Metadata recommendations for encoding netCDF products based on CF convention

Authors

- Antonio S. Cofiño, University of Cantabria, Spain
- Products Team, Production Section, ECMWF.

Status

This document has been produced as recommendation for encoding netCDF products from UERRA, CERA-20C and Copernicus C3S projects. This document does not specify an standard or data management plan.

Abstract

This document provides some recommendations and examples for the encoding of structural metadata in a form suitable for archiving. The main encoding format it is the netCDF file format. These metadata recommendations are based on the CF-1.X Conventions, but some modifications and extensions are applied. The main objective of this recommendations is to provide input to the different data management plans for C3S projects (and others).

Revision history

Version	Date	Comments
V1.0	31/01/2017	First version. English grammar and syntax reviewed.
V2.0	10/03/2017	Incorporating C3S General Assembly comments, suggestions and fixing issues.
V3.0	06/04/2017	Improved example on aggregation section: cell_methods for monthly averages (on forecast_reference_time) of daily maximum (on forecast_period)
V4.0	19/04/2017	Clarification from Seasonal DMP: missing_value and _FillValue attributes
V5.0	10/05/2017	Revision history, status and abstract sections added. Some syntax errors fixed
V6.0	26/05/2017	Change main title and more general purpose introduction section.

Known issues

- Update NEMO example (7.3) with updated grid (Sebastien)
- UERRA examples on surface and isobaric levels (model levels?)

- C3S seasonal example update surface and
- naming variables and attributtes:
    - http://www.unidata.ucar.edu/software/netcdf/docs/netcdf_data_set_components.html#object_name The names of dimensions, variables and attributes (and, in netCDF-4 files, groups, user-defined types, compound member names, and enumeration symbols) consist of arbitrary sequences of alphanumeric characters, underscore '_', period '.', plus '+', hyphen '-', or at sign '@', but beginning with an alphanumeric character or underscore. However names commencing with underscore are reserved for system use.
    - http://cfconventions.org/cf-conventions/v1.6.0/cf-conventions.html#_naming_conventions Variable, dimension and attribute names should begin with a letter and be composed of letters, digits, and underscores. Note that this is in conformance with the COARDS conventions, but is more restrictive than the netCDF interface which allows use of the hyphen character. The netCDF interface also allows leading underscores in names, but the NUG states that this is reserved for system use.

Table of Contents

1 Introduction

The objective of this document is to provide some recommendations (and examples) for the encoding of metadata and data in a form suitable for archiving.

The encoding reference used is the netCDF classic data model, but extension to other encoding formats should be possible. The aim is to be explicit (as possible), to provide: values for file/record specific attributes (and not for overall collections), richer metadata and conventions.

The intention is to be minimalist in approach to allow downstream data re-use beyond the original intent, product development, scientific quality control and provision of long-term preservation. This means that this recommendation is not intended to provide metadata to specific project, experiment or simulation, like attributes for data discovery, or special characteristics.

This recommendation is full based on CF-1.6 Convention and many parts are excerpts from the Convention Document and the Standard Name Table. This document is a distillation of encoding standard described by CF Convention.

1.1 Data models, standards and conventions

The data models, conventions are standards (including *de-facto* ones) which are considered are the following:

1. NetCDF4 Data Model
    a. Classic
    b. Enhanced
2. CF Convention[1]
    a. Standard Name Table[2]
    b. The CF Model Data[3]
3. Unidata's Common Data Model[4]
    a. Scientific Feature Types[5]
4. Attribute Convention for Data Discovery[6]

Reference is also made to the CF-Discussion email list and the CF Metadata Trac, where some practical clarifications and recommendations are also made

2 Data Fields

A data field (DF) is defined as a combination of several data variables. In general, a data field may span data variables in more than one file object -for instance, from different ranges of a time coordinate. Rules for aggregating data variables from one or several files objects into a single data field are needed but are not defined by CF[7].

1http://cfconventions.org
2http://cfconventions.org/standard-names.html
3http://www.met.reading.ac.uk/~david/cfdm_0.8.html
4http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/CDM/
5http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/CDM/CDMfeatures.doc
6http://wiki.esipfed.org/index.php/Attribute_Convention_for_Data_Discovery
7CF aggregation rules proposal, http://cf-trac.llnl.gov/trac/ticket/78

The important assumption is that each DF is independent. However, data stored in netCDF files are often not independent, because coordinate variables are shared between them. This sharing is a means of saving disk space, and any software should be able to alter any DF without altering any other DF. For example, if a given coordinate of a DF is altered this should not affect other DF's in the same file. For example, daily near-surface minimum temperature could share longitude and latitude coordinates variables with 6 hourly instantaneous temperature at 500mb pressure level, but different temporal and vertical coordinate variables should be used. A test on the equality, and/or equivalence, of coordinates between different DF's should be made when attempting to merge DF from different files or records.

A DF *may* have:

1. 0..N **domain indexes:** each domain coordinate has a size (an integer value greater than zero)**.**
2. 1 **data array** whose shape is been determined by the domain indexes. All elements on the data array must be of the same data type (numeric, char or string).
3. A collection of **domain coordinates**: A domain coordinate construct indicates the physical meaning and locations of the cells for a unique **domain indexes** of the field.
4. A collection of **auxiliary coordinates**: An auxiliary coordinate provides auxiliary information for interpreting the cells of an ordered list of one or more **domain coordinate** of the field.
5. A collection of **cell-measures**: A cell measure provides information about the size, shape or location of the cells (n-dimensional in general) defined by an ordered list of one or more domain coordinates of the DF.
6. An ordered collection of **cell-methods**: A cell methods describes how the data values represent variation of the quantity within cells. The methods are not necessary commutative, therefore it is an ordered list of methods.
7. A **Coordinate reference systems**: A coordinate reference system relates the field's coordinate values to locations in a earth reference frame.
8. Other **properties** which represents metadata about the DF. Not all attributes in a netCDF file are properties in this sense. Some of these can be global attributes in a netCDF file. It is assumed that global attribute is also an attribute of every data variable, although it is superseded if the data variable has its own attribute with an identical name.
9. Ancillary fields which are used to identify fields which provide additional metadata (i.e. quality of the data).

The **domain axes, domain coordinates, auxiliary coordinates, cell measures, and cell method describe the domain** in which the DF resides.

3 Coordinate ~~reference~~ systems

Because identification of a coordinate type by its units alone is complicated, two optional methods can be used to provide a direct identification. To identify generic spatial coordinates the use of the attribute `axis` may be attached to a coordinate variable and given one of the values **X**, **Y**, **Z** or **T**. Alternatively the `standard_name` attribute may be used for direct identification.

The values of a coordinate variable indicate the locations of the cells, and the locations of the boundaries between cells are indicated by `bounds` property attached to coordinate variable.

3.1 Horizontal coordinate ~~reference~~ systems

When the coordinate variables for a horizontal grid are not true *longitude* and true *latitude*, it is required that the true latitude and true longitude coordinates be supplied via the `coordinates` attribute. If in addition it is desired to describe the mapping between the given coordinate variables and the true latitude and true longitude coordinates, the property `grid_mapping` may be attached to the data variable.

3.1.1 Regular longitude and latitude

```
netcdf regular_latitude_longitude_grid {
  //global attributes:
    :Conventions = "CF-1.6";
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
  variables:
    double mslp(latitude, longitude) ;
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa" ;
      mslp:grid_mapping = "hcrs" ;
      mslp:coordinates = "latitude longitude" ;
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
}
```

3.1.2 Regular projection grids

```
netcdf regular_projection_grid {
  //global attributes:
    :Conventions = "CF-1.6";
  dimensions:
    x = 565;
    y = 565;
  variables:
    double mslp(y, x);
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa";
      mslp:coordinates = "y x latitude longitude";
      mslp:grid_mapping = "hcrs";
    double x(x);
```

```
      x:standard_name = "projection_x_coordinate";
      x:units = "km";
      x:axis = "X" ;
    double y(y);
      y:standard_name = "projection_y_coordinate";
      y:units = "km";
      y:axis = "Y" ;
    double latitude(y,x) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
    double longitude(y,x) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
    char hcrs;
      hcrs:grid_mapping_name ="lambert_conformal_conic";
      hcrs:latitude_of_projection_origin = 48.0;
      hcrs:longitude_of_central_meridian = 8.0;
      hcrs:standard_parallel = 48.0;
      hcrs:earth_radius = 6371229.0;
}
```

3.1.3 Regular non-projection grids: NEMO/ORCA case

```
netcdf regular_non-projection_grid {
  //global attributes:
    :Conventions = "CF-1.6";
  dimensions:
    x = 292;
    y = 362;
  variables:
    float sit(y, x) ;
      sit:standard_name = "sea_ice_thickness" ;
      sit:units = "m" ;
      sit:coordinates = "y x latitude longitude" ;
    double x(x);
      x:long_name = "i-index of mesh grid";
      x:units = "1";
      x:axis = "X" ;
    double y(y) ;
      y:long_name = "j-index of mesh grid" ;
      y:units = "1" ;
      y:axis = "Y" ;
    double latitude(y, x) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
    double longitude(y, x) ;
```

```
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
}
```

3.2 Vertical coordinate ~~reference~~ systems

Variables representing dimensional height or depth axes must always explicitly include the `units` attribute; there is no default value for this attribute. If the `units` attribute value is a valid pressure unit the default value of the `positive` attribute is `down`.

A vertical coordinate will be identifiable by:

- units of pressure; and/or
- the presence of the `positive` attribute with a value of `up` or `down` (case insensitive); and/or
- by providing the `standard_name` attribute with an appropriate value; and/or
- the `axis` attribute with the value `Z`.

3.2.1 Near-surface fields

```
netcdf near-surface {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
  variables:
    double tas(latitude, longitude) ;
      tas:standard_name = "air_temperature" ;
      tas:units = "K";
      tas:grid_mapping = "hcrs" ;
      tas:coordinates = "height latitude longitude" ;
    double height ;
      height:standard_name = "height";
      height:units = "m";
      height:positive = "up";
      height:axis = "Z";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    height = 2. ;
}
```

3.2.2 Isobaric levels

```
netcdf isobaric_levels {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
    plev = 11 ;
  variables:
    double tas(plev, latitude, longitude) ;
      tas:standard_name = "air_temperature" ;
      tas:units = "K";
      tas:grid_mapping = "hcrs" ;
      tas:coordinates = "plev latitude longitude" ;
   double plev(plev) ;
      plev:standard_name = "air_pressure" ;
      plev:units = "Pa" ;
      plev:positive = "down" ;
      plev:axis = "Z" ;
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    plev = 92500,85000,70000,50000,40000,30000,20000,10000,5000, 3000, 1000 ;
}
```

3.2.3 Depth levels

```
netcdf depth_levels {
  //global attributes:
    :Conventions = "CF-1.6";
  dimensions:
    y = 362;
    x = 292;
    depth = 42;
  variables:
    float so(depth, y, x) ;
      so:standard_name = "sea_water_salinity" ;
      so:units = "0.001" ;
      so:long_name = "Practical Salinity" ;
      so:coordinates = "depth y x latitude longitude" ;
    float depth(depth) ;
      depth:standard_name = "depth_below_geoid" ;
      depth:units = "m" ;
      depth:positive = "down" ;
      depth:axis = "Z" ;
    double y(y) ;
      y:long_name = "j-index of mesh grid" ;
      y:units = "1" ;
      y:axis = "Y" ;
    double x(x);
      x:long_name = "i-index of mesh grid";
      x:units = "1";
      x:axis = "X" ;
    double latitude(y, x) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
    double longitude(y, x) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
}
```

3.3 Time coordinate ~~reference~~ systems

Coordinates representing time must always explicitly include the `units` attribute; there is no default value. The `units` attribute takes a string value formatted as per the recommendations in the Udunits package[8]. The following example of units has been excerpted from the CF-1.6 Standard Document:

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

```
indicates seconds 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
```

8 http://www.unidata.ucar.edu/packages/udunits/

```
can also be written without a colon using one or  two-digits
(indicating hours) or three or four digits (indicating hours
and minutes).
```

If the time zone is omitted the default is UTC, and if both time and time zone are omitted the default is 00:00:00 UTC.

The units year and month should 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 unit `month` is defined exactly as `year/12`.
A time coordinate may be indicated by providing the `standard_name` attribute with an appropriate value, and/or the `axis` attribute with the value `T`.

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 it is recommended that the calendar be specified by the attribute `calendar` which is assigned to the time coordinate. The values currently defined for `calendar`, in the CF Standard Document, are:

- `gregorian` or `standard`: Mixed Gregorian/Julian calendar as defined by Udunits. *This is the default.*
- `proleptic_gregorian`: A Gregorian calendar extended to dates before 1582-10-15. That is, a year is a leap year if either (i) it is divisible by 4 but not by 100 or (ii) it is divisible by 400.
- `noleap` or `365_day`: Gregorian calendar without leap years, i.e., all years are 365 days long.
- `all_leap` or `366_day`**:** Gregorian calendar with every year being a leap year, i.e., all years are 366 days long.
- `360_day`: All years are 360 days divided into 30 day months.
- `Julian`: Julian calendar.
- `None`**:** No calendar.

3.3.1 Analysis time: the forecast reference time

`forecast_reference_time`: The forecast reference time in NWP is the "data time", i.e. the time of the analysis from which the forecast was made.It is not the time for which the forecast is valid; the standard name of `time` should be used for that time (definition from the CF Standard Name Table).

```
netcdf forecast_reference_time {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
  variables:
    double mslp(latitude, longitude) ;
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa" ;
      mslp:grid_mapping = "hcrs" ;
      mslp:coordinates = "reftime latitude longitude" ;
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-26T00:00:00Z";
      reftime:calendar = "gregorian";
      reftime:axis = "T";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
}
```

3.3.2 Forecast: the forecast period

`forecast_period`: Forecast period is the time interval between the forecast reference time and the validity time. A period is an interval of time, or the time-period of an oscillation (definition from the CF Standard Name Table).

```
netcdf forecast_period {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
  variables:
    double mslp(latitude, longitude) ;
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa" ;
      mslp:grid_mapping = "hcrs" ;
      mslp:coordinates = "fcstperiod reftime latitude longitude" ;
    double fcstperiod;
      fcstperiod:standard_name = "forecast_period";
      fcstperiod:units = "hours";
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-26T00:00:00Z";
      reftime:calendar = "gregorian";
      reftime:axis = "T";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
    fcstperiod = 6.0 ;
}
```

3.3.3 Valid time

The valid time is the time for which the forecast is valid; the standard name of `time` should be used for this time.

```
netcdf valid_time {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
  variables:
    double mslp(latitude, longitude) ;
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa" ;
      mslp:grid_mapping = "hcrs" ;
      mslp:coordinates = "time fcstperiod reftime latitude longitude" ;
    double time;
      time:standard_name = "time";
      time:units = "hours since 2016-10-26T00:00:00Z";
      time:calendar = "gregorian";
    double fcstperiod;
      fcstperiod:standard_name = "forecast_period";
      fcstperiod:units = "hours";
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-26T00:00:00Z";
      reftime:calendar = "gregorian";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
    fcstperiod = 6.0 ;
    time = 6.0;
}
```

3.4 Realization discrete coordinates

The spatio-temporal coordinates described in previous sections are continuous. Other geophysical quantities may likewise serve as continuous coordinates, for instance density, temperature or radiation wavelength. By contrast, for some purposes there is a need for coordinates which indicate either an ordered list or an unordered collection, and does not correspond to any continuous quantity variable. Consequently such an axis may be called *discrete*. A discrete coordinate may have a dimension but might not have a netCDF coordinate variable. Instead, there might be one or more auxiliary coordinate with this dimension.

`realization`: The term “realization” is used to label a dimension that can be thought of as a statistical sample, e.g., labeling members of a model ensemble.

A realization coordinate may be indicated by providing the `standard_name` attribute with value `realization`, and/or the `axis` attribute with the value `E`[9].

```
netcdf realization {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
    str31 = 31; //auxiliary dimension for string variables
  variables:
    double mslp(latitude, longitude) ;
      mslp:standard_name = "air_pressure_at_sea_level" ;
      mslp:units = "Pa" ;
      mslp:grid_mapping = "hcrs" ;
      mslp:coordinates = "realization reftime latitude longitude" ;
    char realization(str31);
      realization:standard_name = "realization";
      realization:units = "1";
      realization:axis = "E";
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-26T00:00:00Z";
      reftime:calendar = "gregorian";
      reftime:axis = "T";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
    realization = "member1" ;
}
```

9 This value is pending to be part of the CF Standard. Because the importance of having and ensemble axis to express realization coordinate values it has been included here. See https://cf-trac.llnl.gov/trac/ticket/142

4 Missing and valid data

The netCDF convention (NUG appendix B) defines the `_FillValue`, `missing_value`, `valid_min`, `valid_max`, and `valid_range` attributes to indicate missing and valid data.

As it's been stated on the CF Standard Document, the missing values of a variable with `scale_factor` and/or `add_offset` attributes are interpreted relative to the variable's packed values (i.e. the raw values, the values stored in the netCDF file) not the values that result after the scale and offset are applied. Applications that process variables that have attributes to indicate both a transformation (via a scale and/or offset) and missing values should first check that a data value is valid, and then apply the transformation. Note that values that are identified as missing should not be transformed. Since the missing value is outside the valid range it is possible that applying a transformation to it could result in an invalid operation. For example, the default `_FillValue` is very close to the maximum representable value of IEEE single precision floats, and multiplying it by 100 produces an "Infinity" (using single precision arithmetic).

Coordinate reference system values must no have values representing missing or invalid data. The valid_range attributes could be used to define the allowed range for coordinate reference system values.

```
netcdf valid_missing_data {
//global attributes:
    :Conventions = "CF-1.6";
  dimensions:
    x = 292;
    y = 362;
  variables:
    float sit(y, x) ;
      sit:standard_name = "sea_ice_thickness" ;
      sit:units = "m" ;
      sit:coordinates = "y x latitude longitude" ;
      sit:_FillValue = 9.96921E36 ;
    double x(x);
      x:long_name = "i-index of mesh grid";
      x:units = "1";
      x:axis = "X" ;
    double y(y) ;
      y:long_name = "j-index of mesh grid" ;
      y:units = "1" ;
      y:axis = "Y" ;
    double latitude(y, x) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:valid_min = -90.f ;
      latitude:valid_max = 90.f ;
    double longitude(y, x) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:valid_min = -180.f ;
      longitude:valid_max = 180.f ;
}
```

5 Aggregations

When data does not represent the point values of a field, but instead represents some characteristic of the field within cells of finite "volume", a complete description of the variable should include metadata that describes the domain or extent of each cell, and the characteristic of the field that the cell values represent.

It is possible for a single data value to be the result of an operation whose domain is a disjoint set of cells. This is true for many types of climatological averages, for example, the mean January temperature for the years 1970-2000.

To represent cells correctly, the attribute `bounds` will be added to the appropriate coordinate variable(s). The value of bounds is the name of the variable that contains the vertices of the cell boundaries. A boundary variable will have one more dimension than its associated coordinate or auxiliary coordinate variable. The additional dimension should be the most rapidly varying one, and its size is the maximum number of cell vertices. Since a boundary variable is considered to be part of a coordinate variable's metadata, it is not necessary to provide it with attributes such as `long_name` and `units`.

Note that the boundary variable for a set of `N` contiguous intervals is an array of shape `(N,2)`. Although in this case there will be a duplication of the boundary coordinates between adjacent intervals, this representation has the advantage that it is general enough to handle non-contiguous intervals, without modification.

For some calculations, information is needed about the size, shape or location of the cells that cannot be deduced from the coordinates and bounds. In some cases, additional information regarding the grid cell is required. To indicate extra information about the spatial properties of a variable's grid cells, a `cell_measures` attribute may be defined for a variable. This attribute is a string comprising a list of blank-separated pairs of words of the form `measure: name`. For the moment, `area` and `volume` are the only defined measures, but others may be supported in future. The `name` is the name of the variable containing the measure values, which we refer to as a *measure variable*. The dimensions of the measure variable should be the same as or a subset of the dimensions of the variable to which they are related, but their order is not restricted. In the case of `area`, for example, the field itself might be a function of longitude, latitude, and time, but the variable containing the area values would only include longitude and latitude dimensions (and the dimension order could be reversed, although this is not recommended). The variable must have a `units` attribute and may have other attributes such as a `standard_name`.

To describe the characteristic of a field that is represented by cell values, the `cell_methods` attribute is associated to that variable. This is a string attribute comprising a list of blank-separated words of the form `name: method`. Each `name: method` pair indicates that for a coordinate identified by `name`, the cell values representing the field have been determined or derived by the specified `method`. For example, if data values have been generated by computing time means, then this could be indicated with `cell_methods="time: mean"`, assuming here that the name of the time coordinate is `time`.

If more than one cell method is to be indicated, they should be arranged in the order in which they were applied. The left-most operation is assumed to have been applied first: `cell_methods="time: mean longitude: maximum"`

If a data value is representative of variation over a combination of axes, a single method should be prefixed by the names of all the dimensions involved. For instance, the standard deviation of topographic height within a longitude-latitude grid-box could have `cell_methods="latitude: longitude: standard_deviation"`

To indicate more precisely how the cell method was applied, extra information may be included in parentheses `()` after the identification of the `method`. For example, the standard deviation of daily values could be indicated by `cell_methods="time: standard_deviation (interval: 1 day)"`

If there is both standardised and non-standardised information, the non-standardised follows the standardised information and the keyword `comment:`. For instance, an area-weighted mean over latitude could be indicated `cell_methods="latitude: mean (interval: 1 degree_north comment: area-weighted)"`.

A dimension of size one may be the result of "collapsing" an axis by some statistical operation, for instance by calculating a variance from time series data. It is recommended that dimensions of size one be retained (or corresponding scalar coordinate variables be defined) to enable documentation of the method (through the `cell_methods` attribute) and its domain (through the `cell_bounds` attribute).

To provide an indication that a particular cell method is relevant to the data without having to provide a precise description of the corresponding cell, the `name` that appears in a `name: method` pair may be an appropriate `standard_name` (which identifies the dimension) or the string, `area` (rather than the name of a scalar coordinate variable or a dimension with a coordinate variable). This convention cannot be used, however, if the name of a dimension or scalar coordinate variable is identical to *name*.

By default, the statistical method indicated by `cell_methods` is assumed to have been evaluated over the entire horizontal area of the cell. Sometimes, however, it is useful to limit consideration to only a portion of a cell (e.g. a mean over the sea-ice area). For more details how to indicate this , see the CF Standard Document for more details.

A climatological coordinate may use different statistical methods to represent variation among years, within years and within days. For example, the average January temperature in a climatology is obtained by averaging both within years and over years. This is different from the average January-maximum temperature and the maximum January-average temperature. For the former, we first calculate the maximum temperature in each January, then average these maxima; for the latter, we first calculate the average temperature in each January, then find the largest one. As usual, the statistical operations are recorded in the `cell_methods` attribute, which may have two or three entries for the climatological time dimension. For instance: `cell_methods="”time: max within days time: mean over days”"`

5.1 Daily maximum near-surface temperature

```
netcdf daily_maximum_near-surface_temperature {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
    bnds = 2 ;
  variables:
    double tasmax(latitude, longitude) ;
      tasmax:standard_name = "air_temperature" ;
      tasmax:units = "K";
      tasmax:grid_mapping = "hcrs" ;
            tasmax:coordinates  =  "fcstperiod  reftime  time  height  latitude
longitude" ;
      tasmax:cell_methods = "fcstperiod: maximum (interval: 1 hour)";
    double time;
      time:standard_name = "time";
      time:units = "hours since 2016-10-26T00:00:00Z";
      time:calendar = "gregorian";
    double fcstperiod;
      fcstperiod:standard_name = "forecast_period";
      fcstperiod:units = "hours";
      fcstperiod:bounds = "fcstperiod_bnds";
    double fcstperiod_bnds(bnds);
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-26T00:00:00Z";
      reftime:calendar = "gregorian";
    double height ;
      height:standard_name = "height";
      height:units = "m";
      height:positive = "up";
      height:axis = "Z";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
    fcstperiod = 24.0 ;
    fcstperiod_bnds = 0.0, 24.0;
    time = 12.0 ;
    height = 2.0 ;
}
```

5.2 Monthly mean of daily maximum near-surface temperature

```
netcdf monthly_daily_maximum_near-surface_temperature {
  //global attributes:
    :Conventions = "CF-1.6" ;
  dimensions:
    latitude = 180 ;
    longitude = 360 ;
    bnds2 = 2 ;
  variables:
    double tasmax(latitude, longitude) ;
      tasmax:standard_name = "air_temperature" ;
      tasmax:units = "K";
      tasmax:grid_mapping = "hcrs" ;
            tasmax:coordinates  =  "fcstperiod  reftime  time  height  latitude
longitude" ;
      tasmax:cell_methods = "fcstperiod: maximum (interval: 1 hour) reftime:
mean (interval: 1 day)";
    double time;
      time:standard_name = "time";
      time:units = "hours since 2016-10-01T00:00:00Z";
      time:calendar = "gregorian";
    double fcstperiod;
      fcstperiod:standard_name = "forecast_period";
      fcstperiod:units = "hours";
      fcstperiod:bounds = "fcstperiod_bnds";
    double fcstperiod_bnds(bnds2);
    double reftime;
      reftime:standard_name = "forecast_reference_time";
      reftime:units = "hours since 2016-10-01T00:00:00Z";
      reftime:calendar = "gregorian";
      reftime:bounds = "reftime_bnds" ;
    double reftime_bnds(bnds2);
    double height ;
      height:standard_name = "height";
      height:units = "m";
      height:positive = "up";
      height:axis = "Z";
    double latitude(latitude) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:axis = "Y" ;
    double longitude(longitude) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:axis = "X" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
  data:
    reftime = 0.0 ;
    reftime_bnds = 0.0, 744.0 ;
    fcstperiod = 24.0 ;
    fcstperiod_bnds = 0.0, 24.0;
    time = 372.0 ;
    height = 2.0 ;
}
```

6 Properties for data discovery

The following properties 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 and data discovery mechanisms. The attribute values are all character strings. **When a given attribute appears both as a global attribute and as a variable attribute, the variable's version has precedence**. This attributes are optional, but some experiment, project or convention would enforce some of them.

- `title`: This is a succinct description of what is in the DF.
- `institution`: This specifies where the original data was produced.

- `source`: This is method of production of the original data. If it was model-generated, then “source” should include the model and its version, with as much (relevant) additional information as required to help the user. If it is observational data, “source” should characterize it (e.g., "surface observation" or "radiosonde").

- `history`: This provides an audit trail for modifications to the original data. 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. It is recommended that each line begin with a timestamp indicating the date and time of day that the changes to the file were made.

- `references`: This contains published or web-based references that describe the data or methods used to produce it.

- `comment`: This is for additional miscellaneous information about the data or methods used to produce it.

7 Examples

This section it’s just a few examples from working datasets been used as reference for writing this recommendation document.

7.1 UERRA project dataset

Sample GRIB2 file retrieved from MARS and converted to netCDF-CF using UNIDATA’s netcdf-java library v4.6[10].

```
netcdf UERRA_MARS_sample {
//global attributes:
    :Conventions = "CF-1.6" ;
    :Originating_or_generating_Center = "Norrköping" ;
    :Originating_or_generating_Subcenter = "0" ;
    :GRIB_table_version = "17,0" ;
    :Type_of_generating_process = "Analysis" ;
    :file_format = "GRIB-2" ;
    :history = "Read using CDM IOSP GribCollection v3" ;

dimensions:
  x = 565 ;
  y = 565 ;
  time = 1 ;
  height_above_ground = 1 ;

variables:
  int LambertConformal_Projection ;
    LambertConformal_Projection:grid_mapping_name = "lambert_conformal_conic" ;
    LambertConformal_Projection:latitude_of_projection_origin = 48. ;
    LambertConformal_Projection:longitude_of_central_meridian = 8. ;
```

10 The global attribute featureType has been removed.

```
    LambertConformal_Projection:standard_parallel = 48. ;
    LambertConformal_Projection:earth_radius = 6371229. ;
  float x(x) ;
    x:standard_name = "projection_x_coordinate" ;
    x:units = "km" ;
  float y(y) ;
    y:standard_name = "projection_y_coordinate" ;
    y:units = "km" ;
  double reftime ;
    reftime:units = "Hour since 2008-01-01T00:00:00Z" ;
    reftime:standard_name = "forecast_reference_time" ;
    reftime:long_name = "GRIB reference time" ;
    reftime:calendar = "proleptic_gregorian" ;
  double time(time) ;
    time:units = "Hour since 2008-01-01T00:00:00Z" ;
    time:standard_name = "time" ;
    time:long_name = "GRIB forecast or observation time" ;
    time:calendar = "proleptic_gregorian" ;
  float height_above_ground(height_above_ground) ;
    height_above_ground:units = "m" ;
    height_above_ground:long_name = "Specified height level above ground" ;
    height_above_ground:positive = "up" ;
    height_above_ground:Grib_level_type = 103 ;
    height_above_ground:datum = "ground" ;
  float Temperature_height_above_ground(time, height_above_ground, y, x) ;

    Temperature_height_above_ground:long_name = "Temperature @ Specified height
level above ground" ;

    Temperature_height_above_ground:units = "K" ;

    Temperature_height_above_ground:description = "Temperature" ;

    Temperature_height_above_ground:missing_value = NaNf ;

    Temperature_height_above_ground:grid_mapping =
"LambertConformal_Projection" ;

    Temperature_height_above_ground:coordinates = "reftime time
height_above_ground y x " ;

    Temperature_height_above_ground:Grib_Variable_Id = "VAR_0-0-0_L103" ;

    Temperature_height_above_ground:Grib2_Parameter = 0, 0, 0 ;

    Temperature_height_above_ground:Grib2_Parameter_Discipline =
"Meteorological products" ;

    Temperature_height_above_ground:Grib2_Parameter_Category = "Temperature" ;

    Temperature_height_above_ground:Grib2_Parameter_Name = "Temperature" ;

    Temperature_height_above_ground:Grib2_Level_Type = "Specified height level
above ground" ;

    Temperature_height_above_ground:Grib2_Generating_Process_Type =
"Analysis" ;

}
```

7.2 C3S Seasonal dataset

Sample netCDF file for providers of the C3S Seasonal dataset. For more details see the “Guide to netCDF encoding for C3S (seasonal) providers”[11]

```
netcdf C3S_seasonal_total_precipitation {

//global attributes:
    :Conventions = "CF-1.6 C3S-0.1" ;
    :comment = "Produced by Met Office for C3S" ;
    :lineage = "Produced using GloSea5-GC2 v1.0" ;
    :summary = "Seasonal Forecast data produced by Met Office for C3S" ;
    :experiment_id = "mi-am455_PsuiteChanges" ;
    :title = "Operational seasonal forecast data from UK Met Office." ;
    :forecast_type = "hindcast" ;
    :institute_id = "egrr" ;
    :project = "C3S Seasonal Forecast" ;
    :source = "model-generated, GloSea5-GC2" ;
    :contact = "http://copernicus-support.ecmwf.int" ;
    :references = "doi:10.5194/gmd-8-1509-2015" ;
    :forecast_reference_time = "2017-03-01T00:00:00Z" ;
    :keywords = "C3S,Seasonal Forecast,Met Office" ;
    :model_id = "HadGEM3-GC2.0" ;
    :creation_date = "2016-06-24T02:53:46Z" ;
    :institution = "Met Office" ;
    :history = "2016-06-24T02:53:46Z Produced using Met Office computer" ;

  dimensions:
    leadtime = UNLIMITED ; // (1 currently)
    lat = 180 ;
    bounds = 2 ;
    lon = 360 ;
    str31 = 31;

  variables:
    double reftime ;
      reftime:units = "hours since 2017-03-01T00:00:00Z" ;
      reftime:standard_name = "forecast_reference_time" ;
      reftime:calendar = "gregorian" ;
      reftime:long_name = "Start date of the forecast";
    double leadtime(leadtime) ;
      leadtime:bounds = "leadtime_bounds" ;
      leadtime:units = "hours" ;
```

11 https://software.ecmwf.int/wiki/display/C3SS/Guide+to+NetCDF+encoding+for+C3S+providers

```
      leadtime:standard_name = "forecast_period" ;
      leadtime:long_name = "Time elapsed since the start of the forecast" ;
    double leadtime_bounds(leadtime, bounds) ;
    double time(leadtime) ;
      time:units = "hours since 2017-03-01T00:00:00Z" ;
      time:standard_name = "time" ;
      time:calendar = "gregorian" ;
      time:long_name = "Verification time of the forecast" ;
      time:bounds = "time_bounds";
      time:axis = "T" ;
    double time_bounds(leadtime, bounds) ;
    double lat(lat) ;
      lat:axis = "Y" ;
      lat:bounds = "lat_bounds" ;
      lat:units = "degrees_north" ;
      lat:standard_name = "latitude" ;
      lat:long_name = "latitude" ;
      lat:valid_min = -90.0 ;
      lat:valid_max = 90.0 ;
    double lat_bounds(lat, bounds) ;
    double lon(lon) ;
      lon:axis = "X" ;
      lon:bounds = "lon_bounds" ;
      lon:units = "degrees_east" ;
      lon:standard_name = "longitude" ;
      lon:long_name = "longitude" ;
      lon:valid_min = 0.0 ;
      lon:valid_max = 360.0 ;
    double lon_bounds(lon, bounds) ;
    double lweper(leadtime, lat, lon) ;
      lweper:units = "m" ;
      lweper:cell_methods = "leadtime: sum (interval: 1 hour)" ;
      lweper:grid_mapping = "hcrs" ;
         lweper:long_name  =  "Liquid  Water  Equivalent  of  Total  Precipitation
Amount" ;
      lweper:coordinates = "reftime leadtime time lat lon" ;
      lweper:_FillValue = 1.0e20 ;
    char realization(str31) ;
      realization:standard_name = "realization" ;
      realization:axis = "E" ;
      realization:units = "1" ;
    char hcrs ;
      hcrs:grid_mapping_name = "latitude_longitude" ;
```

```
  data:
    leadtime = 48 ;
    leadtime_bounds = 24, 48 ;
    reftime = 0 ;
    time = 36 ;
    time_bounds = 24, 48 ;
    realization = "r01i01p01" ;
}
```

7.3 NEMO dataset from CERA-20C

This example is been used as reference to be used for archiving the NEMOv3.4 component dataset from the CERA-20C project.

```
netcdf CERA-20C_NEMO_sample_MARS_archiving {
//global attributes:
      :Conventions = "CF-1.6" ;
  dimensions:
    depth = 42 ;
    y = 292 ;
    x = 362 ;
    bnd2 = 2 ;
    str2 = 2;
  variables:
    double time ;
      time:units = "hours since 1900-01-01 00:00:00" ;
      time:standard_name = "time" ;
      time:calendar = "gregorian" ;
      time:axis = "T" ;
      time:bounds = "time_bnds" ;
      time:long_name = "time" ;
    double time_bnds(bnd2) ;
    float longitude(y, x) ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
      longitude:valid_min = -180.f ;
      longitude:valid_max = 180.f ;
      longitude:long_name = "longitude" ;
    float latitude(y, x) ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
      latitude:valid_min = -90.f ;
      latitude:valid_max = 90.f ;
      latitude:long_name = "latitude" ;
    float depth(depth) ;
      depth:axis = "Z" ;
      depth:standard_name = "depth_below_geoid" ;
      depth:units = "m" ;
      depth:positive = "down" ;
      depth:long_name = "Depth levels" ;
    char realization(str2) ;
      realization:standard_name = "realization" ;
      realization:axis = "E" ;
      realization:units = "1" ;
    double y(y) ;
      y:axis = "Y" ;
      y:units = "1" ;
      y:long_name = "j-index of mesh grid" ;
    double x(x) ;
      x:axis = "X" ;
      x:units = "1" ;
      x:long_name = "i-index of mesh grid" ;
    float so(depth, y, x) ;
      so:units = "0.001" ;
      so:standard_name = "sea_water_salinity" ;
      so:_FillValue = 0.f ;
      so:long_name = "Practical Salinity" ;
      so:coordinates = "time realization depth latitude longitude" ;
      so:cell_methods = "time: mean within days (interval: 1 hour comment: from
21:00UTC day before to 21:00UTC same day) time: mean over days" ;
      so:mars_paramid = "151130" ;
  data:
   time = 744 ;
   time_bnds = 744, 1416 ;
   realization = "1" ;
}
```