1.3. Terminology

The terms in this document that refer to components of a netCDF file are defined in the NetCDF User's Guide (NUG) [NUG] NUG. Some of those definitions are repeated below for convenience.

aggregated data

The data of an aggregation variable, after it has been created in memory by an application program.

aggregated dimension

One of an aggregated variable's ~~A~~ dimensions, spanning the full range of the dimension in the contributing fragments ~~of the aggregated data of an aggregation variable.~~

aggregation variable

A variable containing no data but enabling formation of a data array from the arrays ~~whose data is~~ defined ~~as~~ by the contributing ~~an aggregation of~~ fragments~~, rather than containing its own data~~.

ancestor group

A group from which the referring group is descended via direct parent-child relationships

auxiliary coordinate variable

Any netCDF variable that contains coordinate data, but is not a coordinate variable (in the sense of that term defined by the NUG and used by this standard - see below). Unlike coordinate variables, there is no relationship between the name of an auxiliary coordinate variable and the name(s) of its dimension(s).

boundary variable

A boundary variable is associated with a variable that contains coordinate data. When a data value provides information about conditions in a cell occupying a region of space/time or some other dimension, the boundary variable provides a description of cell extent.

CDL syntax

The ascii format used to describe the contents of a netCDF file is called CDL (network Common Data form Language). This format represents arrays using the indexing

conventions of the C programming language, i.e., index values start at 0, and in multidimensional arrays, when indexing over the elements of the array, it is the last declared dimension that is the fastest varying in terms of file storage order. The netCDF utilities ncdump and ncgen use this format (see NUG section on CDL syntax). All examples in this document use CDL syntax.

cell

A region in one or more dimensions whose boundary can be described by a set of vertices. The term *interval* is sometimes used for one-dimensional cells.

coordinate variable

We use this term precisely as it is defined in the NUG section on coordinate variables. It is a one-dimensional variable with the same name as its dimension [e.g., time(time)], and it is defined as a numeric data type with values in strict monotonic order (all values are different, and they are arranged in either consistently increasing or consistently decreasing order). Missing values are not allowed in coordinate variables. Note that an aggregation coordinate variable is stored as a scalar and has the same name as its aggregated dimension (see Section 2.8, "Aggregation Variables").

fragment

The data ~~A constituent part,~~ found in an external dataset included in the array defined by ~~, of the aggregated data of~~ an aggregation variable.

grid mapping variable

A variable used as a container for attributes that define a specific grid mapping. The type of the variable is arbitrary since it contains no data.

interpolation variable

A variable used as a container for attributes that define a specific interpolation method for uncompressing tie point variables. The type of the variable is arbitrary since it contains no data.

latitude dimension

A dimension of a netCDF variable that has an associated latitude coordinate variable.

local apex group

The nearest (to a referring group) ancestor group in which a dimension of an out-of-group coordinate is defined. The word "apex" refers to position of this group at the vertex of the

tree of groups formed by it, the referring group, and the group where a coordinate is located.

longitude dimension

A dimension of a netCDF variable that has an associated longitude coordinate variable.

multidimensional coordinate variable

An auxiliary coordinate variable that is multidimensional.

nearest item

The item (variable or group) that can be reached via the shortest traversal of the file from the referring group following the rules set forth in the Section 2.7, "Groups".

out-of-group reference

A reference to a variable or dimension that is not contained in the referring group.

path

Paths must follow the UNIX style path convention and may begin with either a '/', '..', or a word.

recommendation

Recommendations in this convention are meant to provide advice that may be helpful for reducing common mistakes. In some cases we have recommended rather than required particular attributes in order to maintain backwards compatibility with COARDS. An application must not depend on a dataset's adherence to recommendations.

referring group

The group in which a reference to a variable or dimension occurs.

scalar coordinate variable

A scalar variable (i.e. one with no dimensions) that contains coordinate data. Depending on context, it may be functionally equivalent either to a size-one coordinate variable (Section 5.7, "Scalar Coordinate Variables") or to a size-one auxiliary coordinate variable (Section 6.1, "Labels" and Section 9.2, "Collections, instances, and elements").

sibling group

Any group with the same parent group as the referring group

spatiotemporal dimension

A dimension of a netCDF variable that is used to identify a location in time and/or space.

tie point variable

A netCDF variable that contains coordinates that have been compressed by sampling. There is no relationship between the name of a tie point variable and the name(s) of its dimension(s).

time dimension

A dimension of a netCDF variable that has an associated time coordinate variable.

vertical dimension

A dimension of a netCDF variable that has an associated vertical coordinate variable.

2.8. Aggregation Variables

An *aggregation variable* is a variable which has been formed by combining (i.e. aggregating) multiple *fragments* that are generally stored in *fragment datasets* that are external to the file containing the aggregation variable, i.e. the *aggregation file*. A fragment is an array of data with sufficient metadata for it to be correctly interpreted in the context of the aggregation, as described by Section 2.8.2 "Fragment Interpretation". The aggregation variable does not contain any actual data, instead it contains instructions on how to create its *aggregated data* in memory as an aggregation of the data from each fragment.

Aggregation provides the utility of being able to view, as a single entity, a dataset that has been partitioned across multiple other datasets, whilst taking up very little extra space on disk (since the aggregation file contains no copies of the data in the fragments). Fragment datasets may be CF-compliant or have any other format, thereby allowing an aggregation variable to act as a CF-compliant view of non-CF datasets. Aggregations may also facilitate~~Use cases for storing aggregations include, but are not limited to~~]: data analysis, ~~as it~~by avoid~~ings~~ the computational expense of deriving the aggregation at the time of analysis; archive curation, ~~as the aggregation can~~by act~~ing~~ as a metadata-rich archive index; and the post-processing of model simulation outputs~~s~~, ~~for~~ by spanning ~~combining output data that have been~~ multiple files written at run time but

together constituting a more cohesive and useful simulation product. ~~written to disk as multiple datasets decomposed in time and space~~.

An aggregation variable must be a scalar (i.e. it has no dimensions). It acts as a container for all of the usual attributes that describe ~~the data~~a variable, with the addition of two special attributes: one that defines ~~the~~ its *aggregated ~~dimensions~~shape*, ~~(~~i.e. ~~the~~ its dimensional size after aggregation)~~s of the aggregated data~~; and one that provides the instructions on how the aggregated data is to be created. The data type of the aggregation variable must be the same data type ~~of~~ as each of the ~~aggregated data~~fragments it includes, but the ~~value of~~ the aggregation variable's single element ~~is immaterial~~can be assigned any value.

Aggregation variables may be used as any kind of variable (data variable, coordinate variable, cell measures variable, etc.), but it is recommended that container variables whose data are immaterial (such as grid mapping variables) ~~are~~ not be encoded as aggregation variables.

Any text applying to a variable in the CF conventions applies in exactly the same way to an aggregation variable in the same role; and any reference to the dimensions or data of a variable applies to the aggregated dimensions or aggregated data, respectively, of an aggregation variable. For instance:

- The dimension of a coordinate variable of an aggregation data variable must be included as one of the aggregated dimensions of the aggregation data variable.
- The name of an aggregation coordinate variable (which is a scalar) must be the same as the name of its single aggregated dimension (identified by its aggregated_dimensions attribute), just as the name of a coordinate variable (which is one-dimensional) must be the same as the name of its single dimension.

The details of how to encode and decode aggregation variables are given in this section, with examples provided in Appendix L, Aggregation Variable Examples.

2.8.1. Aggregated Dimensions and Data

The presence of an aggregated_dimensions attribute indicates that the variable it is attached to is an aggregation variable. This attribute records the names of the aggregated dimensions ~~are stored with the aggregation variable's aggregated_dimensions attribute, and it is the presence of this attribute that identifies the variable as an aggregation variable. The value of the~~

Commented [KT1]: Is this true? Or can each of the fragments be of different data types and the aggregation be something different from them?

Commented [KT2]: Can it be left unassigned?

Commented [KT3]: Not sure what you mean by this. Do you mean any of the rules imposed by CF?

~~aggregated_dimensions attribute is~~ as a blank-separated list. ~~of the aggregated dimension names given in t~~The order of the listed names must ~~which~~ match~~es~~ the order of the dimensions of the aggregated data. If the aggregated data is scalar then there are no aggregated dimensions and the aggregated_dimensions attribute must be an empty string. The aggregated dimensions must exist as dimensions in the aggregation file.

The relative positions of fragments contributing to the aggregated data can be specified in a *fragment array*. The fragment array is two dimensional and of size ixj, where i is the number of aggregated dimensions and j is the maximum number of fragments needed to account for all the aggregated data along any of its dimensions. Each element in the fragment array indicates the position of the fragment relative to other fragments in forming the aggregated data.

The fragments which provide the aggregated data are conceptually organised into ~~a~~ *fragment array*~~s~~ that ~~has~~ have the same number of dimensions as the aggregated data. Each dimension of ~~the~~ a fragment array is called a *fragment array dimension*, ~~and~~ which, by virtue of its position, ~~corresponds to the~~ is associated with one of the aggregated dimension~~s~~, but possibly of smaller size. ~~with the same position in the aggregated data~~. The size of a fragment array dimension is equal to the number of fragments that are needed to span its corresponding aggregated dimension. See the Schematic representation of a fragment array for aggregated data.

The aggregated data are created by concatenating the canonical forms of the fragments' data (see Section 2.8.2 "Fragment Interpretation") along each fragment array dimension, and in the order in which they appear in the fragment array.

Example 2.2. Schematic representation of a fragment array for aggregated data

Fragment array position [0, 0, 0]	**Fragment array position** [0, 0, 1]
Fragment location: file_A.nc Fragment data shape: (17, 91, 180) 17 vertical levels	Fragment location: file_B.nc Fragment data shape: (17, 91, 180) 17 vertical levels

Commented [KT4]: Does the order also have to be consistent across each of the fragments?

Commented [KT5]: Why is it considered an aggregation, if there is a single value, presumably obtained from a single fragment. Is this feature included so that you can transform a non-CF-compliant file into a compliant file?

Commented [KT6]: The rest if this section will need to be rewritten if we adopt my suggested alternative approach linking the fragment arrays to their respective places in the aggregated array.

Formatted: Font: Italic

Commented [KT7]: Don't they actually have to be arrays with the same number of dimensions as the aggregated data? Why does it say "conceptually" here?

Commented [KT8]: Or "treated as"? It seems like there is no reorganization done in the usual case. Right?

Commented [KT9R8]: Or perhaps the sentence could read "The aggregated data obtains its data from *fragment arrays* that have the same number of dimensions as the aggregated data."

Commented [KT10]: Doesn't this have to be 90 if the fragments must be non-overlapping and contiguous (and are assumed to really cover the latitude ranges indicated.

[90, 0] degrees north [0, 180) degrees east	[90, 0] degrees north [180, 360) degrees east
Fragment array position [0, 1, 0] Fragment location: file_C.nc Fragment data shape: (17, 45, 180) 17 vertical levels (0, -45] degrees north [0, 180) degrees east	**Fragment array position** [0, 1, 1] Fragment location: file_D.nc Fragment data shape: (17, 45, 180) 17 vertical levels (0, -45] degrees north [180, 360) degrees east
Fragment array position [0, 2, 0] Fragment location: file_E.nc Fragment data shape: (17, 45, 180) 17 vertical levels (-45, -90] degrees north [0, 180) degrees east	**Fragment array position** [0, 2, 1] Fragment location: file_F.nc Fragment data shape: (17, 45, 180) 17 vertical levels (-45, -90] degrees north [180, 360) degrees east

The fragments, stored in six fragment datasets, are arranged in a three-dimensional fragment array with shape (1, 3, 2). Each fragment spans the entirety of the Z dimension, but only a part of the Y-X plane, which has 1 degree resolution. The fragments combine to create three-dimensional aggregated data that have global Z-Y-X coverage, with shape (17, 181, 360). The Z aggregated dimension is spanned by one fragment, the Y aggregated dimension is spanned by three fragments, and the X aggregated dimension is spanned by two fragments. Note that, since this example is a schematic representation, the C or Fortran order of the dimensions is of no consequence. See Aggregation variable example 4 for a CDL representation of this fragment array.

Commented [KT11]: For 1 deg grid, this should be 180.

The fragment array must be defined by an aggregation variable's aggregated_data attribute. This attribute takes a string value comprising blank-separated elements of the form "*feature: variable*", where *feature* is a case-sensitive keyword that identifies a feature of the fragment array, and *variable* is a *fragment array variable* which provides values for that feature. The features and their values

unambiguously define the fragment array. The order of elements in the aggregated_data attribute is not significant.

The features must comprise either all three of the shape, location, and address keywords; or else it must specify ~~both of~~ the shape and value keywords only. No other combinations of keywords are allowed. These features are defined as follows:

shape

Commented [KT12]: I couldn't figure out how to interpret the information recorded by the variable pointed to by "shape". I proposed an alternative approach, which I find much simpler to explain. In any case, after we decide which approach to use, I'll help to rewrite this (assuming I can understand the approach we adopt.

The integer-valued shape fragment array variable defines the shape of each fragment's data in its canonical form (see Section 2.8.2 "Fragment Interpretation"). In general, the shape fragment array variable is two-dimensional, with the size of the slower-varying dimension (i.e. the first dimension in CDL order, representing rows) being the number of fragment array dimensions, and the size of the more rapidly-varying dimension (i.e. the second dimension in CDL order, representing columns) being the size of the largest fragment array dimension. The rows correspond to the fragment array dimensions in the same order, and each row provides the sizes of the fragments along its corresponding dimension of the fragment array, padded with missing values if there are fewer fragments than the number of columns. The sum of non-missing values in a row must therefore equal the size of the corresponding aggregated dimension. See Aggregation variable example 4, which shows the shape fragment array variable for the fragment array described by the Schematic representation of a fragment array for aggregated data. If the aggregated data is scalar then the shape fragment array variable must be a scalar and contain the value 1. See Aggregation variable example 8.

location

The string-valued location fragment array variable defines the locations of fragment datasets. The array is a vector with length equal to the number of fragments. ~~In general its dimensions correspond to, and have the same sizes as, the fragment array dimensions in the same order as they appear in the conceptual fragment array.~~ A fragment dataset is located with a Uniform Resource Identifier (URI) [URI] that must be either an *absolute URI* (a URI that begins with a scheme component followed by a : character, such as file://data/file.nc, https://remote.host/data/file.nc, s3://remote.host/data/file.nc, or locally_meaningful_protocol://UID), or else a *relative-path URI reference* (a URI that is

not an absolute URI and which does not begin with a / or # character, such as file.nc, ../file.nc, or data/file.nc). A relative-path URI reference is taken as being relative to the location of the aggregation file. If the aggregation file is moved to another location, then a fragment dataset identified by an absolute URI will still be accessible, whereas a fragment dataset identified by a relative-path URI reference will also need be moved to preserve the relative reference. Not all fragment dataset locations need be of the same URI type. See Aggregation variable example 1 and Aggregation variable example 2.

The location fragment array variable may have an extra trailing dimension that allows multiple versions of fragments to be specified. Each version must contain equivalent information, so that any version that exists may be selected for use in the aggregated data. This could be useful when it is known that a fragment could be stored in a number of locations, but it is not known which of them might exist at any given time. For instance, when remotely stored and locally cached versions of the same fragment have been defined, an application program could choose to only retrieve the remote version if the local version does not exist. Every fragment must have at least one version, but not all fragments need to have the same number of versions. Where fragments have fewer versions than others, the trailing dimension must be padded with missing values. See Aggregation variable example 2.

A fragment dataset location may be defined with any number of string substitutions, each of which is provided by the location fragment array variable's substitutions attribute. The substitutions attribute takes a string value comprising blank-separated elements of the form "*substitution: replacement*", where *substitution* is a case-sensitive keyword that defines part of a location fragment array variable value which is to be replaced by *replacement* in order to find the actual fragment dataset location. A location fragment array variable value may include any subset of zero or more of the substitution keywords. After replacements have been made, the fragment dataset location must be an absolute URI or a relative-path URI reference. The substitution keyword must have the form ${*}, where * represents any number of any characters. For instance, the fragment dataset location https://remote.host/data/file.nc could be stored as ${path}file.nc, in conjunction with substitutions="${path}: https://remote.host/data/". The order of elements in the substitutions attribute is not significant, and the substitutions for a given fragment must be such that applying them in any order will result in the same fragment dataset location. The use of substitutions can save space in the aggregation file; and in the event that the fragment locations

need to be updated after the aggregation file has been created, it may be possible to achieve this by modifying the substitutions attribute rather than by changing the actual location fragment array variable values. See Aggregation variable example 3.

address

The address key word is followed by the name of a string array or string scalar variable, which in the case of fragment data in netCDF files, will identify which variable should be extracted from each of the fragment files and be included in the aggregated dataset. ~~address~~More generally, the named array or scalar can be of ~~fragment array variable, that may have~~ any data type used to identify variables in the fragment files.~~, defines how to find each fragment within its fragment dataset, i.e. the address of the fragment.~~ When the variable being aggregated goes by different names in different files, an array variable name is specified, but if all the names are the same, then a scalar variable name is adequate. ~~In general it has the same dimensions in the same order as the location fragment array variable, and must contain a non-missing value corresponding to each fragment version. However, if the address fragment array variable is a scalar, then its single value applies to all versions of all fragments. For a netCDF fragment dataset, the address must be the string-valued netCDF variable name of the fragment. Addresses for other fragment dataset formats are allowed, on the understanding that an application program may choose to ignore any values that it does not understand.~~ See Aggregation variable example 1 and Aggregation variable example 6.

Commented [KT13]: I think "identifier" is a better generic term for the label used to find the variable off interest in the fragment file.

value

When the data values within a fragment are all the same, for each fragment, the value fragment array variable allows each fragment to be represented explicitly by its unique data value, rather than by reference to a fragment dataset. The value fragment array variable dimensions correspond to, and have the same sizes as, the fragment array dimensions in the same order as they appear in the conceptual fragment array. The value fragment array variable may have any data type, and contains each fragment's unique value. A fragment that contains wholly missing data is specified by any missing value indicated by the value fragment array variable. See Aggregation variable example 7, which uses an aggregation ancillary variable to make fragment dataset global attributes available to an aggregation data variable.

Commented [KT14]: In general it would be nice to get rid of the distinction of a "conceptual array" and an "array". I'm afraid this just obscures our guidance.

2.8.2. Fragment Interpretation

The data of a fragment must be converted to its *canonical form* prior to being inserted into the aggregated data. The canonical form of a fragment's data is such that:

- The fragment's data, in its entirety, provide the values for a unique (non-overlapping) and contiguous part of the aggregated data. Each fragment can be visualized as a hyperrectangular block with the same dimensions as the aggregated data but with one or more dimensions being smaller. Together the fragment blocks are all combined to form the aggregated data array, leaving no holes.
- The fragment's data dimensions correspond to the aggregated dimensions in the same order.
- The fragment's data have the same units as the aggregation variable.
- The fragment's data can have missing values as indicated by the aggregation variable.
- The fragment's data are not packed (i.e. not stored using a smaller data type than the original data).
- The fragment's data have the same data type as the aggregation variable.

The conversion of the fragment's data to its canonical form is carried out by the application program which is creating the aggregated data in memory. For fragment datasets, the application program may ignore any fragment metadata that are not needed for the conversion to the canonical form, as well as any other variables that might exist in the fragment dataset. A combination of some of the following operations may be required to convert ~~the~~ a fragment's data to its canonical form:

- If~~, and only if, the~~ a fragment's data has been explicitly defined by ~~its~~ a unique scalar value (as opposed to being defined by a fragment dataset), that value must be broadcast~~ing that value~~ across the shape of the canonical form of the fragment's data.
- When a fragment is missing a dimension, a ~~Inserting missing~~ an appropriate size 1 dimension~~s~~ must be inserted to account for it ~~into the fragment's data~~ (e.g. as required when aggregating two-dimensional fragments into three-dimensional aggregated data).
- If a ~~Transforming the~~ fragment's data ~~to have the same~~is of a data type ~~as~~ different from the aggregated data, it must be transformed. Note that some transformations may result in

Commented [KT15]: Check this.

Commented [KT16]: Not sure I understand. I thought the fragment's data was identical to the "original data".

Commented [KT17R16]: Why is this condition imposed?

Commented [KT18]: What defines the "canonical form"? When you say "fragment's data", does this include both the data array and all of its attributes?

Commented [KT19]: Is there only one?

Commented [KT20]: I think it's easier reading if you make the bulleted points into sentences.

a loss of information, such as could be the case when casting floating point numbers to integers.

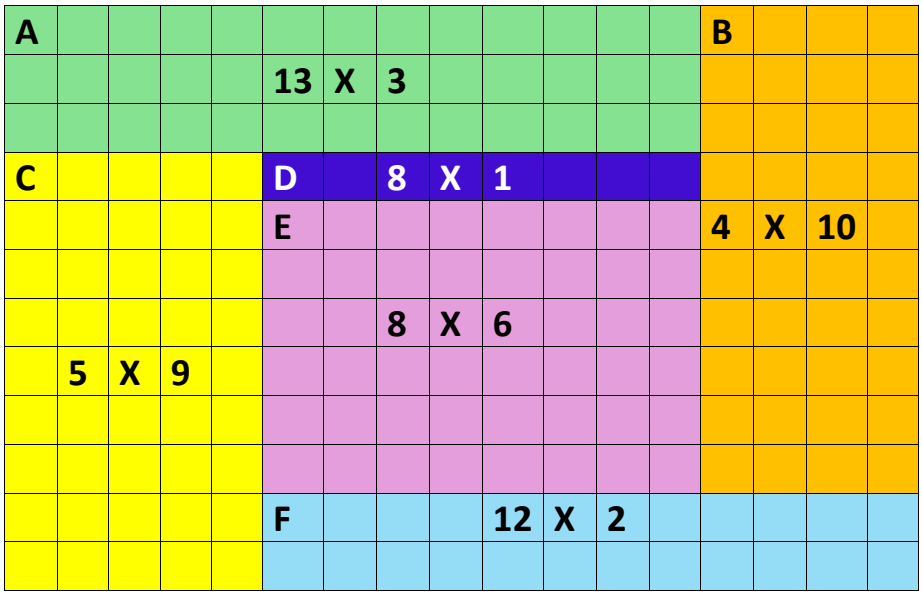
- Any ~~Transforming~~ missing values in the fragment's data may need to be transformed to a different missing value. ~~indicated as missing as defined~~ by the aggregation variable. Note that it is the responsibility of the creator of the aggregation file to ensure that all valid ~~non-missing~~ fragment data values ~~do not coincide with~~are different from any of the missing values ~~indicated~~ defined by the aggregation variable.

- When a ~~Transforming the~~ fragment's data is expressed in units different from but compaitible with ~~to have~~ the aggregation variable's units, the data values must be transformed (e.g. as required when aggregating time fragments whose units have different reference date/times).

- If a ~~Unpacking the~~ fragment's data is packed, it must be unpacked. Note that if the aggregation variable indicates that the aggregated data values are packed (as determined by the attributes defined in Section 8.1, "Packed Data"), then the canonical fragment data values will represent packed values in the aggregated data, and so will be subject to the aggregation variable's unpacking.

Commented [KT21]: Replace with "will be repacked"??

Suppose the following aggregated 2D array (lat x lon) comprises the 6 fragments shown. The boxes outline individual grid cells and the colored sections define the fragments that are pieced together.



Under your mapping scheme, what would the fragment_shape be? (please replace x's with correct values)

fragment_shape= x, x, x,
x, x, x ;

Wouldn't it be easier to decode if you specified the indices in the aggregation array where the first element of a fragment would be saved? Then, for the above you would have (in index space):

fragment_starts_at: 0, 0,
13, 0,
0, 3,
5, 3,
5, 4,
5, 10 ;

Commented [KT22]: Or we might name this "fragment_position" ???

The "shape" keyword would then point to fragment_starts_at, which is dimensioned (# of fragments) x (# of dimensions) = 6 x 2. The code aggregating the fragments would simply insert the entire fragment in the correct location. The ordering of the fragments would be immaterial (although humans might like the ordering I chose above).

Note that for a 3-d fields (lev, lat, lon) with all 3 levels stored in the same file, but broken into the above fragments, you would still have 6 fragments, but there start locations would be:

Formatted: Highlight

fragment_starts_at: 0, 0, 0,
0, 13, 0,
0, 0, 3,
0, 5, 3,
0, 5, 4,
0, 5, 10 ;

If we were to adopt the above approach, then in the following examples "i" would be eliminated as a dimension and replaced with "nfragments", which would be set to the number of fragments contributing to the aggregated variable. We might rename "j" with the more descriptive "ndims". The dimensions of fragment_starts would then be:

int fragment_starts(nfragments,ndims)

Also "fragment_location" would then be declared:

string fragment_location(nfragments)

Appendix L: Aggregation Variable Examples

This appendix contains examples of aggregation variables. Details of how to encode and decode aggregation variables are found in Section 2.8, "Aggregation Variables".

Example L.1 Aggregation variable example 1

```
dimensions:
  time = 12 ;
  level = 1 ;
  latitude = 73 ;
  longitude = 144 ;
  // Fragment array dimensions
  f_time = 2 ;
  f_level = 1 ;
  f_latitude = 1 ;
  f_longitude = 1 ;
  // Fragment shape dimensions
  j = 4 ;        // Equal to the number of aggregated dimensions
  i = 2 ;        // Equal to the size of the largest fragment array dimension

variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "time level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Coordinate variables
  double time(time) ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
  double level(level) ;
    level:standard_name = "height_above_mean_sea_level" ;
    level:units = "m" ;
  double latitude(latitude) ;
```

```
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_time, f_level, f_latitude, f_longitude) ;
  string fragment_address ;
  int fragment_shape(j, i) ;

data:
  temperature = _ ;
  time = 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 ;
  level = ... ;
  latitude = ... ;
  longitude = ... ;
  fragment_location = "January-March.nc", "April-December.nc" ;
  fragment_address = "temperature" ;
  fragment_shape = 3, 9,
                   1, _,
                   73, _,
                   144, _ ;
```

In this example, the temperature data variable is an aggregation variable. Its four-dimensional aggregated data with shape (12, 1, 73, 144) is constructed from two non-overlapping fragments, with data shapes (3, 1, 73, 144) and (9, 1, 73, 144), which span the first 3 and last 9 elements respectively of the time aggregated dimension. The fragment dataset locations are relative-path URI references, and so in this case are assumed to be in the same location as the aggregation file.

The data for the level, latitude and longitude variables are omitted for clarity.

Example L.2 Aggregation variable example 2

```
dimensions:
  time = 12 ;
  level = 1 ;
  latitude = 73 ;
  longitude = 144 ;
  // Fragment array dimensions
  f_time = 2 ;
```

```
  f_level = 1 ;
  f_latitude = 1 ;
  f_longitude = 1 ;
  // Fragment shape dimensions
  j = 4 ;         // Equal to the number of aggregated dimensions
  i = 2 ;         // Equal to the size of the largest fragment array dimension
  // Fragment versions dimension
  versions = 2 ;  // The maximum number of versions for a fragment

variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "time level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Coordinate variables
  double time ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
  double level(level) ;
    level:standard_name = "height_above_mean_sea_level" ;
    level:units = "m" ;
  double latitude(latitude) ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_time, f_level, f_latitude, f_longitude, versions) ;
  string fragment_address ;
  int fragment_shape(j, i) ;

data:
  temperature = _ ;
  time = 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 ;
```

```
level = ... ;
latitude = ... ;
longitude = ... ;
fragment_location = "file://data/January-March.nc",
                    _,
                    "file://data/April-December.nc",
                    "https://remote.host/data/April-December.nc" ;
fragment_address = "temperature" ;
fragment_address_time = "time" ;
fragment_shape = 3, 9,
                 1, _,
                 73, _,
                 144, _ ;
```

This example is similar to Aggregation variable example 1, but now the fragment dataset locations are absolute URIs, and two versions of the second fragment have been provided. The fragment_location variable has the extra trailing dimension versions to accommodate the extra fragment version. There is only one version of the first fragment, so its trailing dimension is padded with missing data.

The data for the level, latitude and longitude variables are omitted for clarity.

Example L.3 Aggregation variable example 3

```
dimensions:
  time = 12 ;
  level = 1 ;
  latitude = 73 ;
  longitude = 144 ;
  // Fragment array dimensions
  f_time = 2 ;
  f_level = 1 ;
  f_latitude = 1 ;
  f_longitude = 1 ;
  // Fragment shape dimensions
  j = 4 ;        // Equal to the number of aggregated dimensions
  j_time = 1 ;   // Equal to the number of aggregated dimensions for time
  i = 2 ;        // Equal to the size of the largest fragment array dimension
  // Fragment versions dimension
  versions = 2 ;  // The maximum number of versions for a fragment
```

Commented [KT23]: Check this. Isn't this in conflict with the declaration below of an aggregation coordinate variable named "time", which is a scalar.?

```
variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "time level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Aggregation coordinate variable
  double time ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
    time:aggregated_dimensions = "time" ;
    time:aggregated_data = "location: fragment_location
                            address: fragment_address_time
                            shape: fragment_shape_time" ;
  // Coordinate variables
  double level(level) ;
    level:standard_name = "height_above_mean_sea_level" ;
    level:units = "m" ;
  double latitude(latitude) ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_time, f_level, f_latitude, f_longitude, versions) ;
    fragment_location:substitutions = "${local}: file://data/
                                       ${remote}: https://remote.host/data/" ;
  string fragment_location_time(f_time, versions) ;
    fragment_location:substitutions = "${local}: file://data/
                                       ${remote}: https://remote.host/data/" ;
  string fragment_address ;
  string fragment_address_time ;
  int fragment_shape(j, i) ;
  int fragment_shape_time(j_time, i) ;
```

Commented [KT24]: Isn't this supposed to be dimensioned time(time)?

```
data:
 temperature = _ ;
 time = _ ;
 level = ... ;
 latitude = ... ;
 longitude = ... ;
 fragment_location = "${local}January-March.nc", _,
                     "${local}April-December.nc", "${remote}April-December.nc" ;
 fragment_location_time = "${local}January-March.nc", _,
                          "${local}April-December.nc", "${remote}April-December.nc" ;
 fragment_address = "temperature" ;
 fragment_address_time = "time" ;
 fragment_shape = 3, 9,
                  1, _,
                  73, _,
                  144, _ ;
 fragment_shape_time = 3, 9 ;
```

Commented [KT25]: A scalar.

This example is similar to Aggregation variable example 2, but now the fragment dataset locations have been defined using the string substitutions given by the substitutions attribute of the fragment_location variable. In addition, time is now an aggregation coordinate variable, with its aggregated data being derived from the same fragment datasets as temperature.

The data for the level, latitude and longitude variables are omitted for clarity.

Example L.4 Aggregation variable example 4

```
dimensions:
 level = 17 ;
 latitude = 181 ;
 longitude = 360 ;
 // Fragment array dimensions
 f_level = 1 ;
 f_latitude = 3 ;
 f_longitude = 2 ;
 // Fragment shape dimensions
 j = 3 ;        // Equal to the number of aggregated dimensions
 i = 3 ;        // Equal to the size of the largest fragment array dimension
```

Commented [KT26]: I this should be 180.

```
variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Coordinate variables
  double level(level) ;
    level:standard_name = "air_pressure" ;
    level:units = "hPa" ;
  double latitude(latitude) ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_level, f_latitude, f_longitude) ;
  string fragment_address ;
  int fragment_shape(j, i) ;

data:
  temperature = _ ;
  level = ... ;
  latitude = ... ;
  longitude = ... ;
  fragment_location = "file_A.nc", "file_B.nc",
                      "file_C.nc", "file_D.nc",
                      "file_E.nc", "file_F.nc" ;
  fragment_address = "temperature" ;
  fragment_shape = 17, _, _,
                   91, 45, 45,
                   180, 180, _ ;
```

Fragment schematic for the above aggregation.

A: [90,0], [0,180]	**B:** [90-0], [180,360]
C: [0,-45], [0,180]	**D:** [0,-45], [180, 360]
E: [-45,-90], [0,180]	**F:** [-45,-90], [180,360]

Note that if my alternative "mapping" approach, described above were adopted, then for this case with 1x1 deg resolution, the start indices would be:

```
fragment_starts_at:  0, 0, 0,
                     0, 0, 180,
                     0, 90, 0,
                     0, 90, 180,
                     0, 135, 0,
                     0, 135, 180
```

This example is an encoding for the conceptual fragment array described in Schematic representation of a fragment array for aggregated data. The temperature data variable is an aggregation of six fragments. The distribution of missing values in the fragment_shape variable indicates that the level aggregated dimension is spanned by one fragment, the latitude aggregated dimension is spanned by three fragments, and the longitude aggregated dimension is spanned by two fragments; and that the shape of the implied fragment array is (1, 3, 2). The row sums of the fragment_shape variable are 17, 181, and 360, which equal the sizes of the level, latitude, and longitude aggregated dimensions, respectively.

The data for the level, latitude and longitude variables are omitted for clarity.

Example L.5 Aggregation variable example 5

```
dimensions:
  time = 12 ;
  level = 1 ;
  latitude = 73 ;
  longitude = 144 ;
```

```
  // Fragment array dimensions
  f_time = 12 ;
  f_level = 1 ;
  f_latitude = 2 ;
  f_longitude = 4 ;
  // Fragment shape dimensions
  j = 4 ;         // Equal to the number of aggregated dimensions
  i = 12 ;        // Equal to the size of the largest fragment array dimension

variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "time level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  double pressure(time, level, latitude, longitude) ;
    temperature:standard_name = "air_pressure" ;
    temperature:units = "hPa" ;
    temperature:cell_methods = "time: mean" ;

  // Coordinate variables
  double time(time) ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
  double level(level) ;
    level:standard_name = "height_above_mean_sea_level" ;
    level:units = "m" ;
  double latitude(latitude) ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_time, f_level, f_latitude, f_longitude) ;
  string fragment_address ;
```

```
  int fragment_shape(j, i) ;

data:
  temperature = _ ;
  pressure = ...  ;
  time = 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 ;
  level = ... ;
  latitude = ... ;
  longitude = ... ;
  fragment_location = ... ;
  fragment_address = "temperature" ;
  fragment_shape = 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                   1, _, _, _, _, _, _, _, _, _, _, _,
                   37, 36, _, _, _, _, _, _, _, _, _, _,
                   36, 36, 36, 36, _, _, _, _, _, _, _, _ ;
```

In this example, the temperature data variable is an aggregation of 96 fragments. The implied fragment array shape is (12, 1, 2, 4), indicating that three of the four aggregated dimensions are spanned by multiple fragments. The pressure data variable is not an aggregation variable.

The data for the pressure, level, latitude and longitude variables, and the fragment_location variable, are omitted for clarity.

Example L.6 Aggregation variable example 6

```
dimensions:
  station = 3 ;
  obs = 15000 ;
  // Fragment array dimensions
  f_station = 3 ;
  // Fragment shape dimensions
  j = 1 ;        // Equal to the number of aggregated dimensions
  i = 3 ;        // Equal to the size of the largest fragment array dimension

variables:
  // Aggregation data variable
  float tas(obs) ;
    tas:standard_name = "air_temperature" ;
    tas:units = "K" ;
    tas:coordinates = "time lat lon alt station_name" ;
```

```
  tas:aggregated_dimensions = "obs" ;
  tas:aggregated_data = "location: fragment_location
                         address: fragment_address
                         shape: fragment_shape" ;
// DSG count variable
int row_size(station) ;
  row_size:long_name = "number of observations per station" ;
  row_size:sample_dimension = "obs" ;

// Aggregation auxiliary coordinate variables
float time ;
  time:standard_name = "time" ;
  time:units = "days since 1970-01-01" ;
  time:aggregated_dimensions = "obs" ;
  time:aggregated_data = "location: fragment_location
                          address: fragment_address_time
                          shape: fragment_shape" ;
float lon(station) ;
  lon:standard_name = "longitude";
  lon:long_name = "station longitude";
  lon:units = "degrees_east";
  lon:aggregated_dimensions = "station" ;
  lon:aggregated_data = "location: fragment_location
                         address: fragment_address_lon
                         shape: fragment_shape_latlon" ;
float lat(station) ;
  lat:standard_name = "latitude";
  lat:long_name = "station latitude" ;
  lat:units = "degrees_north" ;
  lat:aggregated_dimensions = "station" ;
  lat:aggregated_data = "location: fragment_location
                         address: fragment_address_lat
                         shape: fragment_shape_latlon" ;
// Fragment array variables
string fragment_location(f_station) ;
string fragment_address ;
string fragment_address_time(f_station) ;
string fragment_address_lat ;
string fragment_address_lon ;
int fragment_shape(j, i) ;
```

```
  int fragment_shape_latlon(j, i) ;

// global attributes:
  :featureType = "timeSeries" ;

data:
  tas = _ ;
  row_size = 5000, 4000, 6000 ;
  time = _ ;
  lat = _ ;
  lon = _ ;
  fragment_location = "Harwell.nc", "Abingdon.nc", "Lambourne.nc" ;
  fragment_address = "tas" ;
  fragment_address_time = "t1", "t2", "t3" ;
  fragment_address_lat = "lat" ;
  fragment_address_lon = "lon" ;
  fragment_shape = 5000, 4000, 6000 ;
  fragment_shape_latlon = 1, 1, 1 ;
```

Commented [KT27]: This is a scalar.i

In this example, three fragments are aggregated into a collection of DSG timeseries feature types with contiguous ragged array representation. The auxiliary coordinate variables time, lon, and lat are also aggregation variables. The time variables in the fragment datasets all have different netCDF variables names, which differ from the netCDF name of the time aggregation variable. The fragments for all aggregation variables come from the same three fragment datasets, in this case.

No data have been omitted from the CDL.

Example L.7 Aggregation variable example 7

```
dimensions:
  time = 12 ;
  level = 1 ;
  latitude = 73 ;
  longitude = 144 ;
  // Fragment array dimensions
  f_time = 2 ;
  f_level = 1 ;
  f_latitude = 1 ;
  f_longitude = 1 ;
```

```
  // Fragment shape dimensions
  j = 4 ;         // Equal to the number of temperature aggregated dimensions
  i = 2 ;         // Equal to the size of the largest fragment array dimension
  j_uid = 1 ;     // Equal to the number of uid aggregated dimensions

variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:ancillary_variables = "uid" ;
    temperature:aggregated_dimensions = "time level latitude longitude" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Aggregation ancillary variable
  string uid ;
    uid:long_name = "Fragment dataset unique identifiers" ;
    uid:missing_value = "N/A" ;
    uid:aggregated_dimensions = "time" ;
    uid:aggregated_data = "value: fragment_value_uid
                           shape: fragment_shape_uid";
  // Coordinate variables
  double time(time) ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
  double level(level) ;
    level:standard_name = "height_above_mean_sea_level" ;
    level:units = "m" ;
  double latitude(latitude) ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude(longitude) ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location(f_time, f_level, f_latitude, f_longitude) ;
  string fragment_address ;
  int fragment_shape(j, i) ;
```

```
  string fragment_value_uid(f_time) ;
  int fragment_shape_uid(j_uid, i) ;

data:
  temperature = _ ;
  uid = _ ;
  time = 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 ;
  level = ... ;
  latitude = ... ;
  longitude = ... ;
  fragment_location = "January-March.nc", "April-December.nc" ;
  fragment_address = "temperature" ;
  fragment_shape = 3, 9,
                   1, _,
                   73, _,
                   144, _ ;
  fragment_value_uid = "04b9-7eb5-4046-97b-0bf8", "05ee0-a183-43b3-a67-1eca" ;
  fragment_shape_uid = 3, 9 ;
```

This example is similar to Aggregation variable example 1, but now there is the aggregation ancillary variable uid which defines its fragments from the constant values stored in the fragment_value_uid variable, that are intended to be broadcast across the time aggregated dimension.

The data for the level, latitude and longitude variables are omitted for clarity.

Example L.8 Aggregation variable example 8

```
dimensions:

variables:
  // Aggregation data variable
  double temperature ;
    temperature:standard_name = "air_temperature" ;
    temperature:units = "K" ;
    temperature:cell_methods = "time: mean" ;
    temperature:aggregated_dimensions = "" ;
    temperature:aggregated_data = "location: fragment_location
                                   address: fragment_address
                                   shape: fragment_shape" ;
  // Scalar coordinate variables
```

```
  double time ;
    time:standard_name = "time" ;
    time:units = "days since 2001-01-01" ;
  double height ;
    level:standard_name = "height" ;
    level:units = "m" ;
  double latitude ;
    latitude:standard_name = "latitude" ;
    latitude:units = "degrees_north" ;
  double longitude ;
    longitude:standard_name = "longitude" ;
    longitude:units = "degrees_east" ;
  // Fragment array variables
  string fragment_location ;
  string fragment_address ;
  int fragment_shape ;

data:
  temperature = _ ;
  time = 0 ;
  height = 1.5 ;
  latitude = 18.53 ;
  longitude = 73.81 ;
  fragment_location = "file.nc" ;
  fragment_address = "tas" ;
  fragment_shape = 1 ;
```

An example of an aggregation variable with scalar aggregated data.