Volume numbers are sequential, relative to some arbitrary start time, and may wrap. |
| 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”, “axis\_z\_prime”, “axis\_y\_prime”, “axis\_x\_prime”.*  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 follows ISO 8601:  yyyy-mm-ddThh:mm:ssZ  NOTE: the T is optional, any single character may be used in this location. |
| time\_coverage\_end | (string\_length) | char | UTC time of last ray in file. Resolution is integer seconds. Format is:  yyyy-mm-ddThh:mm:ssZ  NOTE: the T is optional, any single character may be used in this location. |
| time\_reference | (string\_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: the T is optional, any single character may be used in this location. |

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

## Coordinate variables

| Variable name | Dimension | Type | Units | Comments |
| --- | --- | --- | --- | --- |
| time | (time) | double | seconds | Coordinate variable for time. Time at center of each ray, in fractional seconds since time\_coverage\_start, or since time\_reference if it exists. |
| range | (range) or (sweep, range) | float | meters | Coordinate variable for range.  Range to center of each bin. If the range geometry is constant for the entire volume, use the 1-D array method. If the range geometry varies from sweep to sweep, use the 2-D array method. Client applications will determine which is in use by examining the dimensionality of the range variable. |

Note: all items are required.

### 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:ss*Z”, where the actual reference time values are used. This unit string is **very important** and must be correct. It should either match time\_reference( if it exists) or time\_coverage\_start.  NOTE: the T is optional, any single character may be used in this location. |
| calendar | string | Defaults to “gregorian” if missing. Options are:  “gregorian” or “standard”, “proleptic\_gregorian”, “noleap” or “365\_day”, “all\_leap” or “366\_day”, “360\_day”, “julian”  See CF conventions for details. |

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

### 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 or float(sweep) | Start range in meters.  If the range variable has dimensions (sweep, range), the float(sweep) form is used. |
| meters\_between\_gates | float or float(sweep) | Gate spacing in meters. Required if  spacing\_is\_constant is “true”. Not applicable otherwise. If the range variable has dimensions (sweep, range), the float(sweep) form is used. |
| axis | string | “radial\_range\_coordinate” |

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

## 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.

## 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 scanning radar, this is the center of rotation of the antenna. 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.

## 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. |
| qc\_procedures | (sweep, string\_length) | char |  | Documents QC procedures per sweep. |

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.

## Sensor pointing variables

| Variable name | Dimension | Type | Units | Comments |
| --- | --- | --- | --- | --- |
| azimuth | (time) | float | degrees | Azimuth of antenna, relative to true north. The azimuth should refer to the center of the dwell. |
| elevation | (time) | float | degrees | Elevation of antenna, relative to the horizontal plane.  The elevation should refer to the center of the dwell. |
| scan\_rate | (time) | float | degrees/s | Actual antenna scan rate. Set to negative if counter-clockwise in azimuth or decreasing in elevation. Positive otherwise. |
| antenna\_transition | (time) | byte |  | 1 if antenna is in transition, i.e. between sweeps, 0 if not. If variable is omitted, the transition will be assumed to be 0 everywhere.  Assumed 0 if missing. |
| 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.

### 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.

### Attributes for elevation(time) variable

| Attribute name | Type | Value |
| --- | --- | --- |
| standard\_name | string | “ray\_elevation\_angle” |
| long\_name | string | “elevation\_angle\_from\_horizontal\_plane” |
| units | string | “degrees” |
| axis | string | “radial\_elevation\_coordinate” |

**Note**: All items are required.

## 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 sections 5.6 and 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.

## 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. Although this is an optional attribute, its use is strongly encouraged. |
| standard\_name | string | CF | CF standard name for field.  See section 6.2. |
| units | string | CF | Units for field |
| \_FillValue (or missing\_value) | same type as field data | CF | Indicates data are missing at this range bin. Use of \_FillValue is preferred. Only use one or the other. |
| 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. |
| 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. |
| is\_quality\_field | string | CF/Radial | “true” or ”false” “true” indicates this is a quality control field. If the attribute is not present, defaults to “false”. |
| flag\_values | array of same type as field data | CF | Array of flag values. These values have special meaning, as documented in flag\_meanings. |
| flag\_meanings | string | CF | Meaning of flag\_values or flag\_masks. Space-separated list in string. |
| flag\_masks | array of same type as field data | CF | Valid bit-wise masks used in a flag field that is comprised of bit-wise combinations of mask values. See flag\_meanings. |
| qualified\_variables | string | CF/Radial | Applicable if is\_quality\_field is “true”. Space-separated list of variables that this variable qualifies. Every field variable in this list should list this variable in its ancillary\_variable attribute. |
| ancillary\_variables | string | CF | Space-separated list of variables to which this variable is related. In particular, this is intended to list the variables that contain quality information about this field. In that case, the quality field will list this field in its qualified\_variable attribute. |
| thresholding\_xml | string | CF/Radial | Thresholding details. Supplied if thresholding has been applied to the field. This should be in self-descriptive XML. (See below for example) |
| legend\_xml | string | CF/Radial | Legend details. Applies to discrete fields. Maps field values to the properties they represent. This should be in self-descriptive XML. (See below for example) |

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

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

### Use of coordinates attribute

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

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

**“*elevation azimuth range*”**

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

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

### Use of flag values - optional

For all data sets, the \_**FillValue** attribute has special meaning – see 1.6 above.

A field variable may make use of more than one reserved value, to indicate a variety of conditions. For example, with radar data, you may wish to indicate that the beam is blocked for a given gate, and that no echo will ever be detected at that gate. That provides more information than just using \_**FillValue**.

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

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

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

### Flag mask fields - optional

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

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

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

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

### Quality control fields - optional

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

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

In addition, the field should have the **qualified\_variables** string attribute. This is a space-separated list of field names that this field qualifies.

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

### Legend XML

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

<legend label="particle\_id">

<category>

<value>1</value>

<label>cloud</label>

<category>

<category>

<value>2</value>

<label>drizzle</label>

<category>

.......

.......

<category>

<value>17</value>

<label>ground\_clutter</label>

<category>

</legend>

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

<thresholding field="DBZ">

<field\_used>

<name>NCP</name>

<min\_val>0.15</min\_val>

</field\_used>

<field\_used>

<name>SNR</name>

<min\_val>-3.0</min\_val>

</field\_used>

<note>NCP only checked if DBZ > 40</note>

</thresholding>

## Ray qualifier fields

Ray qualifier fields are 1-dimensional arrays, with the dimension **time**.

Ray qualifier fields are similar to standard ray meta-data, such as elevation or azimuth. However, qualifier fields are not included in the standard meta-data for a ray, and are custom additions to a data set.

Examples of ray qualifier fields are:

* environmental measurements, for example temperature or rain rate at the radar site.
* sea-surface temperature, or terrain height, for a downward pointing radar from an airborne or spaceborne platform.

Ray qualifier fields 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.

Ray qualifier fields have the following attributes:

| Attribute name | Type | Convention | Description |
| --- | --- | --- | --- |
| long\_name | string | CF | Long name describing the field. Any string is appropriate. Although this is an optional attribute, its use is strongly encouraged. |
| standard\_name | string | CF | CF standard name for field.  See section 6.2. |
| units | string | CF | Units for field |
| \_FillValue (or missing\_value) | same type as field data | CF | Indicates data are missing at this range bin. Use of \_FillValue is preferred. Only use one or the other. |
| scale\_factor | float | CF | Float value = (integer value) \* scale\_factor + add\_offset Only applies to integer types. |
| add\_offset | float | CF |
| 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. |
| flag\_values | array of same type as field data | CF | Array of flag values. These values have special meaning, as documented in flag\_meanings. |
| flag\_meanings | string | CF | Meaning of flag\_values or flag\_masks. Space-separated list in string. |
| flag\_masks | array of same type as field data | CF | Valid bit-wise masks used in a flag field that is comprised of bit-wise combinations of mask values. See flag\_meanings. |

A ray qualifier field may be identified as any field with dimension time, that is not one of the standard time-based metadata fields.

The following is the list of standard time-based metadata fields:

time, elevation, azimuth, ray\_n\_gates, ray\_start\_index, latitude, longitude, altitude, altitude\_agl, scan\_rate, antenna\_transition, georefs\_applied, heading, roll, pitch, drift, rotation, tilt, pulse\_width, rx\_range\_resolution, prt, prt\_ratio, nyquist\_velocity, unambiguous\_range, n\_samples, radar\_measured\_sky\_noise, radar\_measured\_cold\_noise, radar\_measured\_hot\_noise, radar\_measured\_transmit\_power\_h, radar\_measured\_transmit\_power\_v, r\_calib\_index, eastward\_velocity, northward\_velocity, vertical\_velocity, eastward\_wind, northward\_wind, vertical\_wind, heading\_rate, roll\_rate, pitch\_rate

## Spectra field variables

Support for spectra has been added to CfRadial version 1.4.

Because spectra are potentially voluminous, they are stored in a sparse array, such that a spectrum need only be stored for gate locations of interest, rather than at all locations. This is achieved through the use of an integer index variable, which exists for every gate location. If the index is set to -1, no spectrum is stored for that gate. If the index is 0 or positive, it indicates the location of the spectrum in the contiguous array.

### Spectrum index variables

The index variable for a spectrum is an integer variable, stored as a regular *field data variable* as specified in section 4.10 above. There is a spectrum index value for every gate in the CfRadial volume.

The \_FillValue attribute for the index should be set to -1.

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

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

A single index variable can apply to a number of spectra fields.

### Spectrum fields

A spectrum variable will have the dimensions:

(n\_spectra, spectrum\_n\_samples).

These dimension names are not fixed. Any suitable names can be used. There will be multiple dimensions if different spectral data sets have different shapes.

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

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

### Spectrum field attributes

Spectrum fields have the following attributes:

| Attribute name | Type | Convention | Description |
| --- | --- | --- | --- |
| is\_spectrum | string | CF/Radial | Set to “true” to indicate that this is a spectrum variable. |
| long\_name | string | CF | Long name describing the field. Any string is appropriate. Although this is an optional attribute, its use is strongly encouraged. |
| standard\_name | string | CF | CF standard name for field, or from section 6. |
| units | string | CF | Units for field |
| \_FillValue or missing\_value | same type as field data | CF | Indicates data are missing for a given sample. \_FillValue is preferred. |
| scale\_factor | float | CF | See section 4.10. Float value = (integer value) \* scale\_factor + add\_offset. Only applied to integer types. |
| add\_offset | float | CF |
| coordinates | string | CF | See section 4.10. |
| index\_var\_name | string | CF/Radial | Name of index variable for this field. NOTE: a single index variable may be used for multiple spectrum fields. |
| fft\_length | int | CF/Radial | Length used to compute this spectrum |
| block\_avg\_length | int | CF/Radial | Number of block spectra averaged for each output spectrum. |

# 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.

## 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 |  |
| rx\_range\_resolution | (time) | float | meters | Resolution of the raw receiver samples. If missing, assumed to be meters\_between\_gates (4.4.2). Raw data may be resampled before data storage. |
| prt\_mode | (sweep, string\_length) | char |  | Pulsing mode Options are: *“fixed”, “staggered”, “dual”, “hybrid”*. Assumed “*fixed*” if missing. May also be more complicated pulsing schemes, such as HHVV, HHVVH etc. |
| prt | (time) | float | seconds | Pulse repetition time. For staggered prt, also see prt\_ratio. |
| prt\_ratio | (time) | float |  | Ratio of prt/prt2. For dual/staggered prt mode. |
| prt\_sequence | (time, n\_prts) | float | seconds | Sequence of prts used. Optional for fixed, staggered and dual, which can make use of ‘prt’ and ‘prt\_ratio’. Required for more complicated pulsing schemes. |
| polarization\_mode | (sweep, string\_length) | char |  | Options are: *“horizontal”, “vertical”, “hv\_alt”, “hv\_sim”, “circular”* Assumed “*horizontal*” if missing. |
| polarization\_sequence | (sweep, n\_prts) | char |  | This only applies if prt\_mode is set to “hybrid”. As an example, the form of it would be [‘H’,’H’,’V’,’V’,’H’] for HHVVH pulsing. |
| nyquist\_velocity | (time) | float | m/s | Unambiguous velocity. This is the effective nyquist velocity after unfolding. See also the field-specific attributes fold\_limit\_lower and fold\_limit\_upper, 4.10. |
| unambiguous\_range | (time) | float | meters | Unambiguous range |
| n\_samples | (time) | int |  | Number of samples used to compute moments |
| radar\_measured\_sky\_noise | (time) | float | dBm | Noise measured at the receiver when connected to the antenna with no noise source connected. |
| radar\_measured\_cold\_noise | (time) | float | dBm | Noise measured at the receiver when connected to the noise source, but it is not enabled. |
| radar\_measured\_hot\_noise | (time) | float | dBm | Noise measured at the receiver when it is connected to the noise source and the noise source is on. |

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.

## 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.

## 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.

## The *radar\_calibration* sub-convention

Variables in this convention will have the string attribute **meta\_group**, set to the convention name “**radar\_calibration”**.

### Dimensions

| Dimension name | Description |
| --- | --- |
| r\_calib | The number of calibrations available |

Note: required if any radar\_calibration variables are included..

### 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 | Derived antenna gain V channel |
| r\_calib\_xmit\_power\_h | (r\_calib) | float | dBm | Transmit power H channel |
| r\_calib\_xmit\_power\_v | (r\_calib) | float | dBm | Transmit power V channel |
| 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 | 2-way waveguide loss  measurement plane to feed horn, 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 | 2-way radome loss V channel |
| r\_calib\_receiver\_mismatch\_loss | (r\_calib) | float | dB | Receiver filter bandwidth mismatch loss |
| r\_calib\_receiver\_mismatch\_loss\_h | (r\_calib) | float | dB | Receiver filter bandwidth mismatch loss H channel |
| r\_calib\_receiver\_mismatch\_loss\_v | (r\_calib) | float | dB | Receiver filter bandwidth mismatch loss|V channel |
| 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 | Radar constant V channel |
| r\_calib\_probert\_jones\_correction | (r\_calib) | float | dB | Probert Jones antenna correction factor. |
| r\_calib\_dielectric\_factor\_used | (r\_calib) | float |  | This is |K2|in the radar equation. |
| r\_calib\_noise\_hc | (r\_calib) | float | dBm | Measured noise level H co-pol channel |
| r\_calib\_noise\_vc | (r\_calib) | float | dBm | Measured noise level V co-pol channel |
| r\_calib\_noise\_hx | (r\_calib) | float | dBm | Measured noise level H cross-pol channel |
| r\_calib\_noise\_vx | (r\_calib) | float | dBm | Measured noise level V cross-pol channel |
| 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 | Measured receiver gain V co-pol channel |
| r\_calib\_receiver\_gain\_hx | (r\_calib) | float | dB | Measured receiver gain H cross-pol channel |
| r\_calib\_receiver\_gain\_vx | (r\_calib) | float | dB | Measured receiver gain V cross-pol channel |
| r\_calib\_base\_1km\_hc | (r\_calib) | float | dBZ | reflectivity at 1km for SNR=0dB noise-corrected H co-pol channel |
| r\_calib\_base\_1km\_vc | (r\_calib) | float | dBZ | reflectivity at 1km for SNR=0dB  noise-corrected V co-pol channel |
| r\_calib\_base\_1km\_hx | (r\_calib) | float | dBZ | reflectivity at 1km for SNR=0dB  noise-corrected H cross-pol channel |
| r\_calib\_base\_1km\_vx | (r\_calib) | float | dBZ | reflectivity at 1km for SNR=0dB  noise-corrected V cross-pol channel |
| r\_calib\_sun\_power\_hc | (r\_calib) | float | dBm | Calibrated sun power H co-pol channel |
| r\_calib\_sun\_power\_vc | (r\_calib) | float | dBm | Calibrated sun power V co-pol channel |
| r\_calib\_sun\_power\_hx | (r\_calib) | float | dBm | Calibrated sun power H cross-pol channel |
| r\_calib\_sun\_power\_vx | (r\_calib) | float | dBm | Calibrated sun power V cross-pol channel |
| r\_calib\_noise\_source\_power\_h | (r\_calib) | float | dBm | Noise source power H channel |
| r\_calib\_noise\_source\_power\_v | (r\_calib) | float | dBm | Noise source power 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 | Power measurement loss in coax and connectors 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 | Coupler loss into waveguide 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 | Calibration test power 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 |  | Computed receiver slope, ideally 1.0 V co-pol channel |
| r\_calib\_receiver\_slope\_hx | (r\_calib) | float |  | Computed receiver slope, ideally 1.0 H cross-pol channel |
| r\_calib\_receiver\_slope\_vx | (r\_calib) | float |  | Computed receiver slope, ideally 1.0 V cross-pol channel |

Note: all items are optional.

## 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.

## 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: no items are required.

## 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 |
| 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.

# Standard names

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

## Standard names for moments variables

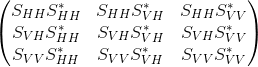
This section lists the proposed standard names for moments data and other fields derived from the raw radar data.

| Standard name | Short name | Units | Already in CF? |
| --- | --- | --- | --- |
| equivalent\_reflectivity\_factor | DBZ | dBZ | yes |
| linear\_equivalent\_reflectivity\_factor | Z | Z | no |
| radial\_velocity\_of\_scatterers\_ away\_from\_instrument | VEL | m/s | yes |
| doppler\_spectrum\_width | WIDTH | m/s | no |
| log\_differential\_reflectivity\_hv | ZDR | dB | no |
| log\_linear\_depolarization\_ratio\_hv | LDR | dB | no |
| log\_linear\_depolarization\_ratio\_h | LDRH | dB | no |
| log\_linear\_depolarization\_ratio\_v | LDRV | dB | no |
| differential\_phase\_hv | PHIDP | degrees | no |
| specific\_differential\_phase\_hv | KDP | degrees/km | no |
| cross\_polar\_differential\_phase | PHIHX | degrees | no |
| cross\_correlation\_ratio\_hv | RHOHV |  | no |
| co\_to\_cross\_polar\_correlation\_ratio\_h | RHOHX |  | no |
| co\_to\_cross\_polar\_correlation\_ratio\_v | RHOXV |  | no |
| log\_power | DBM | dBm | no |
| log\_power\_co\_polar\_h | DBMHC | dBm | no |
| log\_power\_cross\_polar\_h | DBMHX | dBm | no |
| log\_power\_co\_polar\_v | DBMVC | dBm | no |
| log\_power\_cross\_polar\_v | DBMVX | dBm | no |
| linear\_power | PWR | mW | no |
| linear\_power\_co\_polar\_h | PWRHC | mW | no |
| linear\_power\_cross\_polar\_h | PWRHX | mW | no |
| linear\_power\_co\_polar\_v | PWRVC | mW | no |
| linear\_power\_cross\_polar\_v | PWRVX | mW | no |
| signal\_to\_noise\_ratio | SNR | dB | no |
| signal\_to\_noise\_ratio\_co\_polar\_h | SNRHC | dB | no |
| signal\_to\_noise\_ratio\_cross\_polar\_h | SNRHX | dB | no |
| signal\_to\_noise\_ratio\_co\_polar\_v | SNRVC | dB | no |
| signal\_to\_noise\_ratio\_cross\_polar\_v | SNRVX | dB | no |
| normalized\_coherent\_power (Note: this is also known as signal-quality-index) | NCP (SQI) |  | no |
| corrected\_equivalent\_reflectivity\_factor | DBZc | dBZ | no |
| corrected\_radial\_velocity\_of\_scatterers\_ away\_from\_instrument | VELc | m/s | no |
| corrected\_log\_differential\_reflectivity\_hv | ZDRc | dB | no |
| radar\_estimated\_rain\_rate | RRR | mm/hr | no |
| rain\_rate | RR | kg/m2/s | yes |
| radar\_echo\_classification (should be used for PID, HCA, HID etc) | REC | legend | no |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Standard names for spectra variables

This section lists the proposed standard names for spectra field variables. After assuming reciprocity there are 6 unique elements of the covariance matrix and the following names reflect those combinations. In the short name, the notation due to Bringi and Chandrasekar where SVH refers to the backscattering element where horizontal polarization is transmitted, but vertical polarization is received. The \* denotes complex conjugate.

The covariance matrix elements are given as



Elements below the diagonal are conjugate symmetric to elements above the diagonal.

| Standard name | Suggested short name | Units | Already in CF? |
| --- | --- | --- | --- |
| spectrum\_of\_copolar\_horizontal\_to\_copolar\_horizontal | SPEC\_HH\_HH\* |  | no |
| spectrum\_of\_copolar\_horizontal\_to\_crosspolar\_vertical | SPEC\_HH\_VH\* |  | no |
| spectrum\_of\_copolar\_horizontal\_to\_copolar\_vertical | SPEC\_HH\_VV\* |  | no |
| spectrum\_of\_crosspolar\_vertical\_to\_crosspolar\_vertical | SPEC\_VH\_VH\* |  | no |
| spectrum\_of\_crosspolar\_vertical\_to\_copolar\_vertical | SPEC\_VH\_VV\* |  | no |
| spectrum\_of\_copolar\_vertical\_to\_copolar\_vertical | SPEC\_VV\_VV\* |  | no |

## 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.

The following long names are suggested for metadata variables.

| **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 |
| 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 |
| 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* |  |
| 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 |
| 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 |
| 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 |
| 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 |

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

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

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



Figure .: 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.

## 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°.

### 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 **X**i(*r*, *λ*, *φ*) and **X** (*x*, *y*, *z*) is as follows:







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.

### 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:



where  is the pseudo radius of earth. See Rinehart 2004, Chapter 3, for more details.

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





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

## Moving platforms

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

* longitude of instrument
* latitude of instrument
* altitude of instrument
* rotation and tilt of the beam (see above)
* roll, pitch and heading of the platform
* platform motion (UG, VG, WG)
* air motion (Uair, Vair, Wair)

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



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.



## 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.

### Coordinate systems

In addition to the previously-defined **Xi** and **X** coordinate systems, the following intermediate right-handed coordinate systems need to be defined to account for a moving, non-leveled platform:

* **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.
* **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 **Xi** to **X** via **X**a and **X**h.

### 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.

### 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, **X**a, is defined by the right side, (Xa), the heading, (Ya), and the zenith, (Za).

The origin of **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.

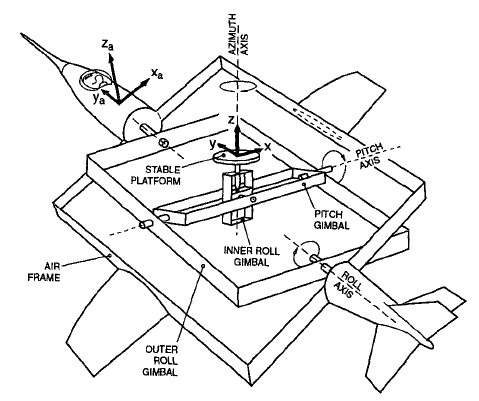


Figure . Moving platform axis definitions and reference frame (reproduced from Lee et al., 1994,originally from Axford, 1968) ©American Meteorological Society. Reprinted with permission.

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

### The sensor coordinate system

In the sensor coordinate system, **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 **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 **X**i can be represented in (*r*, *λ*, *φ*) in **X**.

Table .: Characteristics of 4 types of sensors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensor Type | Type X | Type Y | Type Y-prime | Type Z |
| Principal Axis | Xa | Ya | Ya | Za |
| Reference Plane | YaZa | ZaXa | ZaXa | XaYa |
| 0° Rotation Angle | +Za | +Xa | +Za | +Ya |
| 90° Rotation Angle | +Ya | +Za | +Xa | +Xa |
| 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 |

## Coordinate transformation sequence

The following transformations are carried out to transform the geometry from the instrument-based (**X**i) to the earth-based coordinate system (**X**):

* translate from **X**i to **X**a
* rotate from **X**a to **X**

### Transformation from Xi to Xa

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

#### Type Z sensors

The characteristics are:

* the primary axis is Za
* the reference plane is (Xa, Ya)
* the rotation angle *θ* is 0 in the (Ya, Za) 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 (Xa, Ya) plane, positive above it (for +Za) and negative below it.

The transformation to Χa coordinates is:



#### Type Y sensors

The characteristics are:

* the primary axis is Ya
* the reference plane is (Za, Xa)
* the rotation angle *θ* is 0 in the (Za, Xa) plane, i.e. along the +Xa axis. Rotation increases clockwise from +X, when looking from +Y.
* the tilt angle *τ* is 0 in the (Za, Xa) plane, positive for +Ya.

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

The transformation to **Χ**a coordinates is:



* + - 1. **Type Y-prime sensors**

The characteristics are:

the primary axis is Ya

the reference plane is (Za, Xa)

the rotation angle *θ* is 0 in the (Ya, Za) plane, i.e. along the +Za axis. Rotation increases clockwise from +Z, when looking from -Y.

the tilt angle *τ* is 0 in the (Za, Xa) plane, positive for +Ya.

***Note that the definition of*** ***θ is the convention defined in Lee et al. (1994***

The transformation to **Χ**a coordinates is:



#### Type X sensors

The characteristics are:

* the primary axis is Xa
* the reference plane is (Ya, Za)
* the rotation angle *θ* is 0 in the (Ya, Za) plane, i.e. along the +Za axis. Rotation increases clockwise from +Za, when looking from +Xa.
* the tilt angle *τ* is 0 in the (Ya, Za) plane, positive for +Xa.

The transformation to Χa coordinates is:



### Rotating from Xa to X

Rotating **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*.
* 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:



The transformation matrix for removing the pitch component is:



The transformation matrix for removing the heading component is:



We apply these transformations consecutively:





## Summary of transforming from Xi to X

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

### For type Z radars:



### For type Y radars:



### For type Y-prime radars:



### For type X radars:



### Computing earth-relative azimuth and elevation

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



## Summary of symbol definitions

**Χ**i: instrument-relative coordinate system, (*r*, *θ*, *τ*) or (*r*, *λ*, *φ*)

**Χ**a: platform-relative coordinate system (*x*a, *y*a, *z*a) – see figure 7.2

**Χ**h: coordinate system relative to level platform (no roll or pitch) with heading *H*.

**Χ**: 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 = 

# 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

# Example ncdump of CfRadial file

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

netcdf cfrad.20150626\_120415.982\_to\_20150626\_120831.578\_KDDC\_Surveillance\_SUR {

dimensions:

time = 4200 ;

range = 1832 ;

n\_points = 6087840 ;

sweep = 9 ;

string\_length\_8 = 8 ;

string\_length\_32 = 32 ;

status\_xml\_length = 1 ;

r\_calib = 1 ;

frequency = 1 ;

variables:

int volume\_number ;

volume\_number:long\_name = "data\_volume\_index\_number" ;

volume\_number:units = "" ;

volume\_number:\_FillValue = -9999 ;

char platform\_type(string\_length\_32) ;

platform\_type:long\_name = "platform\_type" ;

platform\_type:options = "fixed, vehicle, ship, aircraft\_fore, aircraft\_aft, aircraft\_tail, aircraft\_belly, aircraft\_roof, aircraft\_nose, satellite\_orbit, satellite\_geostat" ;

char primary\_axis(string\_length\_32) ;

primary\_axis:long\_name = "primary\_axis\_of\_rotation" ;

primary\_axis:options = "axis\_z, axis\_y, axis\_x, axis\_z\_prime, axis\_y\_prime, axis\_x\_prime" ;

char status\_xml(status\_xml\_length) ;

status\_xml:long\_name = "status\_of\_instrument" ;

char instrument\_type(string\_length\_32) ;

instrument\_type:long\_name = "type\_of\_instrument" ;

instrument\_type:options = "radar, lidar" ;

instrument\_type:meta\_group = "instrument\_parameters" ;

float radar\_antenna\_gain\_h ;

radar\_antenna\_gain\_h:long\_name = "nominal\_radar\_antenna\_gain\_h\_channel" ;

radar\_antenna\_gain\_h:units = "db" ;

radar\_antenna\_gain\_h:\_FillValue = -9999.f ;

radar\_antenna\_gain\_h:meta\_group = "radar\_parameters" ;

float radar\_antenna\_gain\_v ;

radar\_antenna\_gain\_v:long\_name = "nominal\_radar\_antenna\_gain\_v\_channel" ;

radar\_antenna\_gain\_v:units = "db" ;

radar\_antenna\_gain\_v:\_FillValue = -9999.f ;

radar\_antenna\_gain\_v:meta\_group = "radar\_parameters" ;

float radar\_beam\_width\_h ;

radar\_beam\_width\_h:long\_name = "half\_power\_radar\_beam\_width\_h\_channel" ;

radar\_beam\_width\_h:units = "degrees" ;

radar\_beam\_width\_h:\_FillValue = -9999.f ;

radar\_beam\_width\_h:meta\_group = "radar\_parameters" ;

float radar\_beam\_width\_v ;

radar\_beam\_width\_v:long\_name = "half\_power\_radar\_beam\_width\_v\_channel" ;

radar\_beam\_width\_v:units = "degrees" ;

radar\_beam\_width\_v:\_FillValue = -9999.f ;

radar\_beam\_width\_v:meta\_group = "radar\_parameters" ;

float radar\_rx\_bandwidth ;

radar\_rx\_bandwidth:long\_name = "radar\_receiver\_bandwidth" ;

radar\_rx\_bandwidth:units = "s-1" ;

radar\_rx\_bandwidth:\_FillValue = -9999.f ;

radar\_rx\_bandwidth:meta\_group = "radar\_parameters" ;

char time\_coverage\_start(string\_length\_32) ;

time\_coverage\_start:long\_name = "data\_volume\_start\_time\_utc" ;

time\_coverage\_start:comment = "ray times are relative to start time in secs" ;

char time\_coverage\_end(string\_length\_32) ;

time\_coverage\_end:long\_name = "data\_volume\_end\_time\_utc" ;

float frequency(frequency) ;

frequency:long\_name = "transmission\_frequency" ;

frequency:units = "s-1" ;

frequency:\_FillValue = -9999.f ;

frequency:meta\_group = "instrument\_parameters" ;

int grid\_mapping ;

grid\_mapping:grid\_mapping\_name = "radar\_lidar\_radial\_scan" ;

grid\_mapping:longitude\_of\_projection\_origin = -99.9688873291016 ;

grid\_mapping:latitude\_of\_projection\_origin = 37.7608337402344 ;

grid\_mapping:altitude\_of\_projection\_origin = 813. ;

grid\_mapping:false\_northing = 0. ;

grid\_mapping:false\_easting = 0. ;

double latitude ;

latitude:long\_name = "latitude" ;

latitude:units = "degrees\_north" ;

latitude:\_FillValue = -9999. ;

double longitude ;

longitude:long\_name = "longitude" ;

longitude:units = "degrees\_east" ;

longitude:\_FillValue = -9999. ;

double altitude ;

altitude:long\_name = "altitude" ;

altitude:units = "meters" ;

altitude:\_FillValue = -9999. ;

altitude:positive = "up" ;

double altitude\_agl ;

altitude\_agl:long\_name = "altitude\_above\_ground\_level" ;

altitude\_agl:units = "meters" ;

altitude\_agl:\_FillValue = -9999. ;

altitude\_agl:positive = "up" ;

int sweep\_number(sweep) ;

sweep\_number:long\_name = "sweep\_index\_number\_0\_based" ;

sweep\_number:units = "" ;

sweep\_number:\_FillValue = -9999 ;

char sweep\_mode(sweep, string\_length\_32) ;

sweep\_mode:long\_name = "scan\_mode\_for\_sweep" ;

sweep\_mode:options = "sector, coplane, rhi, vertical\_pointing, idle, azimuth\_surveillance, elevation\_surveillance, sunscan, pointing, calibration, manual\_ppi, manual\_rhi, sunscan\_rhi" ;

char polarization\_mode(sweep, string\_length\_32) ;

polarization\_mode:long\_name = "polarization\_mode\_for\_sweep" ;

polarization\_mode:options = "horizontal, vertical, hv\_alt, hv\_sim, circular" ;

polarization\_mode:meta\_group = "radar\_parameters" ;

char prt\_mode(sweep, string\_length\_32) ;

prt\_mode:long\_name = "transmit\_pulse\_mode" ;

prt\_mode:options = "fixed, staggered, dual" ;

prt\_mode:meta\_group = "radar\_parameters" ;

char follow\_mode(sweep, string\_length\_32) ;

follow\_mode:long\_name = "follow\_mode\_for\_scan\_strategy" ;

follow\_mode:options = "none, sun, vehicle, aircraft, target, manual" ;

follow\_mode:meta\_group = "instrument\_parameters" ;

float fixed\_angle(sweep) ;

fixed\_angle:long\_name = "ray\_target\_fixed\_angle" ;

fixed\_angle:units = "degrees" ;

fixed\_angle:\_FillValue = -9999.f ;

float target\_scan\_rate(sweep) ;

target\_scan\_rate:long\_name = "target\_scan\_rate\_for\_sweep" ;

target\_scan\_rate:units = "degrees per second" ;

target\_scan\_rate:\_FillValue = -9999.f ;

int sweep\_start\_ray\_index(sweep) ;

sweep\_start\_ray\_index:long\_name = "index\_of\_first\_ray\_in\_sweep" ;

sweep\_start\_ray\_index:units = "" ;

sweep\_start\_ray\_index:\_FillValue = -9999 ;

int sweep\_end\_ray\_index(sweep) ;

sweep\_end\_ray\_index:long\_name = "index\_of\_last\_ray\_in\_sweep" ;

sweep\_end\_ray\_index:units = "" ;

sweep\_end\_ray\_index:\_FillValue = -9999 ;

char rays\_are\_indexed(sweep, string\_length\_8) ;

rays\_are\_indexed:long\_name = "flag\_for\_indexed\_rays" ;

float ray\_angle\_res(sweep) ;

ray\_angle\_res:long\_name = "angular\_resolution\_between\_rays" ;

ray\_angle\_res:units = "degrees" ;

ray\_angle\_res:\_FillValue = -9999.f ;

char r\_calib\_time(r\_calib, string\_length\_32) ;

r\_calib\_time:long\_name = "radar\_calibration\_time\_utc" ;

r\_calib\_time:meta\_group = "radar\_calibration" ;

float r\_calib\_pulse\_width(r\_calib) ;

r\_calib\_pulse\_width:long\_name = "radar\_calibration\_pulse\_width" ;

r\_calib\_pulse\_width:units = "seconds" ;

r\_calib\_pulse\_width:meta\_group = "radar\_calibration" ;

r\_calib\_pulse\_width:\_FillValue = -9999.f ;

float r\_calib\_xmit\_power\_h(r\_calib) ;

r\_calib\_xmit\_power\_h:long\_name = "calibrated\_radar\_xmit\_power\_h\_channel" ;

r\_calib\_xmit\_power\_h:units = "dBm" ;

r\_calib\_xmit\_power\_h:meta\_group = "radar\_calibration" ;

r\_calib\_xmit\_power\_h:\_FillValue = -9999.f ;

float r\_calib\_xmit\_power\_v(r\_calib) ;

r\_calib\_xmit\_power\_v:long\_name = "calibrated\_radar\_xmit\_power\_v\_channel" ;

r\_calib\_xmit\_power\_v:units = "dBm" ;

r\_calib\_xmit\_power\_v:meta\_group = "radar\_calibration" ;

r\_calib\_xmit\_power\_v:\_FillValue = -9999.f ;

float r\_calib\_two\_way\_waveguide\_loss\_h(r\_calib) ;

r\_calib\_two\_way\_waveguide\_loss\_h:long\_name = "radar\_calibration\_two\_way\_waveguide\_loss\_h\_channel" ;

r\_calib\_two\_way\_waveguide\_loss\_h:units = "db" ;

r\_calib\_two\_way\_waveguide\_loss\_h:meta\_group = "radar\_calibration" ;

r\_calib\_two\_way\_waveguide\_loss\_h:\_FillValue = -9999.f ;

float r\_calib\_two\_way\_waveguide\_loss\_v(r\_calib) ;

r\_calib\_two\_way\_waveguide\_loss\_v:long\_name = "radar\_calibration\_two\_way\_waveguide\_loss\_v\_channel" ;

r\_calib\_two\_way\_waveguide\_loss\_v:units = "db" ;

r\_calib\_two\_way\_waveguide\_loss\_v:meta\_group = "radar\_calibration" ;

r\_calib\_two\_way\_waveguide\_loss\_v:\_FillValue = -9999.f ;

float r\_calib\_two\_way\_radome\_loss\_h(r\_calib) ;

r\_calib\_two\_way\_radome\_loss\_h:long\_name = "radar\_calibration\_two\_way\_radome\_loss\_h\_channel" ;

r\_calib\_two\_way\_radome\_loss\_h:units = "db" ;

r\_calib\_two\_way\_radome\_loss\_h:meta\_group = "radar\_calibration" ;

r\_calib\_two\_way\_radome\_loss\_h:\_FillValue = -9999.f ;

float r\_calib\_two\_way\_radome\_loss\_v(r\_calib) ;

r\_calib\_two\_way\_radome\_loss\_v:long\_name = "radar\_calibration\_two\_way\_radome\_loss\_v\_channel" ;

r\_calib\_two\_way\_radome\_loss\_v:units = "db" ;

r\_calib\_two\_way\_radome\_loss\_v:meta\_group = "radar\_calibration" ;

r\_calib\_two\_way\_radome\_loss\_v:\_FillValue = -9999.f ;

float r\_calib\_receiver\_mismatch\_loss(r\_calib) ;

r\_calib\_receiver\_mismatch\_loss:long\_name = "radar\_calibration\_receiver\_mismatch\_loss" ;

r\_calib\_receiver\_mismatch\_loss:units = "db" ;

r\_calib\_receiver\_mismatch\_loss:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_mismatch\_loss:\_FillValue = -9999.f ;

float r\_calib\_radar\_constant\_h(r\_calib) ;

r\_calib\_radar\_constant\_h:long\_name = "calibrated\_radar\_constant\_h\_channel" ;

r\_calib\_radar\_constant\_h:units = "db" ;

r\_calib\_radar\_constant\_h:meta\_group = "radar\_calibration" ;

r\_calib\_radar\_constant\_h:\_FillValue = -9999.f ;

float r\_calib\_radar\_constant\_v(r\_calib) ;

r\_calib\_radar\_constant\_v:long\_name = "calibrated\_radar\_constant\_v\_channel" ;

r\_calib\_radar\_constant\_v:units = "db" ;

r\_calib\_radar\_constant\_v:meta\_group = "radar\_calibration" ;

r\_calib\_radar\_constant\_v:\_FillValue = -9999.f ;

float r\_calib\_antenna\_gain\_h(r\_calib) ;

r\_calib\_antenna\_gain\_h:long\_name = "calibrated\_radar\_antenna\_gain\_h\_channel" ;

r\_calib\_antenna\_gain\_h:units = "db" ;

r\_calib\_antenna\_gain\_h:meta\_group = "radar\_calibration" ;

r\_calib\_antenna\_gain\_h:\_FillValue = -9999.f ;

float r\_calib\_antenna\_gain\_v(r\_calib) ;

r\_calib\_antenna\_gain\_v:long\_name = "calibrated\_radar\_antenna\_gain\_v\_channel" ;

r\_calib\_antenna\_gain\_v:units = "db" ;

r\_calib\_antenna\_gain\_v:meta\_group = "radar\_calibration" ;

r\_calib\_antenna\_gain\_v:\_FillValue = -9999.f ;

float r\_calib\_noise\_hc(r\_calib) ;

r\_calib\_noise\_hc:long\_name = "calibrated\_radar\_receiver\_noise\_h\_co\_polar\_channel" ;

r\_calib\_noise\_hc:units = "dBm" ;

r\_calib\_noise\_hc:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_hc:\_FillValue = -9999.f ;

float r\_calib\_noise\_vc(r\_calib) ;

r\_calib\_noise\_vc:long\_name = "calibrated\_radar\_receiver\_noise\_v\_co\_polar\_channel" ;

r\_calib\_noise\_vc:units = "dBm" ;

r\_calib\_noise\_vc:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_vc:\_FillValue = -9999.f ;

float r\_calib\_noise\_hx(r\_calib) ;

r\_calib\_noise\_hx:long\_name = "calibrated\_radar\_receiver\_noise\_h\_cross\_polar\_channel" ;

r\_calib\_noise\_hx:units = "dBm" ;

r\_calib\_noise\_hx:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_hx:\_FillValue = -9999.f ;

float r\_calib\_noise\_vx(r\_calib) ;

r\_calib\_noise\_vx:long\_name = "calibrated\_radar\_receiver\_noise\_v\_cross\_polar\_channel" ;

r\_calib\_noise\_vx:units = "dBm" ;

r\_calib\_noise\_vx:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_vx:\_FillValue = -9999.f ;

float r\_calib\_receiver\_gain\_hc(r\_calib) ;

r\_calib\_receiver\_gain\_hc:long\_name = "calibrated\_radar\_receiver\_gain\_h\_co\_polar\_channel" ;

r\_calib\_receiver\_gain\_hc:units = "db" ;

r\_calib\_receiver\_gain\_hc:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_gain\_hc:\_FillValue = -9999.f ;

float r\_calib\_receiver\_gain\_vc(r\_calib) ;

r\_calib\_receiver\_gain\_vc:long\_name = "calibrated\_radar\_receiver\_gain\_v\_co\_polar\_channel" ;

r\_calib\_receiver\_gain\_vc:units = "db" ;

r\_calib\_receiver\_gain\_vc:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_gain\_vc:\_FillValue = -9999.f ;

float r\_calib\_receiver\_gain\_hx(r\_calib) ;

r\_calib\_receiver\_gain\_hx:long\_name = "calibrated\_radar\_receiver\_gain\_h\_cross\_polar\_channel" ;

r\_calib\_receiver\_gain\_hx:units = "db" ;

r\_calib\_receiver\_gain\_hx:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_gain\_hx:\_FillValue = -9999.f ;

float r\_calib\_receiver\_gain\_vx(r\_calib) ;

r\_calib\_receiver\_gain\_vx:long\_name = "calibrated\_radar\_receiver\_gain\_v\_cross\_polar\_channel" ;

r\_calib\_receiver\_gain\_vx:units = "db" ;

r\_calib\_receiver\_gain\_vx:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_gain\_vx:\_FillValue = -9999.f ;

float r\_calib\_base\_dbz\_1km\_hc(r\_calib) ;

r\_calib\_base\_dbz\_1km\_hc:long\_name = "radar\_reflectivity\_at\_1km\_at\_zero\_snr\_h\_co\_polar\_channel" ;

r\_calib\_base\_dbz\_1km\_hc:units = "dBZ" ;

r\_calib\_base\_dbz\_1km\_hc:meta\_group = "radar\_calibration" ;

r\_calib\_base\_dbz\_1km\_hc:\_FillValue = -9999.f ;

float r\_calib\_base\_dbz\_1km\_vc(r\_calib) ;

r\_calib\_base\_dbz\_1km\_vc:long\_name = "radar\_reflectivity\_at\_1km\_at\_zero\_snr\_v\_co\_polar\_channel" ;

r\_calib\_base\_dbz\_1km\_vc:units = "dBZ" ;

r\_calib\_base\_dbz\_1km\_vc:meta\_group = "radar\_calibration" ;

r\_calib\_base\_dbz\_1km\_vc:\_FillValue = -9999.f ;

float r\_calib\_base\_dbz\_1km\_hx(r\_calib) ;

r\_calib\_base\_dbz\_1km\_hx:long\_name = "radar\_reflectivity\_at\_1km\_at\_zero\_snr\_h\_cross\_polar\_channel" ;

r\_calib\_base\_dbz\_1km\_hx:units = "dBZ" ;

r\_calib\_base\_dbz\_1km\_hx:meta\_group = "radar\_calibration" ;

r\_calib\_base\_dbz\_1km\_hx:\_FillValue = -9999.f ;

float r\_calib\_base\_dbz\_1km\_vx(r\_calib) ;

r\_calib\_base\_dbz\_1km\_vx:long\_name = "radar\_reflectivity\_at\_1km\_at\_zero\_snr\_v\_cross\_polar\_channel" ;

r\_calib\_base\_dbz\_1km\_vx:units = "dBZ" ;

r\_calib\_base\_dbz\_1km\_vx:meta\_group = "radar\_calibration" ;

r\_calib\_base\_dbz\_1km\_vx:\_FillValue = -9999.f ;

float r\_calib\_sun\_power\_hc(r\_calib) ;

r\_calib\_sun\_power\_hc:long\_name = "calibrated\_radar\_sun\_power\_h\_co\_polar\_channel" ;

r\_calib\_sun\_power\_hc:units = "dBm" ;

r\_calib\_sun\_power\_hc:meta\_group = "radar\_calibration" ;

r\_calib\_sun\_power\_hc:\_FillValue = -9999.f ;

float r\_calib\_sun\_power\_vc(r\_calib) ;

r\_calib\_sun\_power\_vc:long\_name = "calibrated\_radar\_sun\_power\_v\_co\_polar\_channel" ;

r\_calib\_sun\_power\_vc:units = "dBm" ;

r\_calib\_sun\_power\_vc:meta\_group = "radar\_calibration" ;

r\_calib\_sun\_power\_vc:\_FillValue = -9999.f ;

float r\_calib\_sun\_power\_hx(r\_calib) ;

r\_calib\_sun\_power\_hx:long\_name = "calibrated\_radar\_sun\_power\_h\_cross\_polar\_channel" ;

r\_calib\_sun\_power\_hx:units = "dBm" ;

r\_calib\_sun\_power\_hx:meta\_group = "radar\_calibration" ;

r\_calib\_sun\_power\_hx:\_FillValue = -9999.f ;

float r\_calib\_sun\_power\_vx(r\_calib) ;

r\_calib\_sun\_power\_vx:long\_name = "calibrated\_radar\_sun\_power\_v\_cross\_polar\_channel" ;

r\_calib\_sun\_power\_vx:units = "dBm" ;

r\_calib\_sun\_power\_vx:meta\_group = "radar\_calibration" ;

r\_calib\_sun\_power\_vx:\_FillValue = -9999.f ;

float r\_calib\_noise\_source\_power\_h(r\_calib) ;

r\_calib\_noise\_source\_power\_h:long\_name = "radar\_calibration\_noise\_source\_power\_h\_channel" ;

r\_calib\_noise\_source\_power\_h:units = "dBm" ;

r\_calib\_noise\_source\_power\_h:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_source\_power\_h:\_FillValue = -9999.f ;

float r\_calib\_noise\_source\_power\_v(r\_calib) ;

r\_calib\_noise\_source\_power\_v:long\_name = "radar\_calibration\_noise\_source\_power\_v\_channel" ;

r\_calib\_noise\_source\_power\_v:units = "dBm" ;

r\_calib\_noise\_source\_power\_v:meta\_group = "radar\_calibration" ;

r\_calib\_noise\_source\_power\_v:\_FillValue = -9999.f ;

float r\_calib\_power\_measure\_loss\_h(r\_calib) ;

r\_calib\_power\_measure\_loss\_h:long\_name = "radar\_calibration\_power\_measurement\_loss\_h\_channel" ;

r\_calib\_power\_measure\_loss\_h:units = "db" ;

r\_calib\_power\_measure\_loss\_h:meta\_group = "radar\_calibration" ;

r\_calib\_power\_measure\_loss\_h:\_FillValue = -9999.f ;

float r\_calib\_power\_measure\_loss\_v(r\_calib) ;

r\_calib\_power\_measure\_loss\_v:long\_name = "radar\_calibration\_power\_measurement\_loss\_v\_channel" ;

r\_calib\_power\_measure\_loss\_v:units = "db" ;

r\_calib\_power\_measure\_loss\_v:meta\_group = "radar\_calibration" ;

r\_calib\_power\_measure\_loss\_v:\_FillValue = -9999.f ;

float r\_calib\_coupler\_forward\_loss\_h(r\_calib) ;

r\_calib\_coupler\_forward\_loss\_h:long\_name = "radar\_calibration\_coupler\_forward\_loss\_h\_channel" ;

r\_calib\_coupler\_forward\_loss\_h:units = "db" ;

r\_calib\_coupler\_forward\_loss\_h:meta\_group = "radar\_calibration" ;

r\_calib\_coupler\_forward\_loss\_h:\_FillValue = -9999.f ;

float r\_calib\_coupler\_forward\_loss\_v(r\_calib) ;

r\_calib\_coupler\_forward\_loss\_v:long\_name = "radar\_calibration\_coupler\_forward\_loss\_v\_channel" ;

r\_calib\_coupler\_forward\_loss\_v:units = "db" ;

r\_calib\_coupler\_forward\_loss\_v:meta\_group = "radar\_calibration" ;

r\_calib\_coupler\_forward\_loss\_v:\_FillValue = -9999.f ;

float r\_calib\_dbz\_correction(r\_calib) ;

r\_calib\_dbz\_correction:long\_name = "calibrated\_radar\_dbz\_correction" ;

r\_calib\_dbz\_correction:units = "db" ;

r\_calib\_dbz\_correction:meta\_group = "radar\_calibration" ;

r\_calib\_dbz\_correction:\_FillValue = -9999.f ;

float r\_calib\_zdr\_correction(r\_calib) ;

r\_calib\_zdr\_correction:long\_name = "calibrated\_radar\_zdr\_correction" ;

r\_calib\_zdr\_correction:units = "db" ;

r\_calib\_zdr\_correction:meta\_group = "radar\_calibration" ;

r\_calib\_zdr\_correction:\_FillValue = -9999.f ;

float r\_calib\_ldr\_correction\_h(r\_calib) ;

r\_calib\_ldr\_correction\_h:long\_name = "calibrated\_radar\_ldr\_correction\_h\_channel" ;

r\_calib\_ldr\_correction\_h:units = "db" ;

r\_calib\_ldr\_correction\_h:meta\_group = "radar\_calibration" ;

r\_calib\_ldr\_correction\_h:\_FillValue = -9999.f ;

float r\_calib\_ldr\_correction\_v(r\_calib) ;

r\_calib\_ldr\_correction\_v:long\_name = "calibrated\_radar\_ldr\_correction\_v\_channel" ;

r\_calib\_ldr\_correction\_v:units = "db" ;

r\_calib\_ldr\_correction\_v:meta\_group = "radar\_calibration" ;

r\_calib\_ldr\_correction\_v:\_FillValue = -9999.f ;

float r\_calib\_system\_phidp(r\_calib) ;

r\_calib\_system\_phidp:long\_name = "calibrated\_radar\_system\_phidp" ;

r\_calib\_system\_phidp:units = "degrees" ;

r\_calib\_system\_phidp:meta\_group = "radar\_calibration" ;

r\_calib\_system\_phidp:\_FillValue = -9999.f ;

float r\_calib\_test\_power\_h(r\_calib) ;

r\_calib\_test\_power\_h:long\_name = "radar\_calibration\_test\_power\_h\_channel" ;

r\_calib\_test\_power\_h:units = "dBm" ;

r\_calib\_test\_power\_h:meta\_group = "radar\_calibration" ;

r\_calib\_test\_power\_h:\_FillValue = -9999.f ;

float r\_calib\_test\_power\_v(r\_calib) ;

r\_calib\_test\_power\_v:long\_name = "radar\_calibration\_test\_power\_v\_channel" ;

r\_calib\_test\_power\_v:units = "dBm" ;

r\_calib\_test\_power\_v:meta\_group = "radar\_calibration" ;

r\_calib\_test\_power\_v:\_FillValue = -9999.f ;

float r\_calib\_receiver\_slope\_hc(r\_calib) ;

r\_calib\_receiver\_slope\_hc:long\_name = "calibrated\_radar\_receiver\_slope\_h\_co\_polar\_channel" ;

r\_calib\_receiver\_slope\_hc:units = "" ;

r\_calib\_receiver\_slope\_hc:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_slope\_hc:\_FillValue = -9999.f ;

float r\_calib\_receiver\_slope\_vc(r\_calib) ;

r\_calib\_receiver\_slope\_vc:long\_name = "calibrated\_radar\_receiver\_slope\_v\_co\_polar\_channel" ;

r\_calib\_receiver\_slope\_vc:units = "" ;

r\_calib\_receiver\_slope\_vc:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_slope\_vc:\_FillValue = -9999.f ;

float r\_calib\_receiver\_slope\_hx(r\_calib) ;

r\_calib\_receiver\_slope\_hx:long\_name = "calibrated\_radar\_receiver\_slope\_h\_cross\_polar\_channel" ;

r\_calib\_receiver\_slope\_hx:units = "" ;

r\_calib\_receiver\_slope\_hx:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_slope\_hx:\_FillValue = -9999.f ;

float r\_calib\_receiver\_slope\_vx(r\_calib) ;

r\_calib\_receiver\_slope\_vx:long\_name = "calibrated\_radar\_receiver\_slope\_v\_cross\_polar\_channel" ;

r\_calib\_receiver\_slope\_vx:units = "" ;

r\_calib\_receiver\_slope\_vx:meta\_group = "radar\_calibration" ;

r\_calib\_receiver\_slope\_vx:\_FillValue = -9999.f ;

double time(time) ;

time:standard\_name = "time" ;

time:long\_name = "time in seconds since volume start" ;

time:calendar = "gregorian" ;

time:units = "seconds since 2015-06-26T12:04:15Z" ;

time:comment = "times are relative to the volume start\_time" ;

float range(range) ;

range:long\_name = "Range from instrument to center of gate" ;

range:units = "meters" ;

range:spacing\_is\_constant = "true" ;

range:meters\_to\_center\_of\_first\_gate = 2125. ;

range:meters\_between\_gates = 250. ;

int ray\_n\_gates(time) ;

ray\_n\_gates:long\_name = "number\_of\_gates" ;

ray\_n\_gates:units = "" ;

ray\_n\_gates:\_FillValue = -9999 ;

int ray\_start\_index(time) ;

ray\_start\_index:long\_name = "array\_index\_to\_start\_of\_ray" ;

ray\_start\_index:units = "" ;

ray\_start\_index:\_FillValue = -9999 ;

float ray\_start\_range(time) ;

ray\_start\_range:long\_name = "start\_range\_for\_ray" ;

ray\_start\_range:units = "meters" ;

ray\_start\_range:\_FillValue = -9999.f ;

float ray\_gate\_spacing(time) ;

ray\_gate\_spacing:long\_name = "gate\_spacing\_for\_ray" ;

ray\_gate\_spacing:units = "meters" ;

ray\_gate\_spacing:\_FillValue = -9999.f ;

float azimuth(time) ;

azimuth:long\_name = "ray\_azimuth\_angle" ;

azimuth:units = "degrees" ;

azimuth:\_FillValue = -9999.f ;

float elevation(time) ;

elevation:long\_name = "ray\_elevation\_angle" ;

elevation:units = "degrees" ;

elevation:\_FillValue = -9999.f ;

elevation:positive = "up" ;

float pulse\_width(time) ;

pulse\_width:long\_name = "transmitter\_pulse\_width" ;

pulse\_width:units = "seconds" ;

pulse\_width:\_FillValue = -9999.f ;

pulse\_width:meta\_group = "instrument\_parameters" ;

float prt(time) ;

prt:long\_name = "pulse\_repetition\_time" ;

prt:units = "seconds" ;

prt:\_FillValue = -9999.f ;

prt:meta\_group = "instrument\_parameters" ;

float prt\_ratio(time) ;

prt\_ratio:long\_name = "pulse\_repetition\_frequency\_ratio" ;

prt\_ratio:units = "seconds" ;

prt\_ratio:\_FillValue = -9999.f ;

prt\_ratio:meta\_group = "instrument\_parameters" ;

float nyquist\_velocity(time) ;

nyquist\_velocity:long\_name = "unambiguous\_doppler\_velocity" ;

nyquist\_velocity:units = "meters per second" ;

nyquist\_velocity:\_FillValue = -9999.f ;

nyquist\_velocity:meta\_group = "instrument\_parameters" ;

float unambiguous\_range(time) ;

unambiguous\_range:long\_name = "unambiguous\_range" ;

unambiguous\_range:units = "meters" ;

unambiguous\_range:\_FillValue = -9999.f ;

unambiguous\_range:meta\_group = "instrument\_parameters" ;

byte antenna\_transition(time) ;

antenna\_transition:long\_name = "antenna\_is\_in\_transition\_between\_sweeps" ;

antenna\_transition:units = "" ;

antenna\_transition:\_FillValue = -128b ;

antenna\_transition:comment = "1 if antenna is in transition, 0 otherwise" ;

int n\_samples(time) ;

n\_samples:long\_name = "number\_of\_samples\_used\_to\_compute\_moments" ;

n\_samples:units = "" ;

n\_samples:\_FillValue = -9999 ;

n\_samples:meta\_group = "instrument\_parameters" ;

int r\_calib\_index(time) ;

r\_calib\_index:long\_name = "calibration\_data\_array\_index\_per\_ray" ;

r\_calib\_index:units = "" ;

r\_calib\_index:\_FillValue = -9999 ;

r\_calib\_index:meta\_group = "radar\_calibration" ;

r\_calib\_index:comment = "This is the index for the calibration which applies to this ray" ;

float measured\_transmit\_power\_h(time) ;

measured\_transmit\_power\_h:long\_name = "measured\_radar\_transmit\_power\_h\_channel" ;

measured\_transmit\_power\_h:units = "dBm" ;

measured\_transmit\_power\_h:\_FillValue = -9999.f ;

measured\_transmit\_power\_h:meta\_group = "radar\_parameters" ;

float measured\_transmit\_power\_v(time) ;

measured\_transmit\_power\_v:long\_name = "measured\_radar\_transmit\_power\_v\_channel" ;

measured\_transmit\_power\_v:units = "dBm" ;

measured\_transmit\_power\_v:\_FillValue = -9999.f ;

measured\_transmit\_power\_v:meta\_group = "radar\_parameters" ;

float scan\_rate(time) ;

scan\_rate:long\_name = "antenna\_angle\_scan\_rate" ;

scan\_rate:units = "degrees per second" ;

scan\_rate:\_FillValue = -9999.f ;

scan\_rate:meta\_group = "instrument\_parameters" ;

short DBZ(n\_points) ;

DBZ:long\_name = "radar\_reflectivity" ;

DBZ:standard\_name = "equivalent\_reflectivity\_factor" ;

DBZ:units = "dBZ" ;

DBZ:sampling\_ratio = 1.f ;

DBZ:\_FillValue = -32768s ;

DBZ:scale\_factor = 0.001411481f ;

DBZ:add\_offset = 17.25f ;

DBZ:grid\_mapping = "grid\_mapping" ;

DBZ:coordinates = "time range" ;

short VEL(n\_points) ;

VEL:long\_name = "radial\_velocity" ;

VEL:standard\_name = "radial\_velocity\_of\_scatterers\_away\_from\_instrument" ;

VEL:units = "m/s" ;

VEL:sampling\_ratio = 1.f ;

VEL:\_FillValue = -32768s ;

VEL:scale\_factor = 0.0009842219f ;

VEL:add\_offset = -0.25f ;

VEL:grid\_mapping = "grid\_mapping" ;

VEL:coordinates = "time range" ;

short WIDTH(n\_points) ;

WIDTH:long\_name = "spectrum\_width" ;

WIDTH:standard\_name = "doppler\_spectrum\_width" ;

WIDTH:units = "m/s" ;

WIDTH:sampling\_ratio = 1.f ;

WIDTH:\_FillValue = -32768s ;

WIDTH:scale\_factor = 0.0002899258f ;

WIDTH:add\_offset = 9.5f ;

WIDTH:grid\_mapping = "grid\_mapping" ;

WIDTH:coordinates = "time range" ;

short ZDR(n\_points) ;

ZDR:long\_name = "differential\_reflectivity" ;

ZDR:standard\_name = "log\_differential\_reflectivity\_hv" ;

ZDR:units = "dB" ;

ZDR:sampling\_ratio = 1.f ;

ZDR:\_FillValue = -32768s ;

ZDR:scale\_factor = 0.000241287f ;

ZDR:add\_offset = 0.03125f ;

ZDR:grid\_mapping = "grid\_mapping" ;

ZDR:coordinates = "time range" ;

short PHIDP(n\_points) ;

PHIDP:long\_name = "differential\_phase" ;

PHIDP:standard\_name = "differential\_phase\_hv" ;

PHIDP:units = "deg" ;

PHIDP:sampling\_ratio = 1.f ;

PHIDP:\_FillValue = -32768s ;

PHIDP:scale\_factor = 0.3525968f ;

PHIDP:add\_offset = 11553.19f ;

PHIDP:grid\_mapping = "grid\_mapping" ;

PHIDP:coordinates = "time range" ;

short RHOHV(n\_points) ;

RHOHV:long\_name = "cross\_correlation" ;

RHOHV:standard\_name = "cross\_correlation\_ratio\_hv" ;

RHOHV:units = "" ;

RHOHV:sampling\_ratio = 1.f ;

RHOHV:\_FillValue = -32768s ;

RHOHV:scale\_factor = 1.286864e-05f ;

RHOHV:add\_offset = 0.63f ;

RHOHV:grid\_mapping = "grid\_mapping" ;

RHOHV:coordinates = "time range" ;

// global attributes:

:Conventions = "CF-1.6" ;

:Sub\_conventions = "CF-Radial instrument\_parameters radar\_parameters radar\_calibration" ;

:version = "CF-Radial-1.3" ;

:title = "" ;

:institution = "" ;

:references = "" ;

:source = "ARCHIVE 2 data" ;

:history = "" ;

:comment = "" ;

:author = "" ;

:original\_format = "NEXRAD" ;

:driver = "RadxConvert(NCAR)" ;

:created = "2016/05/22 22:50:15.274" ;

:start\_datetime = "2015-06-26T12:04:15Z" ;

:time\_coverage\_start = "2015-06-26T12:04:15Z" ;

:end\_datetime = "2015-06-26T12:08:31Z" ;

:time\_coverage\_end = "2015-06-26T12:08:31Z" ;

:instrument\_name = "KDDC" ;

:site\_name = "" ;

:scan\_name = "Surveillance" ;

:scan\_id = 212 ;

:platform\_is\_mobile = "false" ;

:n\_gates\_vary = "true" ;

:ray\_times\_increase = "true" ;

}

1. The rotation angle, *θ*’, defined inprevious 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 -Ya-axis) that has been the convention for airborne tail Doppler radars. *θ*’=0° points to +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. [↑](#footnote-ref-1)