CF Conformance Requirements and Recommendations 1.12 draft

- 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.

UGRID Conventions

Requirements and recommendations relating to the UGRID conventions for storing unstructured (or flexible mesh) data in netCDF files are not described here, but are nonetheless considered as part of the CF conformance requirements and recommendations. See https://github.com/ugrid-conventions/ugrid-conventions for the UGRID conformance requirements and recommendations.

2.1 Filename

Requirements:

- Filename must have ".nc" suffix.

2.2 Data Types

Requirements:

- Any text stored in a CF attribute or variable must be represented in Unicode Normalization Form C and encoded in UTF-8.
- Any attribute of variable-length string type must be a scalar (not an array).

2.3 Naming Conventions

Recommendations:

- Variable, dimension and attribute names should begin with a letter and be composed of letters (A-Z, a-z), digits (0-9), and underscores(_). This corresponds to ASCII characters in the decimal ranges (65-90), (97-122), (48-57), and (95). The corresponding Unicode codepoints are (U+0041-U+005A), (U+0061-U+007A), (U+0030-U+0039), and (U+005F).
- 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 Variables

Requirements:

- A one-dimensional string-valued variable must not have the same name as its dimension (see "Coordinate variable" in Section 1.3).

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 1.12 draft conventions must indicate this by setting the global `Conventions` attribute to contain the CF string value "CF-1.12-draft".

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

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`.
- Dimensionless units for volume fractions defined by UDUNITS (`ppv`, `ppmv`, `ppbv`, `pptv`, `ppqv`) are not allowed in the `units` attribute of any variable which also has a `standard_name` attribute.
- If present, the `units_metadata` attribute must have one of these values: `temperature: on_scale`, `temperature: difference`, `temperature: unknown`, `leap_seconds: none`, `leap_seconds: utc`, or `leap_seconds: unknown`.
- 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.
- If the `standard_name` attribute includes the `standard_error` modifier, the `units_metadata` attribute, if present, must have the value `temperature: difference`.
- If a variable has a `units` attribute that involves a temperature unit, and has a `cell_methods` attribute includes any entry with any of the methods `range`, `standard_deviation` or `variance`, then the `units_metadata` attribute, if present, must have the value `temperature: difference`.
- A variable must not have a `units_metadata` attribute if it has no `units` attribute, or if its `units` do not involve a temperature unit or a reference time unit.

Recommendations:

- The units `level`, `layer`, and `sigma_level` are deprecated.
- Any variable whose `units` involve a temperature unit should also have a `units_metadata` attribute.

3.2 Long Name

Recommendations:

- All data variables and variables containing coordinate data should use either the `long_name` or the `standard_name` attributes to describe their contents.

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.

- The units of a variable named by the `formula_terms` attribute must be consistent with the units defined in Appendix D.

4.4.1 Time Coordinate Units

Requirements:

- The time `units` of a time coordinate variable must contain a reference datetime.

Recommendations:

- Units of `year` and `month` and any equivalent units should be used with caution.
- UDUNITS permits a number of alternatives to the word `since` in the units of time coordinates. All the alternatives have exactly the same meaning in UDUNITS. For compatibility with other software, CF strongly recommends that `since` should be used.

4.4.2 Calendar

Requirements:

- The `calendar` attribute may only be attached to time coordinate variables.
- If present, the value of the `calendar` attribute must be one of the standardized values (case insensitive) detailed in this section, unless the `month_length` attribute is present, in which case the `calendar` attribute must *not* take one of the standardized values.
- The reference datetime of a time coordinate variable must be a legal datetime in the specified calendar.

Recommendations:

- A time coordinate variable should have a `calendar` attribute.
- If the `calendar` attribute is `standard`, `gregorian` (deprecated) or `julian` or absent, the use of time coordinates in year 0 and reference datetimes in year 0 is deprecated.
- The value `standard` should be used instead of `gregorian` in the `calendar` attribute.
- The time coordinate should not cross the date 1582-10-15 when the default mixed Gregorian/Julian calendar is in use.

4.4.3 Leap Seconds

Requirements:

- The reference datetime in time `units` is not allowed to contain seconds equal to or greater than 60, except for valid leap seconds if the `calendar` is `utc`.

- A time coordinate variable must not have a `units_metadata` attribute if it has a `calendar` attribute with a value *other than* one of the following values: `standard`, `gregorian` (deprecated), `proleptic_gregorian`, `julian`.
- If a time coordinate variable has a `units_metadata` attribute then it must have one of these values: `leap_seconds: none`, `leap_seconds: utc`, or `leap_seconds: unknown`.

Recommendations:

- A time coordinate variable should have a `units_metadata` attribute if it has no `calendar` attribute, or if it has a `calendar` attribute with one of the following values: `standard`, `gregorian` (deprecated), `proleptic_gregorian`, `julian`.

4.4.5 Explicitly Defined Calendar

- The `month_lengths`, `leap_year`, and `leap_month` attributes may only be attached to time coordinate variables.
- If the `calendar` attribute of a time coordinate variable 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.

5 Coordinate Systems and Domain

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 three exceptions. First, a label variable of type `char` will have a trailing dimension for the maximum string length. Second, if an auxiliary coordinate variable of a data variable that has been compressed by gathering (8.2 Lossless Compression by Gathering) does not span the compressed dimension, then its dimensions may be any subset of the data variable's uncompressed dimensions, i.e. any of the dimensions of the data variable except the compressed dimension, and any of the dimensions listed by the `compress` attribute of the compressed coordinate variable. Third, 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.
- Deprecated attributes for the `grid_mapping_name` are: `scale_factor_at_projection_origin` for `grid_mapping_name = lambert_cylindrical_equal_area` only, where `standard_parallel` should instead be used; and `straight_vertical_longitude_from_pole` for `grid_mapping_name = polar_stereographic` only, where `longitude_of_projection_origin` should instead be used.

5.8 Domain Variables

Requirements:

- Domain variables must have a `dimensions` attribute.
- The type of the `dimensions` attribute is a string whose value is a blank separated list of dimension names. All specified dimensions must exist in the file. The string may be empty.
- The dimensions of each variable named by the `coordinates` attribute must be a subset of zero or more of the dimensions named by the `dimensions` attribute, 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 domain and coordinates.
- The dimensions of each variable named by the `cell_measures` attribute must be a subset of zero or more of the dimensions named by the `dimensions` attribute.

Recommendations:

- Domain variables should have a `long_name` attribute.
- Domain variables should not have any of the attributes marked in Appendix A as applicable to data variables except those which are also marked as applicable to domain variables.

6.1 Labels

Requirements:

- A string variable that is named by a `coordinates` attribute is a label variable. If the variable is of type `string` it must have at most one dimension, which must match one of those of the data variable. If the variable is of type `char` it must have one or two dimensions, where the trailing (CDL order) or sole dimension is for the maximum string length. If there are two dimensions, the 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 be a numeric data type.
- 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. The trailing dimension must be of size two if the associated variable is one-dimensional, and of size greater than two if the associated variable has more than one dimension.
- Any elements of the boundary variable which contain the `FillValue` must form a consecutive block at the end of the trailing dimension.
- If the associated coordinate variable is one-dimensional and of size greater than one, the

bounds of each cell must be ordered in the same sense as the coordinates (increasing or decreasing).

- A boundary variable can only have inheritable attributes, i.e. any of those marked "BI" in the "Use" column of Appendix A, if they are also present on its parent coordinate variable.
- If a boundary variable has an inheritable attribute then its data type and its value must be exactly the same as the parent variable's attribute.
- 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 include inheritable attributes, i.e. any of those marked "BI" in the "Use" column of Appendix A.

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, with one exception: If a cell measure variable of a data variable that has been compressed by gathering (8.2 Lossless Compression by Gathering) does not span the compressed dimension, then its dimensions may be any subset of the data variable's uncompressed dimensions, i.e. any of the dimensions of the data variable except the compressed dimension, and any of the dimensions listed by the `compress` attribute of the compressed coordinate variable.
- 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 must be sized to contain a single string. If it is a variable of type `string`, it must be scalar or one-dimensional with a length of one. If it is a variable of type `char`, it must be one-dimensional or two-dimensional with a leading dimension (the number of strings) of length 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 either type `float` or type `double`, and if both are present they must be the same type.
- If the `scale_factor` and `add_offset` are type `float`, the data variable must be one of these types: `byte`, `unsigned byte`, `short`, `unsigned short`.
- If the `scale_factor` and `add_offset` are type `double`, the data variable must be one of these types: `byte`, `unsigned byte`, `short`, `unsigned short`, `int`, `unsigned int`.

8.2 Lossless 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).
- The associated coordinate variable must not have an associated boundary variable.

8.3 Lossy Compression by Coordinate Subsampling

Requirements:

- When attached to a data variable, the type of the `tie_points` attribute is a string whose value is a list of blank separated word groups of the following form, in which brackets indicate optional text: `tie_point_variable: [tie_point_variable: ⋯] interpolation_variable`. Each `tie_point_variable` token specifies a tie point variable that must exist in the file, and each `interpolation_variable` token specifies a variable that must exist in the file.
- An interpolation variable must have one of the string-valued attributes `interpolation_name` or `interpolation_description`, but not both. The legal values for the `interpolation_name` attribute are contained in the Interpolation Methods section of Appendix J.
- An interpolation variable must have the attribute `computational_precision`. The legal values for the `computational_precision` attribute are contained in the Interpolation Method Implementation subsection of the Lossy Compression by Coordinate Subsampling section of chapter 8.

- An interpolation variable must have a `tie_point_dimensions` attribute that is a string whose value is a list of blank separated word groups of the following form, in which brackets indicate optional text: `interpolation_dimension: tie_point_interpolation_dimension [interpolation_zone_dimension]`. Each `interpolation_dimension` token specifies a unique interpolation dimension of the parent data variable, each `tie_point_interpolation_dimension` token specifies the tie point interpolation dimension of a unique tie point index variable, and each `interpolation_zone_dimension` token specifies a unique interpolation zone dimension. The tie point interpolation dimensions and interpolation zone dimensions must not be dimensions of the parent data variable.
- The tie point variables associated with each `interpolation_variable` token must all span the same dimensions, which comprise a subset of zero or more dimensions of the parent data variable with the addition of all of the tie point interpolation dimensions identified by the `tie_point_dimensions` attribute of the interpolation variable. A tie point variable must not span both a tie point interpolation dimension and its corresponding interpolation dimension, as defined by the `tie_point_dimensions` mapping.
- An interpolation variable must have a `tie_point_indices` attribute that is a string whose value is a list of blank separated word pairs of the following form: `interpolation_dimension: tie_point_index_variable`. The `interpolation_dimension` tokens specify the same interpolation dimensions as the `tie_point_dimensions` attribute, and each `tie_point_index_variable` token specifies a tie point index variable that must exist in the file.
- A tie point index variable must be a one-dimensional variable with an integer data type.
- The dimension of a tie point index variable must be a tie point interpolation dimension identified by the `tie_point_dimensions` attribute.
- The values of a tie point index variable must be non-negative integers. The first value must be zero, and each subsequent value must be greater than or equal to the previous value. If a value differs by zero or one from its previous value, then it must differ by two or more from its subsequent value.
- The size of an interpolation zone dimension must be equal to the size of the corresponding tie point interpolation dimension minus the number of interpolation areas for that tie point interpolation dimension. The number of interpolation areas is equal one plus the number of occurences of adjacent values differing by zero or one in the corresponding tie point index variable.
- When attached to an interpolation variable, the type of the `interpolation_parameters` attribute is a string whose value is list of blank separated word pairs in the form `term: var`. For each valid `interpolation_name`, the legal values for `term` are described by the "Interpolation Parameter terms" table entry in the Interpolation Methods section of Appendix J. The values of `var` must be interpolation parameter variables that exist in the file.
- The dimensions of an interpolation parameter variable must be a subset of zero or more of the dimensions of the corresponding tie point variables, with the exception that a tie point interpolation dimension may be replaced with its corresponding interpolation zone dimension, as defined by the `tie_point_dimensions` mapping.
- If a tie point variable has `bounds_tie_points` attribute then it must be a string whose value is a single variable name. The specified variable must exist in the file.
- A bounds tie point variable must have the same dimensions as its associated tie points

coordinate variable.

- A bounds tie point variable must be a numeric data type.
- A bounds tie point variable must not have the `_FillValue` or `missing_value` attributes. The requirements on all other bounds tie point variable attributes are the same as for bounds variables described in 7.1 Cell Boundaries.

Recommendations:

- An interpolation variable should have 0 dimensions.
- The recommendations on bounds tie point variable attributes are the same as for bounds variables described in 7.1 Cell Boundaries.

8.4 Lossy Compression via Quantization

Requirements:

- Quantization container variables must have two string-valued attributes, `algorithm` and `implementation`.
- The value of `algorithm` must be one of the values permitted by this section.
- Only floating-point type variables can be quantized. Quantized variables are identified by having a string-valued attribute named `quantization`.
- The value of `quantization` must be the name of the quantization container variable which exists in the file.
- Variables that were quantized must have an integer type attribute named either `quantization_nsb` (if the corresponding quantization variable has the `algorithm` attribute value `bitround`) or `quantization_nsd` (if the corresponding quantization variable has one of the `algorithm` attribute values `bitgroom`, `digitround`, or `granular_bitround`).
- The value of `quantization_nsb` must be in the range `1 <= NSB <= 23` for data type `float` or `real`, and `1 <= NSB <= 52` for data type `double`.
- The value of `quantization_nsd` must be in the range `1 <= NSD <= 7` for data type `float` or `real`, and `1 <= NSD <= 15` for data type `double`.
- Variables that serve as a coordinate variable, or are named by a `coordinates`, `formula_terms`, or `cell_measures` attribute of any other variable must not have a `quantization` attribute.
- The value of `implementation` must take the form "*software-name* version *version-string* [(*optional-information*)]". where brackets indicate optional words.

Appendix D Parametric Vertical Coordinates

Requirements:

- For each element `k` of a vertical coordinate variable with `standard_name = "ocean_sigma_z_coordinate"`, one and only one of the formula terms `sigma(k)` and `zlev(k)` must be missing data. If the optional formula term `nsigma` is supplied, it must equal the number of elements of `zlev` which contain missing data.

Recommendations:

- For a vertical coordinate variable with `standard_name = "ocean_sigma_z_coordinate"`, the formula term `nsigma` should be omitted.
- Versions of the standard before 1.9 should not be used for vertical coordinate variables with `standard_name = "ocean_sigma_z_coordinate"` because these versions are defective in their definition of this coordinate.