NCAR-CSM NetCDF Conventions

Revisions

10 November 1996, First Draft

14 November 1997, Version 1.0

This document contains a description of the netCDF conventions adopted for the output datasets of the NCAR Climate Systems Model. The convention is designed for the representation of gridded geophysical data.

1. Overview

1.1 Convention Purpose and Philosophy

The netCDF interface enables but does not require the creation of *self-describing* datasets. The purpose of this convention is to require conforming datasets to contain sufficient metadata that they are self-describing in the sense that each variable in the file has an associated description of what it represents, including physical units if appropriate, and that each value can be located in space and time. Also required is a higher level description pertaining to the variables as a group that gives the source of the data and a history of manipulations that may have been performed on it.

An important potential benefit of a convention is that it can enable software tools that display data and perform operations on specified subsets of the data to do their tasks with minimal user intervention. It is possible using netCDF to provide the metadata describing how a field is located in time and space in many different ways that a human reader would immediately recognize as equivalent. The purpose in restricting how the metadata is represented is to make it practical to write software that allows a machine to be able to figure out what data goes where without need for human intervention. The main restrictions we impose are to require the use of coordinate variables and several variable attributes. No restrictions are placed on the order of a variable's dimensions.

1.2 Summary

The **`long_name`** attribute is required for all variables.

The use of coordinate variables is required for all dimensions that correspond to 1D space or time coordinates. Since coordinate variables as defined in the NetCDF User's Guide are only adequate for describing rectilinear grids, we define a **`coordinates`** attribute below to deal with data on irregular grids.

The **`units`** attribute is required for all variables used to describe coordinates, and for all other variables for which it is appropriate. The values of the units attributes are character arrays that are recognized by the UNIDATA's udunits package whenever possible. Exceptions are that the

units "degree(s)" are not allowed. This convention describes some new values of the units attribute to be used in certain circumstances in which no udunits conformable value exists.

The global attributes **`title`**, **`source`**, and **`history`** are required to provide a minimal description of what the data is, where it came from, and what's been done to it.

The **`Conventions`** attribute is required to identify the location of this document.

In addition to the required features listed above, several conventions have been established to help in the description of gridded data that does not represent pointwise values of the field, but rather represents some characteristic of the field over intervals in space and/or time. A common example of this is time averaged data.

1.3 Relationship to the COARDS Convention

This convention has been significantly influenced by the COARDS convention. The differences that may result in files that are not COARDS compliant are the following:

1) COARDS restricts the order of a variable's dimensions. Because of I/O performance considerations it may not be desirable for all CSM component models to output their data in conformance with the COARDS requirement. This convention places no restrictions on the order of dimensions.

2) COARDS addresses the issue of dimensionless vertical coordinates, but does not provide a convention that is sufficiently general for our purpose. We would like to use the **`units`** attribute to be able to locate the vertical coordinate in space analogously to how the COARDS conventions for latitude and longitude allow one to unambiguously locate points in the horizontal plane. In the case of a dimensionless coordinate some rule is required to convert to one with dimensions. Since there are many possible rules depending on the particular coordinate, we require the value of **`units`** to be specific to each coordinate system. Thus we reject "level" and "layer" as being too vague. "sigma_level" describes a particular type of dimensionless coordinate, but not the one used in the atmospheric component of CSM. Below we provide additional conventions for dimensionless vertical coordinate **`units`** attributes.

3) COARDS restricts the names allowed by the netCDF interface by not including the use of the hyphen character, and by recommending that names be case insensitive. We agree that it is better not to use the hyphen character, but feel that case sensitive names should be allowed as is assumed by the netCDF interface.

The NCAR-CSM convention is more general than COARDS. It is possible to write netCDF files that are compliant with both NCAR-CSM and COARDS, but NCAR-CSM conventions don't restrict files to being COARDS compliant.

2. Conventions for required attributes

2.1 Long_name attribute

The **long_name** attribute is required for all variables. The value is a character array that contains a long descriptive name.

2.2 Units attribute

The **units** attribute is required for all variables that describe coordinates, and for all other variables whose values represent a dimensional quantity. The values of the units attributes are character arrays that are recognized by the UNIDATA's udunits package whenever possible. The units `degree(s)` are not allowed (see sections 2.3.4 and 2.3.5 for appropriate units of longitude and latitude). Udunits doesn't have specifications for dimensionless quantities like percent, ppb, and fraction. We recommend that the **units** attribute still be used, but have not standardized any dimensionless specifiers except for those to indicate vertical coordinates as defined in section 2.3.3 below.

2.3 Coordinates

The term *coordinate variable* is defined in the netCDF User's Guide to be a 1D array that is used to describe a coordinate in a rectilinear grid. This convention uses the term coordinate variable in that sense. When non-rectilinear grids are used then coordinate information must be represented using higher dimension arrays which do not fit the coordinate variable model. To handle this case the **coordinates** attribute is introduced below. The term *coordinate* will be used generically to refer to either the 1D or multidimensional case.

The use of coordinate variables is required for all space or time coordinates that can be represented as 1D arrays.

The values of a coordinate variable must be either strictly increasing or strictly decreasing. The values need not be evenly spaced. Missing values are not permitted. A multidimensional coordinate may use the **_FillValue** or **missing_value** attributes to indicate grid points that are not used, e.g., in a *reduced grid* where the number of longitudes on a latitude circle decreases as one moves toward the poles.

The **units** attribute is required for all space or time coordinates. The reason for this requirement is that the **units** attributes along with the coordinate values provides a means of precisely locating data values. In the rectilinear case the orientation of a coordinate axis can be determined from the **units** attribute alone.

2.3.1 Time

We require the units for a time coordinate to be parsable by Unidata's udunit library as in the following modified excerpt from the udunits documentation:

```
      days since 1992-10-8 15:15:42.5 -6:00

indicates days since October 8th, 1992 at 3 hours, 15 minutes and 42.5
seconds in the afternoon in the time zone which is six hours to the west of
```

```
Coordinated Universal Time (i.e. Mountain Daylight Time).  The time zone
specification can also be written without a colon using one or two-digits
(indicating hours) or three or four digits (indicating hours and minutes).
```

The acceptable units for time are listed in the file udunits.dat. The most commonly used of these strings (and their abbreviations) includes `day (d), hour (hr, h), minute (min)` and `second (sec, s)`. Plural forms are also acceptable. The date string may include date alone; date and time; or date, time, and time zone. The date string is required.

We recommend that the unit `year` be used with caution. The udunits package defines a `year` to be exactly 365.242198781 days (the interval between 2 successive passages of the sun through vernal equinox). It is not a calendar year. Udunits includes the following definitions for years: a `common_year` is 365 days, a `leap_year` is 366 days, a `Julian_year` is 365.25 days, and a `Gregorian_year` is 365.2425 days.

The calendar calculations done by the udunits package use a mixed Gregorian/Julian calendar, i.e., dates prior to 1582-10-15 are assumed to use the Julian calendar. Time coordinates that use other calendars are thus not able to make use of the udunits library for this purpose. However, it is still required to use the time unit format described above as this contains all the information required to make calendar calculations once the calendar has been specified. We describe a **`calendar`** attribute for the time coordinate variable below that may be used for this purpose.

Coordinate variables representing climatological time (an axis of 12 months, 4 seasons, etc. that is located in no particular year) should be encoded like other time axes but with the added restriction that they be encoded to begin in the year 0000. For example,

```
days since 0000-06-15 00:00 0
```

indicates days since beginning of the model run starting Jun 15, 0Z.

Time coordinates used for paleoclimate research my involve calendars based on different orbital parameters from those of the present. In this case additional information besides that contained in the **`calendar`** attribute may be required. A description of the orbital parameters may be included by using the attribute **`orbital_parameters`** described below. A complete description of the calendar may be included using the global attribute **`paleo_calendar`**.

2.3.2 Dimensional Vertical Coordinates

The acceptable units for dimensional vertical (depth or height) coordinate variables are:

1) Units of pressure as listed in the file udunits.dat. For vertical axes the most commonly used of these include include `bar, millibar (mbar), decibar (dbar), atmosphere (atm), pascal (Pa)`, and `hPa`.

2) Units of length as listed in the file udunits.dat. For vertical axes the most commonly used of these include `meter (metre, m), centimeter (cm), decimeter (dm), kilometer (km),` and `feet (ft)`.

3) Other units listed in the file udunits.dat that may under certain circumstances reference vertical position such as units of density or temperature.

Plural forms are also acceptable.

The direction of positive (i.e., the direction in which the coordinate is increasing), whether up or down, cannot in all cases be inferred from the units. The direction of positive is useful for applications displaying the data. For this reason the attribute **`positive`** as defined in the COARDS convention is required if the vertical axis units are not a valid unit of pressure. Otherwise its inclusion is optional. The **`positive`** attribute may have the value `up` or `down` (case insensitive).

For example, if an oceanographic netCDF file encodes the depth of the surface as 0 and the depth of 1000 meters as 1000 then the axis would use attributes as follows:

```
        axis_name:units="meters";
        axis_name:positive="down";
```

If, on the other hand, the depth of 1000 meters were represented as -1000 then the value of the **`positive`** attribute would have been `up`. If the **`units`** attribute value is a valid pressure unit the default value of the **`positive`** attribute is `down`.

Notice that a vertical coordinate variable will be identifiable either by having units of pressure or by the presence of the **`positive`** attribute with a value of `up` or `down` (case insensitive).

2.3.3 Dimensionless Vertical Coordinates

For dimensionless vertical coordinates we introduce conventions that are designed to facilitate the calculation of the corresponding dimensional coordinates that are required to locate the data spatially. For each of the dimensionless vertical coordinates described below an example of the convention in CDL notation is given for a dimension with the name `z`.

1) Hybrid Sigma Pressure.

```
float z(z) ;
       z:long_name = "hybrid level at layer midpoints" ;
       z:units = "hybrid_sigma_pressure" ;
       z:positive = "down" ;
       z:A_var = "hyam" ;
       z:B_var = "hybm" ;
       z:P0_var = "pref" ;
       z:PS_var = "psurf" ;
```

A **`units`** attribute of `hybrid_sigma_pressure` means that the pressure at gridpoint (x(i),y(j),z(k)) is given by p(i,j,k) = A(k)*PO + B(k)*PS(i,j) where the variable names for A, B, P0, and PS are given by the attributes A_var, B_var, P0_var, and PS_var respectively.

2) Sigma.

```
float z(z) ;
       z:long_name = "sigma level at layer midpoints" ;
       z:units = "sigma_level" ;
       z:positive = "down" ;
       z:B_var = "z" ;
       z:P0_var = "ptop" ;
       z:PS_var = "psurf" ;
```

A **units** attribute of `sigma_level` means that the pressure at gridpoint (x(i),y(j),z(k)) is given by p(i,j,k) = P0 + B(k)*(PS(i,j)-P0) where the variable names for B, P0, and PS are given by the attributes B_var, P0_var, and PS_var respectively.

We recommend that the rule for converting from a dimensionless to a dimensional coordinate be included as a global attribute whose name is same as that of the units specifier. Here is an example for the CCM3 vertical coordinate:

```
// global attributes:
                :hybrid_sigma_pressure = "\n",
    "Pressure at a grid point (lon(i),lat(j),lev(k)) is computed    \n",
    "using the formula:                                            \n",
    "           p(i,j,k) = A(k)*PO + B(k)*PS(i,j)                  \n",
    "where A, B, PO, and PS are contained in the variables whose    \n",
    "names are given by the attributes of the vertical coordinate   \n",
    "variable A_var, B_var, P0_var, and PS_var respectively.        \n",
    "" ;
```

2.3.4 Latitude

The recommended unit of latitude is `degrees_north`. Also acceptable are `degree_north`, `degree_N`, and `degrees_N`.

2.3.5 Longitude

The recommended unit of longitude is `degrees_east` (eastward positive). Also acceptable are `degree_east`, degree_E, and `degrees_E`. The unit `degrees_west` (westward positive) is not recommended because it implies a negative conversion factor from `degrees_east`.

Longitudes may be represented modulo 360. Thus, for example, -180, 180, and 540 are all valid representations of the International Dateline and 0 and 360 are both valid representations of the Prime Meridian. Note, however, that the sequence of numerical longitude values stored in the netCDF file must be monotonic in a non-modulo sense.

2.3.6 Non-rectilinear coordinates

When multidimensional variables are required to describe a coordinate, these variables are identified as coordinates by use of the **coordinates** attribute. The value of the attribute is a string containing the names of variables that describe the coordinate system. There must be at

least as many coordinates as there are dimensions for the field variable. There may be more however as in the case of describing a field defined along a trajectory through space.

Examples in CDL notation:

1) Latitude and longitude coordinates both requiring 2D variables:

```
dimensions:
        nlon = 128 ;
        nlat = 64 ;
        nlev = 18 ;
variables:
        float lon(nlat,nlon) ;
                lon:long_name = "longitude" ;
                lon:units = "degrees_east" ;
        float lat(nlat,nlon) ;
                lat:long_name = "latitude" ;
                lat:units = "degrees_north" ;
        float z(nlev) ;
                z:long_name = "level" ;
                z:units = "mbar" ;
        float T(nlev,nlat,nlon) ;
                T:long_name = "temperature" ;
                T:units = "K" ;
                T:coordinates = "lon lat z" ;
```

The coordinates attribute of T tells you that grid point (i,j,k) is located at (lon(i,j),lat(i,j),z(k)). The location of this position in physical space is determined by the interpretations of the coordinates themselves. Because of this there is no restriction on the order in which the coordinate names appear in the coordinate attribute string. Notice that while the vertical coordinate could have been described by using a coordinate variable, this is not required when the coordinate attribute is used. Neither is it disallowed.

2) Longitude coordinate requiring 2D variable and missing values:

```
dimensions:
        nlon = 128 ;
        lat = 64 ;
variables:
        float lon(lat,nlon) ;
                lon:long_name = "longitude" ;
                lon:units = "degrees_east" ;
                lon:_FillValue = -999.f ;
        float lat(lat) ;
                lat:long_name = "latitude" ;
                lat:units = "degrees_north" ;
        float PS(lat,nlon) ;
                PS:long_name = "surface pressure" ;
                PS:units = "Pa" ;
                PS:coordinates = "lon lat" ;
                PS:_FillValue = 1.e36f
```

The 2D longitude coordinate indicates that the longitude values depend on the latitude, and there are missing values. This is the situation when using a grid in which the number of longitude points on a latitude circle decreases going towards the poles.

3) Trajectories:

```
dimensions:
        time = 1000 ;
variables:
        float lon(time) ;
                lon:long_name = "longitude" ;
                lon:units = "degrees_east" ;
        float lat(time) ;
                lat:long_name = "latitude" ;
                lat:units = "degrees_north" ;
        float z(time) ;
                z:long_name = "level" ;
                z:units = "km" ;
                z:positive = "up" ;
         double time(time) ;
                 time:long_name = "time" ;
                 time:units = "days since 1970-01-01 00:00:00" ;
                time:calendar = "gregorian" ;
        float O3(time) ;
                O3:long_name = "ozone concentration" ;
                O3:units = "ppbv" ;
                O3:coordinates = "lon lat z time" ;
```

2.4 Global Attributes

The required global attributes are intended to provide information about where the data came from and what has been done to it. This information is mainly for the benefit of human readers. These attributes are all character arrays. For readability in ncdump outputs it is recommended to embed newline characters into the arrays to break them into lines.

`title`
: A succinct description of what is in the data set.

`source`
: Source of the data. For CSM data this could be the specific model component that produced the data along with a description of modifications made to the standard model component

`history`
: Contains a line for each invocation of a program and arguments that were used to derive the file. Well-behaved generic netCDF filters will automatically append their name and the parameters with which they were invoked to the global history attribute of an input netCDF file. We recommend that each line begin with a timestamp indicating the time and date that the program was invoked.

`Conventions`

A conforming dataset should use the value `NCAR-CSM` which indicates that the conventions followed by the dataset may be found on the host machine ftp.unidata.ucar.edu in the directory pub/netcdf/Conventions/NCAR-CSM/.

3. Conventions for optional attributes

3.1 Calendar / Orbital Parameters

In order to calculate a new date and time given a base date, base time and a time increment one must know what calendar to use. For this purpose we recommend that the attribute **`calendar`** be assigned to time coordinate (or be global) whenever its **`units`** are of the form `time units since base date, base time`. The values currently defined for **`calendar`** are:

`gregorian`
: Modern calendar. (Default)

`noleap`
: Modern calendar without leap years, i.e., all years are 365 days long.

`julian`
: Julian calendar.

`n kyr B.P.`
: A generic designation for a paleoclimate calendar valid n thousand years before present where n should be replaced by the appropriate value. A precise definition of the calendar may be included by using the global attribute **`paleo_calendar`**.

In paleoclimate research the calendar may be described with reference to an orbit that is different from that of present day. This should be described by using an attribute of the time coordinate (or global) called **`orbital_parameters`** to give a list of the names of variables that contain the eccentricity, obliquity, and perihelion data. The method of attaching calendar dates to the orbit can be described using the global attribute **`paleo_calendar`**, for example:

```
// global attributes:
                :paleo_calendar = "\n",
    "First day and length of angular months for 126 kyr B.P. based  \n",
    "on a 365 day year with vernal equinox fixed to March 21 (the   \n",
    "Day/Month values refer to the present calendar):               \n",
    "          Day/Month  Length                                    \n",
    "January     25/12      34                                      \n",
    "February    28/01      31                                      \n",
    "March       28/02      32                                      \n",
    "April       01/04      30                                      \n",
    "May         01/05      29                                      \n",
    "June        30/05      27                                      \n",
    "July        26/06      28                                      \n",
    "August      24/07      28                                      \n",
    "September   21/08      28                                      \n",
    "October     18/09      32                                      \n",
    "November    20/10      32                                      \n",
    "December    21/11      34                                      \n",
    "" ;
```

3.2 Representing values on intervals

It is often the case that data on a grid does not represent the point values of some field variable but instead represents some characteristic of the field (i.e., the result of some mathematical operation performed on the field values) over intervals of space and/or time. Typically this might be a weighted average or perhaps the minimum or maximum values over the intervals. Hence we require methods to describe both the characteristic of the field over intervals and what the intervals are that correspond to the grid points.

To represent the characteristic of the field over intervals we introduce attributes of the form **coord_op**, where the string **coord** should be replaced by the actual name of the coordinate, for example **lat_op**, or **time_op**. The values of these attributes are character arrays that describe the operation that has been performed on the corresponding coordinate. The currently defined values are:

```
"point"   - value is at a point (default, does not need to be specified)
"minimum" - minimum of values over an interval
"maximum" - maximum of values over an interval
"sum"     - sum of values over an interval
"average" - average of values over an interval
"rms"     - root mean square of values over an interval
"range"   - difference between maximum and minimum values over an interval
```

coord_op attributes may be applied to individual variables or may be global which implies that the operation has been applied to all variables which use the indicated coordinate. If **coord_op** is a global attribute, it may still be applied to a variable to override the value of the global attribute.

To represent the intervals we add the attribute **bounds** to the appropriate coordinate. The value of **bounds** is the name of the variable that contains the interval boundaries. The the case where the grid is defined by coordinate variables, and where the intervals are contiguous this can be a variable whose dimension size is one larger than the dimension size of the corresponding coordinate variable. The relationship between a coordinate variable, say x, and the variable that contains the interval bounds, say x_bound, is that the value x(i) is contained in the interval with boundaries x_bound(i) and x_bound(i+1). In the more general case (still 1D) where the intervals may be disjoint or possibly overlapping then the variable containing the bounds is 2D with the 2nd dimension (in C notation) being the same size as the dimension of the corresponding coordinate variable. In this case the relationship is that the value x(i) is contained in the interval with boundaries x_bound(0,i) and x_bound(1,i).

Examples in CDL notation:

1) Time averaged variable, contiguous intervals:

```
dimensions:
        time_bound = 4 ;
        time = 3 ;
variables:
        double time(time) ;
                time:long_name = "time" ;
                time:units = "days since 1970-01-01 00:00:00" ;
```

```
                time:calendar = "gregorian" ;
                time:bounds = "time_bound" ;
        double time_bound(time_bound) ;
                 time_bound:long_name = "time interval boundaries" ;
                 time_bound:units = "days since 1970-01-01 00:00:00" ;
        float gaTS(time) ;
                gaTS:long_name = "global average surface temperature" ;
                gaTS:units = "K" ;
                gaTS:time_op = "average" ;
data:
 time = .25, .5, .75 ;
 time_bound = 0., .25, .5, .75 ;
```

The variable **gaTS** represents 6 hour time averages. The first time sample corresponds to a time average starting at 1970-01-01 0Z and ending at 1970-01-01 6Z. The values of the time coordinate are arbitrarily set to the end of the corresponding averaging interval.

2) Time averaged variable, disjoint intervals:

```
dimensions:
        time = 3 ;
variables:
        double time(time) ;
                 time:long_name = "time" ;
                 time:units = "days since 1970-01-01 00:00:00" ;
                time:calendar = "gregorian" ;
                time:bounds = "time_bound" ;
        double time_bound(2,time) ;
                 time_bound:long_name = "time interval boundaries" ;
                 time_bound:units = "days since 1970-01-01 00:00:00" ;
        float gaTS(time) ;
                gaTS:long_name = "global average surface temperature" ;
                gaTS:units = "K" ;
                gaTS:time_op = "average" ;
data:
 time = 31., 396., 761. ;
 time_bound(0,*) = 0., 365., 730. ;
 time_bound(1,*) = 31., 396., 761. ;
```

The variable **gaTS** represents monthly time averages for January 1970-1972.

3.3) **non-ordinal** coordinates

It is useful to adopt a convention for certain types of dimensions that do not map into netCDF coordinate variables, but that aren't simply generic ordinal dimensions. For example, we save certain quantities associated with particular islands or ocean basins. Here the coordinate value is the character string containing the island or basin name which cannot be used as a netCDF coordinate. One can use character string variables that have the same name as the dimension with **_label** attached to associate these values with the corresponding dimensions. e.g,.

```
dimensions:
        time = UNLIMITED ; // (12 currently)
        z_t = 45 ;
```

```
        nchar = 132 ;
        islands = 8 ;
        basins = 8 ;
variables:
        double time(time) ;
                time:long_name = "time" ;
                time:units = "days since 0000-00-00 00:00:00" ;
        float z_t(z_t) ;
                z_t:long_name = "Depth (T grid)" ;
                z_t:units = "centimeters" ;
                z_t:positive = "down" ;
        char islands_label(islands, nchar) ;
                islands_label:long_name = "Islands" ;
        char basins_label(basins, nchar) ;
                basins_label:long_name = "Ocean Basins" ;
        float T_horz(time, basins, z_t) ;
                T_horz:long_name = "Horizontal Average Potential Temperature"
;
                T_horz:units = "celsius" ;
                T_horz:time_rep = "instantaneous" ;
        float pisle(time, islands) ;
                pisle:long_name = "Island Streamfunction" ;
                pisle:units = "centimeters^3/second" ;
                pisle:time_rep = "instantaneous" ;
```

3.4) Flux direction

We encourage making the direction of all flux quantities explicit by the use of the variable attribute **`flux_direction`**. The values should be either **`up`** or **`down`**.

References

The following references are all accessable from the UNIDATA home page: http://www.unidata.ucar.edu/.

NetCDF User's Guide, Version 3. See section 2.3.1 for a description of coordinate variables and section 8.1 for a description of the long_name, units, title, history, and Conventions attributes.

COARDS Conventions.

UDUNITS library.