# CF Conformance Requirements and Recommendations 1.8

- The following is a list of requirements and recommendations for a CF conforming netCDF file. They are organized by the section of the CF document that they pertain to.
- This document is intended to be a concise summary of the CF Conventions document. If there are any discrepencies between the two, the conventions document is the ultimate authority.
- This document will be updated as required to correct mistakes or add new material required for completeness or clarity.

## 2.1 Filename

**Requirements:**

- Filename must have ".nc" suffix.

## 2.2 Data Types

**Requirements:**

- CF attributes that take string values must be 1D character arrays.

## 2.3 Naming Conventions

**Recommendations:**

- Variable, dimension and attribute names should begin with a letter and be composed of letters, digits, and underscores.
- No two variable names should be identical when case is ignored.

## 2.4 Dimensions

**Requirements:**

- The dimensions of a variable must all have different names.

**Recommendations:**

- If any or all of the dimensions of a variable have the interpretations (as given by their units or axis attribute) of time (T), height or depth (Z), latitude (Y), or longitude (X) then those dimensions should appear in the relative order T, then Z, then Y, then X in the CDL definition corresponding to the file.
- In files that are meant to conform to the COARDS subset of CF, any dimensions of a variable other than space and time dimensions should be added "to the left" of the space and time dimensions as represented in CDL.

## 2.5.1 Missing data, valid and actual range of data

**Requirements:**

- The `valid_range` attribute must not be present if the `valid_min` and/or `valid_max` attributes are present.
- The `_FillValue` attribute must be the same type as its associated variable.
- The `missing_value` attribute must be the same type as its associated variable.
- The `actual_range` attribute must be of the same type as its associated variable unless there is a `scale_factor` and/or `add_offset` attribute, in which case it must be of the same type as those attributes.
- The `actual_range` attribute must have two elements, of which the first exactly equals the minimum non-missing value occurring in the associated variable after any `scale_factor` and `add_offset` are applied, and the second exactly equals the maximum value in the same way.
- There must not be an `actual_range` attribute if all the data values of the associated variable equal the missing value.
- If both the `actual_range` and `valid_range/valid_min/valid_max` are specified, the values of the `actual_range` must be valid values.

**Recommendations:**

- The value of the `_FillValue` attribute should not be within a specified valid range.
- If both `missing_value` and `_FillValue` be used, they should have the same value.

## 2.6.1 Identification of Conventions

**Requirements:**

- The `Conventions` attribute must be a single text string containing a list of convention names separated by blank space or commas, one of which shall be the full CF string as described below.
- Files that conform to the CF version 1.8 conventions must indicate this by setting the global `Conventions` attribute to contain the CF string value "CF-1.8".

## 2.6.2 Description of File Contents

**Requirements:**

- The `title`, `history`, `institution`, `source`, `references`, and `comment` attributes are all type string.

**Recommendations:**

- The `title` and `history` attributes are only defined as global or groups attributes. If they are used as per variable attributes a CF compliant application should treat them exactly as it would treat any other unrecognized attribute.

## 2.6.3 External variables

**Requirements:**

- The `external_variables` attribute is of string type and contains a blank-separated list of variable names.
- No variable named by `external_variables` is allowed in the file.

# 2.7 Groups

**Requirements:**

- The `Conventions` and `external_variables` attributes must not be used in non-root groups.
- If any dimension of an out-of-group variable has the same name as a dimension of the referring variable, the two must be the same dimension (i.e. they must have the same netCDF dimension ID).
- Variable or dimension paths must follow a UNIX style file convention. They must be formed of words (composed of letters, digits and underscores) and be separated by the slash character ('/'). Paths may begin with either '/', '...' or a word.
- The variable or dimension referenced must exist in the file unless it is an external variable. References can be absolute, relative or with no path, in which case, the variable or dimension must be found in one of the following (in order of precedence):
  - In the referring group
  - In the ancestor group (starting from the direct ancestor and proceeding toward the root group, until it is found)
  - By the lateral search algorithm for coordinate variables only.

**Recommendations:**

- NUG-coordinate variables that are not in the referring group or one of its direct ancestors should be referenced by absolute or relative paths rather than relying on the lateral search algorithm.

# 3 Description of the Data

**Recommendations:**

- All variables should use either the `long_name` or the `standard_name` attributes to describe their contents. Exceptions are boundary and climatology variables.

# 3.1 Units

**Requirements:**

- The `units` attribute is required for all variables that represent dimensional quantities (except

for boundary variables defined in section 7.1 and climatology variables defined in section 7.4 ).

- The type of the `units` attribute is a string that must be recognizable by the udunits package. Exceptions are the units `level, layer, and sigma_level`.
- The `units` of a variable that specifies a `standard_name` must be physically equivalent to the canonical units given in the standard name table, as modified by the `standard_name` modifier, if there is one, according to Appendix C, and then modified by all the methods listed in order by the `cell_methods` attribute, if one is present, according to Appendix E.

**Recommendations:**

- The units `level`, `layer`, and `sigma_level` are deprecated.

## 3.3 Standard Name

**Requirements:**

- The `standard_name` attribute takes a string value comprised of a standard name optionally followed by one or more blanks and a standard name modifier.
- The legal values for the standard name are contained in the standard name table.
- The legal values for the standard name modifier are contained in Appendix C, Standard Name Modifiers.
- If a variable has a `standard_name` of `region` or `area_type`, it must have value(s) from the permitted list.

**Recommendataions:**

- Use of the `standard_name` modifiers `status_flag` and `number_of_observations` is deprecated, and the corresponding `standard_names` are recommended instead.

## 3.5 Flags

**Requirements:**

- The `flag_values` attribute must have the same type as the variable to which it is attached.
- If the `flag_values` attribute is present then the `flag_meanings` attribute must be specified.
- The type of the `flag_meanings` attribute is a string whose value is a blank separated list of words or phrases, each consisting of characters from the alphanumeric set and the following five: '_', '-', '.', '+', '@'.
- The number of `flag_values` attribute values must equal the number of words or phrases appearing in the `flag_meanings` string.
- The number of `flag_masks` attribute values must equal the number of words or phrases appearing in the `flag_meanings` string.
- Variables with a `flag_masks` attribute must have a type that is compatible with bit field expression (char, byte, short and int), not floating-point (float, real, double), and the `flag_masks` attribute must have the same type.

- The `flag_masks` attribute values must be non-zero.
- The `flag_values` attribute values must be mutually exclusive among the set of `flag_values` attribute values defined for that variable.

**Recommendations:**

- When `flag_masks` and `flag_values` are both defined, the Boolean AND of each entry in `flag_values` with its corresponding entry in `flag_masks` should equal the `flag_values` entry, ie, the mask selects all the bits required to express the value.

# 4 Coordinate Types

**Requirements:**

- The `axis` attribute may only be attached to coordinate variables and geometry node coordinate variables (Chapter 7).
- The only legal values of axis are `X`, `Y`, `Z`, and `T` (case insensitive).
- The `axis` attribute must be consistent with the coordinate type deduced from `units` and `positive`.
- The `axis` attribute is not allowed for auxiliary coordinate variables.
- A data variable must not have more than one coordinate variable with a particular value of the `axis` attribute.

# 4.3 Vertical (height or depth) Coordinate

**Requirements:**

- The only legal values for the `positive` attribute are `up` or `down` (case insensitive).

**Recommendations:**

- The `positive` attribute should be consistent with the sign convention implied by the definition of the `standard_name`, if both are provided.

# 4.3.3 Parameterized Vertical Coordinate

**Requirements:**

- The `formula_terms` attribute is only allowed on a coordinate variable which has a `standard_name` listed in Appendix C.
- The type of the `formula_terms` attribute is a string whose value is list of blank separated word pairs in the form `term: var`. The legal values `term` are contained in Appendix C for each valid `standard_name`. The values of `var` must be variables that exist in the file.
- Where indicated by the appropriate definition in Appendix D, the `standard_name` attributes of variables named by the `formula_terms` attribute must be consistent with the `standard_name` of the coordinate variable it is attached to, according to the appropriate definition in Appendix D.

- The `computed_standard_name` attribute is only allowed on a coordinate variable which has a `formula_terms` attribute.
- The `computed_standard_name` attribute is a string whose value must be consistent with the `standard_name` of the coordinate variable it is attached to, and in some cases also with the `standard_name` attributes of variables named by the `formula_terms` attribute, according to the appropriate definition in Appendix D.

# 4.4 Time Coordinate

**Requirements:**

- The time units of a time coordinate variable must contain a reference time.
- The reference time of a time coordinate variable must be a legal time in the specified calendar.

**Recommendations:**

- The use of a reference time in the year 0 to indicate climatological time is deprecated. This restriction only applies to the real-world calendar as used by the udunits package.
- Units of `year` and `month` and any equivalent units should be used with caution.

# 4.4.1 Calendar

**Requirements:**

- The attributes `calendar`, `month_lengths`, `leap_year`, and `leap_month` may only be attached to time coordinate variables.
- The standardized values of the calendar attribute are `gregorian`, `standard`, `proleptic_gregorian`, `noleap`, `365_day`, `all_leap`, `366_day`, `360_day`, `julian`, and `none` (case insensitive). If the `calendar` attribute is given a non-standard value, then the attribute `month_lengths` is required, along with `leap_year` and `leap_month` as appropriate.
- The type of the `month_lengths` attribute must be an integer array of size 12.
- The values of the `leap_month` attribute must be in the range 1-12.
- The values of the `leap_year` and `leap_month` attributes are integer scalars.

**Recommendations:**

- The attribute `leap_month` should not appear unless the attribute `leap_year` is present.
- The time coordinate should not cross the date 1582-10-15 when the default mixed Gregorian/Julian calendar is in use.

# 5 Coordinate Systems

**Requirements:**

- All of a variable's dimensions that are latitude, longitude, vertical, or time dimensions must

have corresponding coordinate variables.

- A coordinate variable must have values that are strictly monotonic (increasing or decreasing).
- A coordinate variable must not have the `_FillValue` or `missing_value` attributes.
- The type of the `coordinates` attribute is a string whose value is a blank separated list of variable names. All specified variable names must exist in the file.
- The dimensions of each auxiliary coordinate must be a subset of the dimensions of the variable they are attached to, with two exceptions. First, a label variable which will have a trailing dimension for the maximum string length. Second a ragged array (Chapter 9, Discrete sampling geometries and Appendix H) uses special, more indirect, methods to connect the data and coordinates.

**Recommendations:**

- The name of a multidimensional coordinate variable should not match the name of any of its dimensions.
- All horizontal coordinate variables (in the Unidata sense) should have an `axis` attribute.
- All horizontal coordinate variables (in the unidata sense) should have an `axis` attribute.

## 5.6 Grid Mappings and Projections

# Requirements:

- The type of the `grid_mapping` attribute is a string whose value is of the following form, in which brackets indicate optional text:

```
grid_mapping_name[: coord_var [coord_var ...]] [grid_mapping_name: [coord_var ...
]]
```

- Note that in its simplest form the attribute comprises just a grid_mapping_name as a single word.
- Each grid_mapping_name is the name of a variable (known as a grid mapping variable), which must exist in the file.
- Each coord_var is the name of a coordinate variable or auxiliary coordinate variable, which must exist in the file. If it is an auxiliary coordinate variable, it must be listed in the coordinates attribute.
- The grid mapping variables must have the `grid_mapping_name` attribute. The legal values for the `grid_mapping_name` attribute are contained in Appendix F.
- The data types of the attributes of the grid mapping variable must be specified in Table 1 of Appendix F.
- If present, the `crs_wkt` attribute must be a text string conforming to the CRS WKT specification described in reference [OGC_CTS].
- `reference_ellipsoid_name`, `prime_meridian_name`, `horizontal_datum_name` and `geographic_crs_name`

must be all defined if any one is defined.

- If `projected_crs_name` is defined then `geographic_crs_name` must be also.

**Recommendations:**

- The grid mapping variables should have 0 dimensions.

# 6.1 Labels

**Requirements:**

- A variable of character type that is named by a `coordinates` attribute is a label variable. This variable must have one or two dimensions. The trailing (CDL order) or sole dimension is for the maximum string length. If there are two dimensions, leading dimension (CDL order) must match one of those of the data variable.

# 7.1 Cell Boundaries

**Requirements:**

- The type of the `bounds` attribute is a string whose value is a single variable name. The specified variable must exist in the file.
- A boundary variable must have the same dimensions as its associated variable, plus have a trailing dimension (CDL order) for the maximum number of vertices in a cell.
- A boundary variable must be a numeric data type.
- If a boundary variable has `units`,`standard_name`, `axis`, `positive`, `calendar`, `leap_month`, `leap_year` or `month_lengths` attributes, they must agree with those of its associated variable.
- Starting with version 1.7, a boundary variable must have a `formula_terms` attribute when it contains bounds for a parametric vertical coordinate variable that has a `formula_terms` attribute. In this case the same terms and named variables must appear in both except for terms that depend on the vertical dimension. For such terms the variable name appearing in the boundary variable's `formula_terms` attribute must differ from that found in the `formula_terms` attribute of the coordinate variable itself. The boundary variable of the `formula_terms` variable must have the same dimensions as the `formula_terms` variable, plus a trailing dimension (CDL order) for the maximum number of vertices in a cell, which must be the same as the trailing dimension of the boundary variable of the parametric vertical coordinate variable. If a named variable in the `formula_terms` attribute of the vertical coordinate variable depends on the vertical dimension and is a coordinate, scalar coordinate or auxiliary coordinate variable then its bounds attribute must be consistent with the equivalent term in `formula_terms` attribute of the boundary variable.

**Recommendations:**

- The points specified by a coordinate or auxiliary coordinate variable should lie within, or on the boundary, of the cells specified by the associated boundary variable.
- Boundary variables should not have the `_FillValue`, `missing_value`, `units`, `standard_name`, `axis`, `positive`, `calendar`, `leap_month`, `leap_year` or `month_lengths` attributes.

## 7.2 Cell Measures

**Requirements:**

- The type of the `cell_measures` attribute is a string whose value is list of blank separated word pairs in the form `measure: var`. The valid values for `measure` are `area` or `volume`. The `var` token specifies a variable that must either exist in the file or be named by the `external_variables` attribute. The dimensions of the variable specified by `var` must be the same as, or be a subset of, the dimensions of the variable to which they are related.
- A measure variable must have units that are consistent with the measure type, i.e., square meters for area measures and cubic meters for volume measures.

## 7.3 Cell Methods

**Requirements:**

- The type of the `cell_methods` attribute is a string whose value is one or more blank separated word lists, each with the form

  ```
  dim1: [dim2: [dim3: ...]] method [where type1 [over type2]] [within|over
  days|years] [(comment)]
  ```

  where brackets indicate optional words. The valid values for `dim1` [`dim2` [`dim3` ...] ] are the names of dimensions of the data variable, names of scalar coordinate variables of the data variable, valid standard names, or the word `area`. The valid values of `method` are contained in Appendix E. The valid values for `type1` are the name of a string-valued auxiliary or scalar coordinate variable with a `standard_name` of `area_type`, or any string value allowed for a variable of `standard_name` of `area_type`. If `type2` is a string-valued auxiliary coordinate variable, it is not allowed to have a leading dimension (the number of strings) of more than one. When the method refers to a climatological time axis, the suffixes for within and over may be appended.

- A given dimension name may only occur once in a `cell_methods` string. An exception is a climatological time dimension.
- The comment, if present, must take the form ([`interval:` *value unit* [`interval:` ...] `comment:`] *remainder* )

  The *remainder* text is not standardized. If no `interval` clauses are present, the entire comment is therefore not standardized. There may be zero `interval` clauses, one `interval` clause, or exactly as many `interval` clauses as there are `dims` to which the method applies. The *value* must be a valid number and the *unit* a string that is recognizable by the udunits package.

**Recommendations:**

- If a data variable has any dimensions or scalar coordinate variables referring to horizontal, vertical or time dimensions, it should have a `cell_methods` attribute with an entry for each of these spatiotemporal dimensions or scalar coordinate variables. (The horizontal dimensions may be covered by an area entry.)

- Except for entries whose cell method is point, all numeric coordinate variables and scalar coordinate variables named by `cell_methods` should have `bounds` or `climatology` attributes.

## 7.4 Climatological Statistics

**Requirements:**

- The `climatology` attribute may only be attached to a time coordinate variable.
- The type of the `climatology` attribute is a string whose value is a single variable name. The specified variable must exist in the file.
- A climatology variable must have the same dimension as its associated time coordinate variable, and have a trailing dimension (CDL order) of size 2.
- A climatology variable must be a numeric data type.
- If a climatology variable has `units`, `standard_name`, or `calendar` attributes, they must agree with those of its associated variable.
- A climatology variable must not have `_FillValue` or `missing_value` attributes.

## 7.5 Geometries

**Requirements:**

- One of the dimensions of the data variable with geometry must be the number of geometries to which the data applies.
- The type of the `geometry` attribute is a string whose value is the name of a geometry container variable. The variable name must exist in the file.
- The geometry container variable must hold `geometry_type` and `node_coordinates` attributes.
- The only legal values of geometry_type are `point`, `line`, and `polygon` (case insensitive).
- For a line `geometry_type`, each geometry must have a minimum of two node coordinates.
- For a polygon `geometry_type`, each geometry must have a minimum of three node coordinates.
- The type of the `node_coordinates` attribute is a string whose value is a blank separated list of variable names. All specified variable names must exist in the file.
- The geometry node coordinate variables must each have an `axis` attribute.
- A geometry container variable must not have more than one node coordinate variable with a particular value of the `axis` attribute.
- The `grid_mapping` and `coordinates` attributes can be carried by the geometry container variable provided they are also carried by the data variables associated with the container.
- If a coordinate variable named by a `coordinates` attribute carried by the geometry container variable or its parent data variable has a `nodes` attribute, then the `nodes` attribute must be a string whose value is a single variable name. The specified variable must be a node coordinate variable that exists in the file.
- If coordinate variables have a `nodes` attribute, then the grid mapping of the coordinate variables

must be the same as the grid mapping of the variables indicated by the `nodes` attribute.

- The geometry node coordinate variables must all have the same single dimension, which is the total number of nodes in all the geometries.
- Nodes for polygon exterior rings must be put in anticlockwise order (viewed from above) and polygon interior rings in clockwise order.
- The single dimension of the part node count variable should equal the total number of parts in all the geometries.
- When more than one geometry instance is present and the `node_count` attribute on the geometry container is missing, the geometry type must be `point`, and the dimension of the node coordinate variables must be one of the dimensions of the data variable.
- If a `part_node_count` variable and a `node_count` variable are present for a given geometry container, then the sum of `part_node_count` values must equal the sum of `node_count` values.
- If the `interior_ring` attribute is present on the geometry container, then the `part_node_count` attribute must also be present on the geometry container.
- The interior ring variable must contain the value 0 to indicate an exterior ring polygon and 1 to indicate an interior ring polygon.
- The single dimension of the interior ring variable must be the same dimension as that of the part node count variable.

# 8.1 Packed Data

**Requirements:**

- The `scale_factor` and `add_offset` attributes must be the same numeric data type.
- If `scale_factor` and `add_offset` are a different type than the variable, then they must be either type float or type double.
- If `scale_factor` and `add_offset` are a different type than the variable, then the variable must be type byte, short or int.

**Recommendations:**

- If `scale_factor` and `add_offset` are type float, the variable should not be of type int.

# 8.2 Compression by Gathering

**Requirements:**

- The `compress` attribute may only be attached to a coordinate variable with an integer data type.
- The type of the `compress` attribute is a string whose value is a blank separated list of dimension names. The specified dimensions must exist in the file.
- The values of the associated coordinate variable must be in the range starting with 0 and going up to the product of the compressed dimension sizes minus 1 (CDL index conventions).