Chapter 9 Discrete Sampling Geometries

This chapter provides representations for **discrete sampling geometries,** such as time series, vertical profiles and trajectories. Discrete sampling geometry datasets are characterized by a dimensionality that is lower than that of the space-time region that is sampled; discrete sampling geometries are typically “paths” through space-time.

9.1 Features and feature types

Each type of discrete sampling geometry (point, time series, profile or trajectory) is defined by the relationships among its spatiotemporal coordinates. We refer to the type of discrete sampling geometry as its **featureType**. The term “**feature”** refers herein to a **single instance of the discrete sampling geometry** (such as a single time series). The representation of such features in a CF dataset was supported previous to the introduction of this chapter using a particular convention, which is still supported (that described by section 9.3.1). This chapter describes further conventions which offer advantages of efficiency and clarity for storing a collection of features in a single file. When using these new conventions, <u>the features contained within a collection must always be of the same type; and all the collections in a CF file must be of the same feature type.</u> (Future versions of CF may allow mixing of multiple feature types within a file.) Table 9.1 presents the feature types covered by this chapter.

featureType	**Description of a single feature with this discrete sampling geometry**	
	Form of a data variable containing values defined on a collection of these features	**Mandatory space-time coordinates for a collection of these features**
point	a single data point (having no implied coordinate relationship to other points)	
	data(i)	x(i) y(i) t(i)
timeSeries	a series of data points at the same spatial location with monotonically(*) increasing times	
	data(i,o)	x(i) y(i) t(i,o)
trajectory	a series of data points along a path through space with monotonically (*) increasing times	
	data(i,o)	x(i,o) y(i,o) t(i,o)
profile	an ordered (*) set of data points along a vertical line at a fixed horizontal position and fixed time	
	data(i,o)	x(i) y(i) z(i,o) t(i)
timeSeriesProfile	a series of profile features at the same horizontal position with monotonically increasing (*) times	
	data(i,p,o)	x(i) y(i) z(i,p,o) t(i,p)
trajectoryProfile	a series of profile features located at points ordered (*) along a trajectory	
	data(i,p,o)	x(i,p) y(i,p) z(i,p,o) t(i,p)

(*) see discussion of disordered coordinates in section 9.5

Table 9.1. Logical structure and mandatory coordinates for discrete sampling geometry featureTypes.

In Table 9.1 the spatial coordinates x and y typically refer to longitude and latitude but other horizontal coordinates could also be used (see sections 4 and 5.6). The spatial coordinate z refers to vertical position. The time coordinate is indicated as t. The space-time coordinates that are indicated for each feature are mandatory. However a featureType may also include other space-time coordinates which are not mandatory (notably the z coordinate). The array subscripts that are shown illustrate only the logical structure of the data. The subscripts found in actual CF files are determined by the specific type of representations (see section 9.3).

The designation of dimensions as mandatory precludes the encoding of data variables where geo-positioning cannot be described as a discrete point location. Problematic examples include:

- time series that refer to a geographical region (e.g. the northern hemisphere), a volume (e.g. the troposphere), or a geophysical quantity in which geolocation information is inherent (e.g. the Southern Oscillation Index (SOI) is the difference between values at two point locations);
- vertical profiles that similarly represent geographically area-averaged values; and
- paths in space that indicate a geographically located feature, but lack a suitable time coordinate (e.g. a meteorological front).

Future versions of CF will generalize the concepts of geolocation to encompass these cases. As of CF version 1.6 such data can be stored using the representations that are documented here by two means: 1) by utilizing the orthogonal multidimensional array representation and omitting the featureType attribute; or 2) by assigning arbitrary coordinates to the mandatory dimensions. For example a globally-averaged latitude position (90s to 90n) could be represented arbitrarily (and poorly) as a latitude position at the equator.

9.2 Collections, instances and elements

In Table 9.1 the dimension with subscript i identifies a particular feature within a collection of features. It is called the **instance dimension**. One-dimensional variables in a Discrete Geometry CF file, which have *only* this dimension (such as x(i) y(i) and z(i) for a timeseries), are **instance variables**. Instance variables provide the metadata that differentiates individual features.

The subscripts o and p distinguish the data elements that compose a single feature. For example in a collection of **timeSeries** features, each time series instance, i, has data values at various times, o. In a collection of **profile** features, the subscript, o, provides the index position along the vertical axis of each profile instance. We refer to data values in a feature as its **elements**, and to the dimensions of o and p as **element dimensions**. Each feature can have its own set of element subscripts o and p. For instance, in a collection of timeSeries features, each individual timeSeries can have its own set of times. The notation t(i,o) means there is a set of times with subscripts o for the elements of each feature i. Feature instances within a collection need not have the same numbers of elements. If the features do all have the same number of elements, and the sequence of element coordinates is identical for all features, savings in simplicity and space are achievable by storing only one copy of these coordinates. This is the essence of the orthogonal multidimensional representation (see section 9.3.1).

If there is only a single feature to be stored in a data variable, there is no need for an instance dimension and it is permitted to omit it. The data will then be one-dimensional, which is a special (degenerate) case of the multidimensional array representation. The instance variables will be scalar coordinate variables; the data variable and other auxiliary coordinate variables will have only an element dimension and not have an instance dimension, e.g. data(o) and t(o) for a single timeSeries.

9.3 Representations of collections of features in data variables

The individual features within a collection need not necessarily contain the same number of elements. For instance observed *in situ* time series will commonly contain unique numbers of time points, reflecting different deployment dates of the instruments. Other data sources, such as the output of numerical models, may commonly generate features of identical size. CF offers multiple representations to allow the storage to be optimized for the character of the data. Four types of representation are utilized in this chapter:

- two **multidimensional array representations,** in which each feature instance is allocated the identical amount of storage space. In these representations the instance dimension and the element dimension(s) are distinct CF coordinate axes (typical of coordinate axes discussed in chapter 4); and
- two **ragged array representations,** in which each feature is provided with the minimum amount of space that it requires. In these representations the instances of the individual features are stacked sequentially along the same array dimension as the elements of the features; we refer to this combined dimension as the **sample dimension**.

In the multidimensional array representations, data variables have both an instance dimension and an element dimension. The dimensions may be given in any order. If there is a need for either the instance or an element dimension to be the netCDF unlimited dimension (so that more features or more elements can be appended), then that dimension must be the outer dimension of the data variable i.e. the leading dimension in CDL.

In the ragged array representations, the instance dimension (`i`), which sequences the individual features within the collection, and the element dimension, which sequences the data elements of each feature (`o` and `p`), both occupy the same dimension (the sample dimension). If the sample dimension is the netCDF unlimited dimension, new data can be appended to the file.

In all representations, the instance dimension (which is also the sample dimension in ragged representations) may be set initially to a size that is arbitrarily larger than what is required for the features which are available at the time that the file is created. Allocating unused array space in this way (pre-filled with missing values -- see also section 9.6, *Missing data*), can be useful as a means to reserve space that will be available to add features at a later time.

9.3.1 Orthogonal multidimensional array representation

The **orthogonal multidimensional array representation**, the simplest representation, can be used if each feature instance in the collection has identical coordinates along the element axis of the features. For example, for a collection of the timeSeries that share a common set of times, or a collection of profiles that share a common set of vertical levels, this is likely to be the natural representation to use. In both examples, there will be longitude and latitude coordinate variables, x(i), y(i), that are one-dimensional and defined along the instance dimension.

Table 9.2 illustrates the storage of a data variable using the orthogonal multidimensional array representation. The data variable holds a collection of 4 features. The individual features, distinguished by color, are sequenced along the horizontal axis by the instance dimension indices, i1, i2, i3, i4. Each instance contains three elements, sequenced along the vertical with element dimension indices, o1, o2, o3. The i and o subscripts would be interchanged (i.e. Table 9.2 would be transposed) if the element dimension were the netCDF unlimited dimension.

(i1, o1)	(i2, o1)	(i3, o1)	(i4, o1)
(i1, o2)	(i2, o2)	(i3, o2)	(i4, o2)
(i1, o3)	(i2, o3)	(i3, o3)	(i4, o3)

Table 9.2 The storage of a data variable using the orthogonal multidimensional array representation (subscripts in CDL order).

The instance variables of a dataset corresponding to Table 9.2 will be one-dimensional with size 4 (for example, the latitude locations of timeSeries),

lat(i1)	lat(i2)	lat(i3)	lat(i4)

and the element coordinate axis will be one-dimensional with size 3 (for example, the time

time(o1)
time(o2)
time(o3)

time(o4)

coordinates that are shared by all of the timeSeries). This representation is consistent with the multidimensional fields described in chapter 5; the characteristic that makes it atypical from chapter 5 (though not incompatible) is that the instance dimension is a discrete axis (see section 4.5).

9.3.2 Incomplete multidimensional array representation

The **incomplete multidimensional array representation** can used if the features within a collection do not all have the same number of elements, but sufficient storage space is available to allocate the number of elements required by the longest feature to all features. That is, features that are shorter than the longest feature must be padded with missing values to bring all instances to the same storage size. This representation sacrifices storage space to achieve simplicity for reading and writing.

Table 9.3 illustrates the storage of a data variable using the orthogonal multidimensional array representation. The data variable holds a collection of 4 features. The individual features, distinguished by color, are sequenced by the instance dimension indices, i1, i2, i3, i4. The instances contain respectively 2, 4, 3 and 6 elements, sequenced by the element dimension index with values of o1, o2, o3, ... The i and o subscripts would be interchanged (i.e. Table 9.3 would be transposed) if the element dimension were the netCDF unlimited dimension.

(i1, o1)	(i2, o1)	(i3, o1)	(i4, o1)
(i1, o2)	(i2, o2)	(i3, o2)	(i4, o2)
	(i2, o3)	(i3, o3)	(i4, o3)
	(i2, o4)		(i4, o4)
			(i4, o5)
			(i4, o6)

Table 9.3. The storage of data using the incomplete multidimensional array representation (subscripts in CDL order).

9.3.3 Contiguous ragged array representation

The **contiguous ragged array representation** can be used only if the size of each feature is known at the time that it is created. In this representation the data for each feature will be contiguous on disk, as shown in Table 9.4.

(i1, o1)
(i1, o2)
(i2, o1)
(i2, o2)
(i2, o3)
(i2, o4)
(i3, o1)

(i3, o2)
(i3, o3)
(i4, o1)
(i4, o2)
(i4, o3)
(i4, o4)
(i4, o5)
(i4, o6)

Table 9.4. The storage of data using the contiguous ragged representation (subscripts in CDL order).

In this representation, the file contains a **count variable**, which must be of type integer and

count(i1)	count(i2)	count(i3)	count(i4)
2	4	3	6

must have the instance dimension as its sole dimension. The count variable contains the number of elements that each feature has. This representation and its count variable are identifiable by the presence of an attribute, **sample_dimension,** found on the count variable, which names the sample dimension being counted. For indices that correspond to features, whose data have not yet been written, the count variable should have a value of zero or a missing value.

9.3.4 Indexed ragged array representation

The **indexed ragged array representation** stores the features interleaved along the sample dimension in the data variable as shown in Table 9.4. The canonical use case for this representation is the storage of real-time data streams that contain reports from many sources; the data can be written as it arrives.

(i1, o1)	0
(i2, o1)	1
(i3, o1)	2
(i4, o1)	3
(i4, o2)	3
(i2, o2)	1
(i4, o3)	3
(i4, o4)	3
(i1, o2)	0
(i2, o3)	1
(i3, o2)	2
(i4, o5)	3
(i3, o3)	2

(i2, o4)	1
(i4, o6)	3

Table 9.4 The storage of data using the indexed ragged representation (subscripts in CDL order). The left hand column illustrates a data variable; the right hand column contains the values of the index variable.

In this representation, the file contains an **index variable**, which must be of type integer, and must have the sample dimension as its single dimension. The index variable contains the zero-based index of the feature to which each element belongs. This representation is identifiable by the presence of an attribute, **instance_dimension,** on the index variable, which names the dimension of the instance variables. For those indices of the sample dimension, into which data have not yet been written, the index variable should be pre-filled with missing values.

9.4 FeatureType attribute

A global attribute, **`featureType`**, is required for all Discrete Geometry representations except the orthogonal multidimensional array representation, for which it is highly recommended. The exception is allowed for backwards compatibility, as discussed in 9.3.1. A Discrete Geometry file may include arbitrary numbers of data variables, but (as of CF v1.6) all of the data variables contained in a single file must be of the single feature type indicated by the global `featureType` attribute, if it is present.[1] The value assigned to the `featureType` attribute is case-insensitive; it must be one of the string values listed in the left column of Table 9.1.

9.5 Coordinates and metadata

Every feature within a Discrete Geometry CF file must be unambiguously associated with an collection of instance variables that identify the feature and provide other metadata as needed to describe it. Every element of every feature must be unambiguously associated with its space and time coordinates and with the feature that contains it. The `coordinates` attribute must be attached to every data variable to indicate the spatiotemporal coordinate variables that are needed to geo-locate the data.

Where feasible a variable with the attribute **`cf_role`** should be included. The only acceptable values of cf_role for Discrete Geometry CF data sets are `"timeseries_id", "profile_id",` and `"trajectory_id"`. The variable carrying the cf_role attribute may have any data type. When a variable is assigned this attribute, it must provide a unique identifier for each feature instance. CF files that contain timeSeries, profile or trajectory featureTypes, should include only a single occurrence of a `cf_role` attribute; CF files that contain timeSeriesProfile or trajectoryProfile may contain two occurrences, corresponding to the two levels of structure in these feature types.

It is not uncommon for observational data to have two sets of coordinates for particular coordinate axes of a feature: a nominal point location and a more precise location that varies with the elements in the feature. For example, although an idealized vertical profile is measured at a fixed horizontal position and time, a realistic representation might include the time variations and horizontal drift that occur during the duration of the sampling. Similarly, although an idealized time series exists at a fixed lat-long position, a realistic representation of a moored ocean time series might include the "watch cycle" excursions of horizontal position that occur as a result of tidal currents.

CF Discrete Geometries provides a mechanism to encode both the nominal and the precise positions, while retaining the semantics of the idealized feature type. Only the set of coordinates which are regarded as the nominal (default or preferred) positions should be indicated by the attribute `axis`, which should be assigned string values to indicate the orientations of the axes (`"X", "Y", "Z"`, or `"T"`). See example A9.2.3.2. Auxiliary coordinate variables containing the nominal and the precise positions should be listed in the relevant `coordinates` attributes of data variables. In orthogonal representations the nominal positions could be coordinate variables, which do not need to be listed in the `coordinates` attribute, rather than auxiliary coordinate variables.

Coordinate bounds may optionally be associated with coordinate variables and auxiliary coordinate variables using the `bounds` attribute, following the conventions described in section 7.1. Coordinate bounds are especially important for accurate representations of model output data using discrete geometry representations; they record the boundaries of the model grid cells.

If there is a vertical coordinate variable or auxiliary coordinate variable, it must be identified by the means specified in section 4.3. The use of the attribute `axis="Z"` is recommended for clarity. A `standard_name` attribute (see section 3.3) that identifies the vertical coordinate is recommended, e.g. "altitude", "height", etc. . (See the CF Standard Name Table).

While the semantics of time series and profiles requires an ordered (monotonic) progression of coordinates along the time axis or the vertical axis (respectively), there are circumstances, such as real time data collection, under which it is desirable to allow these coordinates to be disordered. Since CF coordinate axes may not be disordered, auxiliary coordinate axes must be used in these circumstances. In general, it is recommended that coordinates be ordered, where ever it is feasible to do so, as some application software may assume ordered coordinates.

9.6 Missing Data

Auxiliary coordinate variables (spatial and time) must contain missing values to indicate a void in data storage in the file but must not have missing data for any other reason. This situation may arise for unused elements in the incomplete multidimensional array representation, and in any representation if the instance dimension is set to a larger size than the number of features currently stored. It is not permitted for auxiliary coordinate variables to have missing values for elements where there is non-missing data. Where *any* auxiliary coordinate variable contains a missing value, *all* other coordinate, auxiliary coordinate and data values corresponding to that element should *also* contain missing values. Data variables should (as usual) also contain missing values to indicate when there is no valid data available for the element, although the coordinates are valid.

Similarly, for indices where the instance variable identified by `cf_role` contains a missing value indicator, all other instance variable should also contain missing values.

Appendix A9: Annotated Examples of Discrete Geometries

A9.1 Annotated examples: Point Data

To represent data at scattered locations and times with no implied relationship among coordinate positions, both data and coordinates must share the same (sample) instance dimension.   Because each feature contains only a single data element, there is no need for a separate element dimension. The representation of point features is a special, degenerate case of the standard four representations. The `coordinates` attribute is used on the data variables to unambiguously identify the relevant space and time auxiliary coordinate variables.

Example A9.1.1. Point data.

```
dimensions:
   obs = 1234 ;

variables:
   double time(obs) ;
       time:standard_name = “time”;
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(obs) ;
       lon:standard_name = "longitude";
       lon:long_name = "longitude of the observation";
       lon:units = "degrees_east";
   float lat(obs) ;
       lat:standard_name = "latitude";
       lat:long_name = "latitude of the observation" ;
       lat:units = "degrees_north" ;
   float alt(obs) ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   float humidity(obs) ;
       humidity:standard_name = "specific_humidity" ;
       humidity:coordinates = "time lat lon alt" ;
   float temp(obs) ;
       temp:standard_name = "air_temperature" ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt" ;

attributes:
   :featureType = "point";
```

In this example, the humidity(i) and temp(i) data are associated with the coordinate values time(i), lat(i), lon(i), and alt(i). The obs dimension may optionally be the netCDF unlimited dimension of the netCDF file.

A9.2 Annotated examples: Time Series Data

Data may be taken over periods of time at a set of discrete point, spatial locations called stations (see also discussion in 9.1). The set of elements at a particular station is referred to as a timeSeries feature and a data variable may contain a collection of such features. The instance dimension in the case of timeSeries specifies the number of time series in the collection and is also referred to as the **station dimension**. The instance variables, which have just this dimension, including latitude and longitude for example, are also referred to as **station variables** and are considered to contain information describing the stations. The station variables may contain missing values, allowing one to reserve space for additional stations that may be added at a later time, as discussed in section 9.6. In addition,

- It is strongly recommended that there should be a station variable (which may be of any type) with the attribute `cf_role=”timeseries_id”`, whose values uniquely identify the stations.
- It is recommended that there should be station variables with standard_name attributes `"station_description"`, `"surface_altitude"` and `“station_wmo_id”` when applicable.

All the representations described in section 9.3 can be used for time series. The global attribute `featureType=”timeSeries”` (case-insensitive) must be included.

A9.2.1 Orthogonal multidimensional array representation of timeSeries

If the time series instances have the same number of elements and the time values are identical for all instances, you may use the orthogonal multidimensional array representation. This has either a one-dimensional coordinate variable, time(time), provided the time values are ordered monotonically, or a one-dimensional auxiliary coordinate variable, time(o), where o is the element dimension. In the former case, listing the time variable in the `coordinates` attributes of the data variables is optional.

Example A9.2.1.1. Timeseries with common element times in a time coordinate variable using the orthogonal multidimensional array representation.

```
dimensions:
  station = 10 ;  // measurement locations
  time = UNLIMITED ;
variables:
  float humidity(station,time) ;
    humidity:standard_name = "specific humidity" ;
    humidity:coordinates = "lat lon alt" ;
  double time(time) ;
    time:standard_name = "time";
    time:long_name = "time of measurement" ;
    time:units = "days since 1970-01-01 00:00:00" ;
  float lon(station) ;
    lon:standard_name = "longitude";
    lon:long_name = "station longitude";
    lon:units = "degrees_east";
  float lat(station) ;
    lat:standard_name = "latitude";
    lat:long_name = "station latitude" ;
    lat:units = "degrees_north" ;
  float alt(station) ;
    alt:long_name = "vertical distance above the surface" ;
    alt:standard_name = "height" ;
    alt:units = "m";
    alt:positive = "up";
    alt:axis = "Z";
  char station_name(station, name_strlen) ;
    station_name:long_name = "station name" ;
    station_name:cf_role = "timeseries_id";
attributes:
    :featureType = "timeSeries";
```

In this example, `humidity(i,o)` is element o of time series i, and associated with the coordinate values `time(o)`, `lat(i)`, and `lon(i)`. Either the instance (station) or the element (time) dimension may optionally be the netCDF unlimited dimension.

A9.2.2 Incomplete multidimensional array representation of timeSeries

Much of the simplicity of the orthogonal multidimensional representation can be preserved even in cases where individual time series have different time coordinate values. All time series must be allocated the amount of storage needed by the longest, so the use of this representation will trade off simplicity against storage space in some cases.

Example A9.2.2.1. Timeseries of station data in the incomplete multidimensional array representation.

```
dimensions:
   station = UNLIMITED ;
   obs = 13 ;

variables:
   float lon(station) ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat(station) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   float alt(station) ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   char station_name(station, name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";
   int station_info(station) ;
       station_info:long_name = "any kind of station info" ;
   float station_elevation(station) ;
       station_elevation:long_name = "height above the geoid" ;
       station_elevation:standard_name = "surface_altitude" ;
       station_elevation:units = "m";

   double time(station, obs) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
       time:missing_value = -999.9;
   float humidity(station, obs) ;
       humidity:standard_name = “specific_humidity” ;
       humidity:coordinates = "time lat lon alt" ;
       humidity:_FillValue = -999.9;
   float temp(station, obs) ;
       temp:standard_name = “air_temperature” ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt" ;
       temp:_FillValue = -999.9;

attributes:
       :featureType = "timeSeries";
```

In this example, the humidity(i,o) and temp(i,o) data for element o of time series i are associated with the coordinate values time(i,o), lat(i), lon(i) and alt(i). Either the instance (station) dimension or the element (obs) dimension could be the unlimited dimension of a netCDF file. Any unused elements of the data and auxiliary coordinate variables must contain the missing data flag value(section 9.6).

A9.2.3 Single time series, including deviations from a nominal fixed spatial location

When the intention of a data variable is to contain only a single time series, the preferred encoding is a special case of the multidimensional array representation.

Example A9.2.3.1. A single timeseries.

```
dimensions:
   time = 100233 ;
   name_strlen = 23 ;

variables:
   float lon ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   float alt ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   char station_name(name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";

   double time(time) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
       time:missing_value = -999.9;
   float humidity(time) ;
       humidity:standard_name = “specific_humidity” ;
       humidity:coordinates = "time lat lon alt" ;
       humidity:_FillValue = -999.9;
   float temp(time) ;
       temp:standard_name = “air_temperature” ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt" ;
       temp:_FillValue = -999.9;

attributes:
       :featureType = "timeSeries";
```

While an idealized time series is defined at a single, stable point location, there are examples of time series, such as cabled ocean surface mooring measurements, in which the precise position of the observations varies slightly from a nominal fixed point. In the following example we show how the spatial positions of such a time series should be encoded in CF. Note that although this example shows only a single time series, the technique is applicable to all of the representations.

Example A9.2.3.2. A single timeseries with time-varying deviations from a nominal point spatial location

```
dimensions:
   time = 100233 ;
   name_strlen = 23 ;

variables:
   float lon ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
       lon:axis = “X”;
   float lat ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
       lat: axis = “Y” ;
   float precise_lon (time);
       precise_lon:standard_name = "longitude";
       precise_lon:long_name = "station longitude";
       precise_lon:units = "degrees_east";
   float precise_lat (time);
       precise_lat:standard_name = "latitude";
       precise_lat:long_name = "station latitude" ;
       precise_lat:units = "degrees_north" ;
   float alt ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   char station_name(name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";

   double time(time) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
       time:missing_value = -999.9;
   float humidity(time) ;
       humidity:standard_name = “specific_humidity” ;
       humidity:coordinates = "time lat lon alt precise_lon 
precise_lat" ;
       humidity:_FillValue = -999.9;
   float temp(time) ;
       temp:standard_name = “air_temperature” ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt precise_lon precise_lat " ;
       temp:_FillValue = -999.9;

attributes:
       :featureType = "timeSeries";
```

A9.2.4 Contiguous ragged array representation of timeSeries

When the time series have different lengths and the data values for entire time series are available to be written in a single operation, the contiguous ragged array representation is efficient.

Example A9.2.4.1. Timeseries of station data in the contiguous ragged array representation.

```
dimensions:
   station = 23 ;
   obs = 1234 ;

variables:
   float lon(station) ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat(station) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   float alt(station) ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   char station_name(station, name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";
   int station_info(station) ;
       station_info:long_name = "some kind of station info" ;
   int row_size(station) ;
       row_size:long_name = "number of observations for this station "
;
       row_size:sample_dimension = "obs" ;

   double time(obs) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float humidity(obs) ;
       humidity:standard_name = “specific_humidity” ;
       humidity:coordinates = "time lat lon alt" ;
       humidity:_FillValue = -999.9;
   float temp(obs) ;
       temp:standard_name = “air_temperature” ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt" ;
       temp:_FillValue = -999.9;

attributes:
       :featureType = "timeSeries";
```

The data humidity(o) and temp(o) are associated with the coordinate values time(o), lat(i), lon(i), and alt(i), where i indicates which time series. Time series i comprises the data elements from

```
   rowStart(i) to rowStart(i) + row_size(i) - 1

where

   rowStart(i) = 0 if i = 0
   rowStart(i) = rowStart(i-1) + row_size(i-1) if i > 0
```

The variable, `row_size`, is the count variable containing the length of each time series feature.   It is identified by having an attribute with name **sample_dimension** whose value is name of the sample dimension (`obs` in this example). The sample dimension could optionally be the netCDF unlimited dimension. The variable bearing the **sample_dimension** attribute must have the instance dimension (`station` in this example) as its single dimension, and must be of type integer.   This variable implicitly partitions into individual instances all variables that have the sample dimension. The auxiliary coordinate variables `lat`, `lon`, `alt` and `station_name` are station variables.

A9.2.5 Indexed ragged array representation of timeSeries

When time series with different lengths are written incrementally, the indexed ragged array representation is efficient.

Example A9.2.5.1. Timeseries of station data in the indexed ragged array representation.

```
dimensions:
   station = 23 ;
   obs = UNLIMITED ;

variables:
   float lon(station) ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat(station) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   float alt(station) ;
       alt:long_name = "vertical distance above the surface" ;
       alt:standard_name = "height" ;
       alt:units = "m";
       alt:positive = "up";
       alt:axis = "Z";
   char station_name(station, name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";
   int station_info(station) ;
       station_info:long_name = "some kind of station info" ;
   int stationIndex(obs) ;
       stationIndex:long_name = "which station this obs is for" ;
       stationIndex:instance_dimension= "station" ;
   double time(obs) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float humidity(obs) ;
       humidity:standard_name = “specific_humidity” ;
       humidity:coordinates = "time lat lon alt" ;
       humidity:_FillValue = -999.9;
   float temp(obs) ;
       temp:standard_name = “air_temperature” ;
       temp:units = "Celsius" ;
       temp:coordinates = "time lat lon alt" ;
       temp:_FillValue = -999.9;

attributes:
       :featureType = "timeSeries";
```

The humidity(o) and temp(o) data are associated with the coordinate values time(o), lat(i), lon(i), and alt(i), where i = stationIndex(o) is a zero-based index indicating which time series. Thus, time(0), humidity(0) and temp(0) belong to the element of the `station` dimension that is indicated by `stationIndex(0)`; time(1), humidity(1) and temp(1) belong to element `stationIndex(1)` of the `station` dimension, etc.

The variable, `stationIndex` , is identified as the index variable by having an attribute with name of **instance_dimension** whose value is the instance dimension (station in this example). The variable bearing the **instance_dimension** attribute must have the sample dimension (obs in this example) as its single dimension, and must be type integer. This variable implicitly assigns the station to each value of any variable having the sample dimension. The sample dimension need not be the netCDF unlimited dimension, though it commonly is.

A9.3 Annotated examples: Profile Data

A series of connected observations along a vertical line, like an atmospheric or ocean sounding, is called a profile. For each profile, there is a single time, lat and lon. A data variable may contain a collection of profile features. The instance dimension in the case of profiles specifies the number of profiles in the collection and is also referred to as the **profile dimension**. The instance variables, which have just this dimension, including latitude and longitude for example, are also referred to as **profile variables** and are considered to be information about the profiles. It is strongly recommended that there always be a profile variable (of any data type) with `cf_role` attribute `"profile_id"`, whose values uniquely identify the profiles. The profile variables may contain missing values. This allows one to reserve space for additional profiles that may be added at a later time, as discussed in section 9.6. All the representations described in section 9.1.3 can be used for profiles. The global attribute featureType=”profile” (case-insensitive) should be included if all data variables in the file contain profiles.

A9.3.1 Orthogonal multidimensional array representation of profiles

If the profile instances have the same number of elements and the vertical coordinate values are identical for all instances, you may use the orthogonal multidimensional array representation. This has either a one-dimensional coordinate variable, z(z), provided the vertical coordinate values are ordered monotonically, or a one-dimensional auxiliary coordinate variable, alt(o), where o is the element dimension. In the former case, listing the vertical coordinate variable in the **coordinates** attributes of the data variables is optional.

Example A9.3.1.1. Atmospheric sounding profiles for a common set of vertical coordinates stored in the orthogonal multidimensional array representation.

```
dimensions:
   z = 42 ;
   profile = 142 ;

variables:
   int profile(profile) ;
         profile:cf_role = "profile_id";
   double time(profile);
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(profile);
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(profile);
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;

   float z(z) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
       z:axis = "Z" ;

   float pressure(profile, z) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat z" ;

   float temperature(profile, z) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
       temperature:units = "Celsius" ;
       temperature:coordinates = "time lon lat z" ;

   float humidity(profile, z) ;
       humidity:standard_name = "relative_humidity" ;
```

```
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat z" ;

attributes:
   :featureType = "profile";
```

The pressure(i,o), temperature(i,o), and humidity(i,o) data for element o of profile i are associated with the coordinate values time(i), lat(i), and lon(i). The vertical coordinate for element o in each profile is altitude z(o). Either the instance (profile) or the element (z) dimension could be the netCDF unlimited dimension.

A9.3.2 Incomplete multidimensional array representation of profiles

If there are the same number of levels in each profile, but they do not have the same set of vertical coordinates, one can use the incomplete multidimensional array representation, which the vertical coordinate variable is two-dimensional e.g. replacing z(z) in Example A9.3.1 with alt(profile,z). This representation also allows one to have a variable number of elements in different profiles, at the cost of some wasted space. In that case, any unused elements of the data and auxiliary coordinate variables must contain missing data values (section 9.6).

A9.3.3 Single profile

When a single profile is stored in a file, there is no need for the profile dimension; the data arrays are one-dimensional. This is a special case of the orthogonal multidimensional array representation (9.3.1).

Example A9.3.3.1. Data from a single atmospheric sounding profile.

```
dimensions:
   z = 42 ;

variables:
   int profile ;
       profile:cf_role = "profile_id";

   double time;
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon;
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat;
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;

   float z(z) ;
```

```
        z:standard_name = “altitude”;
        z:long_name = "height above mean sea level" ;
        z:units = "km" ;
        z:positive = "up" ;
        z:axis = "Z" ;

    float pressure(z) ;
        pressure:standard_name = "air_pressure" ;
        pressure:long_name = "pressure level" ;
        pressure:units = "hPa" ;
        pressure:coordinates = "time lon lat z" ;

    float temperature(z) ;
        temperature:standard_name = "surface_temperature" ;
        temperature:long_name = "skin temperature" ;
        temperature:units = "Celsius" ;
        temperature:coordinates = "time lon lat z" ;

    float humidity(z) ;
        humidity:standard_name = "relative_humidity" ;
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat z" ;

attributes:
    :featureType = "profile";
```

The pressure(o), temperature(o), and humidity(o) data is associated with the coordinate values time, z(o), lat, and lon. The profile variables time, lat and lon, shown here as scalar, could alternatively be one-dimensional time(profile), lat(profile), lon(profile) if a size-one profile dimension were retained in the file.

A9.3.4 Contiguous ragged array representation of profiles

When the number of vertical levels for each profile varies, and one can control the order of writing, one can use the contiguous ragged array representation. The canonical use case for this is when rewriting raw data, and you expect that the common read pattern will be to read all the data from each profile.

Example A9.3.4.1. Atmospheric sounding profiles for a common set of vertical coordinates stored in the contiguous ragged array representation.

```
dimensions:
    obs = UNLIMITED ;
    profile = 142 ;

variables:
    int profile(profile) ;
        profile:cf_role = "profile_id";
    double time(profile);
        time:standard_name = "time";
        time:long_name = "time" ;
        time:units = "days since 1970-01-01 00:00:00" ;
    float lon(profile);
        lon:standard_name = "longitude";
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
    float lat(profile);
        lat:standard_name = "latitude";
        lat:long_name = "latitude" ;
        lat:units = "degrees_north" ;
     int rowSize(profile) ;
        rowSize:long_name = "number of obs for this profile " ;
        rowSize:sample_dimension = "obs" ;

    float z(obs) ;
        z:standard_name = “altitude”;
        z:long_name = "height above mean sea level" ;
        z:units = "km" ;
        z:positive = "up" ;
        z:axis = "Z" ;

    float pressure(obs) ;
        pressure:standard_name = "air_pressure" ;
        pressure:long_name = "pressure level" ;
        pressure:units = "hPa" ;
        pressure:coordinates = "time lon lat z" ;

    float temperature(obs) ;
        temperature:standard_name = "surface_temperature" ;
        temperature:long_name = "skin temperature" ;
        temperature:units = "Celsius" ;
        temperature:coordinates = "time lon lat z" ;

    float humidity(obs) ;
        humidity:standard_name = "relative_humidity" ;
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat z" ;

attributes:
    :featureType = "profile";
```

The pressure(o), temperature(o), and humidity(o) data is associated with the coordinate values time(i), z(o), lat(i), and lon(i), where i indicates which profile. All elements for one profile are contiguous along the sample dimension. The sample dimension (obs) may be the unlimited dimension or not. All variables that have the instance dimension (profile) as their single dimension are considered to be information about the profiles.

The count variable (row_size) contains the number of elements for each profile, and is identified by having an attribute with name "sample_dimension" whose value is the sample dimension being counted. It must have the profile dimension as its single dimension, and must be type integer. The elements are associated with the profile using the same algorithm as in A9.2.4.

A9.3.5 Indexed ragged array representation of profiles

When the number of vertical levels for each profile varies, and one cannot write them contiguously, one can use the indexed ragged array representation. The canonical use case is when writing real-time data streams that contain reports from many profiles, arriving randomly. If the sample dimension is the unlimited dimension, this allows data to be appended to the file.

Example A9.3.5.1. Atmospheric sounding profiles for a common set of vertical coordinates stored in the indexed ragged array representation.

```
dimensions:
   obs = UNLIMITED ;
   profile = 142 ;

variables:
   int profile(profile) ;
       profile:cf_name = "profile_id";
   double time(profile);
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(profile);
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(profile);
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;

   int parentIndex(obs) ;
       parentIndex:long_name = "index of profile " ;
       parentIndex:instance_dimension= "profile" ;

    float z(obs) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
       z:axis = "Z" ;

   float pressure(obs) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat z" ;

   float temperature(obs) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
       temperature:units = "Celsius" ;
       temperature:coordinates = "time lon lat z" ;
   float humidity(obs) ;
       humidity:standard_name = "relative_humidity" ;
       humidity:long_name = "relative humidity" ;
       humidity:units = "%" ;
       humidity:coordinates = "time lon lat z" ;

attributes:
   :featureType = "profile";
```

The pressure(o), temperature(o), and humidity(o) data are associated with the coordinate values time(i), z(o), lat(i), and lon(i), where i indicates which profile. The sample dimension (obs) may be the unlimited dimension or not. The profile index variable (parentIndex) is identified by having an attribute with name of "instance_dimension" whose value is the profile dimension name. It must have the sample dimension as its single dimension, and must be type integer. Each value in the profile index variable is the zero-based profile index that the element belongs to. The elements are associated with the profiles using the same algorithm as in A9.2.5**.**

A9.4 Annotated examples: Trajectory Data

Data may be taken along discrete paths through space, each path constituting a connected set of points called a trajectory, for example along a flight path, a ship path or the path of a parcel in a Lagrangian calculation. A data variable may contain a collection of trajectory features. The instance dimension in the case of trajectories specifies the number of trajectories in the collection and is also referred to as the **trajectory dimension**. The instance variables, which have just this dimension, are also referred to as **trajectory variables** and are considered to be information about the trajectories. It is strongly recommended that there always be a trajectory variable (of any data type) with the attribute `cf_role=”trajectory_id”` attribute, whose values uniquely identify the trajectories. The trajectory variables may contain missing values. This allows one to reserve space for additional trajectories that may be added at a later time, as discussed in section 9.6. All the representations described in section 9.3 can be used for trajectories. The global attribute featureType=”trajectory” (case-insensitive) should be included if all data variables in the file contain trajectories.

A9.4.1 Multidimensional array representation of trajectories

When storing multiple trajectories in the same file, and the number of elements in each trajectory is the same, one can use the multidimensional array representation. This representation also allows one to have a variable number of elements in different trajectories, at the cost of some wasted space. In that case, any unused elements of the data and auxiliary coordinate variables must contain missing data values (section 9.6).

Example A9.4.1.1. Trajectories recording atmospheric composition in the incomplete multidimensional array representation.

```
dimensions:
   obs = 1000 ;
   trajectory = 77 ;

variables:
   char trajectory(trajectory, name_strlen) ;
     trajectory:cf_role = "trajectory_id";
     trajectory:long_name = "trajectory name" ;
   int trajectory_info(trajectory) ;
       trajectory_info:long_name = "some kind of trajectory info"

   double time(trajectory, obs) ;
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(trajectory, obs) ;
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(trajectory, obs) ;
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;

   float z(trajectory, obs) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
       z:axis = "Z" ;

   float O3(trajectory, obs) ;
       O3:standard_name = “mass_fraction_of_ozone_in_air”;
       O3:long_name = "ozone concentration" ;
       O3:units = "1e-9" ;
       O3:coordinates = "time lon lat z" ;

   float NO3(trajectory, obs) ;
       NO3:standard_name = “mass_fraction_of_nitrate_radical_in_air”;
       NO3:long_name = "NO3 concentration" ;
       NO3:units = "1e-9" ;
       NO3:coordinates = "time lon lat z" ;

attributes:
   :featureType = "trajectory";
```

The NO3(i,o) and O3(i,o) data for element o of trajectory i are associated with the coordinate values time(i,o), lat(i,o), lon(i,o), and z(i,o). Either the instance (trajectory) or the element (obs) dimension could be the netCDF unlimited dimension. All variables that have trajectory as their only dimension are considered to be information about that trajectory.

If the trajectories all have the same set of times, the time auxiliary coordinate variable could be one-dimensional time(obs), or replaced by a one-dimensional coordinate variable time(time), where the size of the time dimension is now equal to the number of elements of each trajectory. In the latter case, listing the time coordinate variable in the coordinates attribute is optional.

A9.4.2 Single Trajectory

When a single trajectory is stored in the data variable, there is no need for the trajectory dimension and the arrays are one-dimensional. This is a special case of the multidimensional array representation.

Example A9.4.2.1. A single trajectory recording atmospheric composition.

```
dimensions:
   time = 42;

variables:
   char trajectory(name_strlen) ;
       trajectory:cf_role = "trajectory_id";

   double time(time) ;
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(time) ;
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(time) ;
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;
   float z(time) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
        z:axis = "Z" ;

   float O3(time) ;
       O3:standard_name = “mass_fraction_of_ozone_in_air”;
       O3:long_name = "ozone concentration" ;
       O3:units = "1e-9" ;
       O3:coordinates = "time lon lat z" ;

   float NO3(time) ;
       NO3:standard_name = “mass_fraction_of_nitrate_radical_in_air”;
       NO3:long_name = "NO3 concentration" ;
       NO3:units = "1e-9" ;
       NO3:coordinates = "time lon lat z" ;

attributes:
   :featureType = "trajectory";
```

The NO3(o) and O3(o) data are associated with the coordinate values time(o), z(o), lat(o), and lon(o). In this example, the time coordinate is ordered, so time values are contained in a coordinate variable i.e. time(time) and time is the element dimension. The time dimension may be unlimited or not.

Note that structurally this looks like unconnected point data as in example 9.5. The presence of the featureType = "trajectory" global attribute indicates that in fact the points are connected along a trajectory.

A9.4.3 Contiguous ragged array representation of trajectories

When the number of elements for each trajectory varies, and one can control the order of writing, one can use the contiguous ragged array representation. The canonical use case for this is when rewriting raw data, and you expect that the common read pattern will be to read all the data from each trajectory.

Example A9.4.3.1. Trajectories recording atmospheric composition in the contiguous ragged array representation.

```
dimensions:
   obs = 3443;
   trajectory = 77 ;

variables:
   char trajectory(trajectory, name_strlen) ;
         trajectory:cf_role = "trajectory_id";
   int rowSize(trajectory) ;
       rowSize:long_name = "number of obs for this trajectory " ;
       rowSize:sample_dimension = "obs" ;

   double time(obs) ;
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(obs) ;
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(obs) ;
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;
   float z(obs) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
        z:axis = "Z" ;

   float O3(obs) ;
       O3:standard_name = “mass_fraction_of_ozone_in_air”;
       O3:long_name = "ozone concentration" ;
```

```
        O3:units = "1e-9" ;
        O3:coordinates = "time lon lat z" ;

   float NO3(obs) ;
        NO3:standard_name = "mass_fraction_of_nitrate_radical_in_air";
        NO3:long_name = "NO3 concentration" ;
        NO3:units = "1e-9" ;
        NO3:coordinates = "time lon lat z" ;

attributes:
   :featureType = "trajectory";
```

The O3(o) and NO3(o) data are associated with the coordinate values time(o), lat(o), lon(o), and alt(o). All elements for one trajectory are contiguous along the sample dimension. The sample dimension (obs) may be the unlimited dimension or not. All variables that have the instance dimension (trajectory) as their single dimension are considered to be information about that trajectory.

The count variable (row_size) contains the number of elements for each trajectory, and is identified by having an attribute with name "sample_dimension" whose value is the sample dimension being counted. It must have the trajectory dimension as its single dimension, and must be type integer. The elements are associated with the trajectories using the same algorithm as in A9.2.4.

A9.4.4 Indexed ragged array representation of trajectories

When the number of elements at each trajectory vary, and the elements cannot be written in order, one can use the indexed ragged array representation. The canonical use case is when writing real-time data streams that contain reports from many trajectories. The data can be written as it arrives; if the sample dimension is the unlimited dimension, this allows data to be appended to the file.

Example A9.4.4.1. Trajectories recording atmospheric composition in the indexed ragged array representation.

```
dimensions:
   obs = UNLIMITED ;
   trajectory = 77 ;

variables:
   char trajectory(trajectory, name_strlen) ;
        trajectory:cf_role = "trajectory_id";

   int trajectory_index(obs) ;
        trajectory_index:long_name = "index of trajectory this obs
belongs to " ;
        trajectory_index:instance_dimension= "trajectory" ;
   double time(obs) ;
        time:standard_name = "time";
        time:long_name = "time" ;
        time:units = "days since 1970-01-01 00:00:00" ;
   float lon(obs) ;
        lon:standard_name = "longitude";
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
   float lat(obs) ;
        lat:standard_name = "latitude";
        lat:long_name = "latitude" ;
        lat:units = "degrees_north" ;
   float z(obs) ;
        z:standard_name = “altitude”;
        z:long_name = "height above mean sea level" ;
        z:units = "km" ;
        z:positive = "up" ;
        z:axis = "Z" ;

   float O3(obs) ;
        O3:standard_name = “mass_fraction_of_ozone_in_air”;
        O3:long_name = "ozone concentration" ;
        O3:units = "1e-9" ;
        O3:coordinates = "time lon lat z" ;

   float NO3(obs) ;
        NO3:standard_name = “mass_fraction_of_nitrate_radical_in_air”;
        NO3:long_name = "NO3 concentration" ;
        NO3:units = "1e-9" ;
        NO3:coordinates = "time lon lat z" ;

attributes:
   :featureType = "trajectory";
```

The O3(o) and NO3(o) data are associated with the coordinate values time(o), lat(o), lon(o), and alt(o). All elements for one trajectory will have the same trajectory index value. The sample dimension (obs) may be the unlimited dimension or not.

The index variable (trajectory_index) is identified by having an attribute with name of "instance_dimension" whose value is the trajectory dimension name. It must have the sample dimension as its single dimension, and must be type integer. Each value in the trajectory_index variable is the zero-based trajectory index that the element belongs to. The elements are associated with the trajectories using the same algorithm as in A9.2.5.

A9.5 Annotated examples: Time Series of Profiles

When profiles are taken repeatedly at a station, one gets a time series of profiles (see also section A9.2 for discussion of stations and time series). The resulting collection of profiles is called a timeSeriesProfile. A data variable may contain a collection of such timeSeriesProfile features, one feature per station. The instance dimension in the case of a timeSeriesProfile is also referred to as the **station dimension**. The instance variables, which have just this dimension, including latitude and longitude for example, are also referred to as **station variables** and are considered to contain information describing the stations. The station variables may contain missing values. This allows one to reserve space for additional stations that may be added at a later time, as discussed in section 9.6. In addition,

- It is strongly recommended that there should be a station variable (which may be of any type) with `cf_role` attribute `"timeseries_id"`, whose values uniquely identify the stations.
- It is recommended that there should be station variables with standard_name attributes "station_description", "surface_altitude" and “station_wmo_id” when applicable.

TimeSeriesProfiles are more complicated than timeSeries because there are two element dimensions (profile and vertical). Each time series has a number of profiles from different times as its elements, and each profile has a number of data from various levels as its elements. It is strongly recommended that there always be a variable (of any data type) with the profile dimension and the `cf_role` attribute `"profile_id"`, whose values uniquely identify the profiles.

A9.5.1 Multidimensional array representations of timeSeriesProfiles

When storing time series of profiles at multiple stations in the same data variable, if there are the same number of time points for all timeSeries, and the same number of vertical levels for every profile, one can use the multidimensional array representation:

Example A9.5.1.1. Time series of atmospheric sounding profiles from a set of locations stored in a multidimensional array representation.

```
dimensions:
   station = 22 ;
   profile = 3002 ;
   z = 42 ;

variables:
   float lon(station) ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat(station) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   char station_name(station, name_strlen) ;
       station_name:cf_role = "timeseries_id" ;
       station_name:long_name = "station name" ;
   int station_info(station) ;
       station_name:long_name = "some kind of station info" ;

   float alt(station, profile , z) ;
       alt:standard_name = “altitude”;
       alt:long_name = "height above mean sea level" ;
```

```
        alt:units = "km" ;
        alt:positive = "up" ;
         alt:axis = "Z" ;

    double time(station, profile ) ;
        time:standard_name = "time";
        time:long_name = "time of measurement" ;
        time:units = "days since 1970-01-01 00:00:00" ;
        time:missing_value = -999.9;

    float pressure(station, profile , z) ;
        pressure:standard_name = "air_pressure" ;
        pressure:long_name = "pressure level" ;
        pressure:units = "hPa" ;
        pressure:coordinates = "time lon lat alt" ;

    float temperature(station, profile , z) ;
        temperature:standard_name = "surface_temperature" ;
        temperature:long_name = "skin temperature" ;
        temperature:units = "Celsius" ;
        temperature:coordinates = "time lon lat alt" ;

    float humidity(station, profile , z) ;
        humidity:standard_name = "relative_humidity" ;
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat alt" ;

attributes:
 :featureType = "timeSeriesProfile";
```

The pressure(i,p,o), temperature(i,p,o), and humidity(i,p,o) data for element o of profile p at station i are associated with the coordinate values time(i,p), z(i,p,o), lat(i), and lon(i). Any of the three dimensions could be the netCDF unlimited dimension, if it might be useful to be able enlarge it.

If all of the profiles at any given station have the same set of vertical coordinates values, the vertical auxiliary coordinate variable could be dimensioned alt(station, z). If all the profiles have the same set of vertical coordinates, the vertical auxiliary coordinate variable could be one-dimensional alt(z), or replaced by a one-dimensional coordinate variable z(z), provided the values are ordered monotonically. In the latter case, listing the vertical coordinate variable in the coordinates attribute is optional.

If the profiles are taken at all stations at the same set of times, the time auxiliary coordinate variable could be one-dimensional time(profile), or replaced by a one-dimensional coordinate variable time(time), where the size of the time dimension is now equal to the number of profiles at each station. In the latter case, listing the time coordinate variable in the coordinates attribute is optional.

If there is only a single set of levels and a single set of times, the multidimensional array representation is formally orthogonal:

Example A9.5.1.2. Time series of atmospheric sounding profiles from a set of locations stored in an orthogonal multidimensional array representation.

```
dimensions:
  station = 10 ;  // measurement locations
  pressure = 11 ; // pressure levels
  time = UNLIMITED ;
variables:
  float humidity(time,pressure,station) ;
    humidity:standard_name = “specific_humidity” ;
    humidity:coordinates = "lat lon" ;
  double time(time) ;
  time:standard_name = "time";
    time:long_name = "time of measurement" ;
    time:units = "days since 1970-01-01 00:00:00" ;
  float lon(station) ;
    lon:long_name = "station longitude";
    lon:units = "degrees_east";
  float lat(station) ;
    lat:long_name = "station latitude" ;
    lat:units = "degrees_north" ;
  float pressure(pressure) ;
  pressure:standard_name = "air_pressure" ;
    pressure:long_name = "pressure" ;
    pressure:units = "hPa" ;
    pressure:axis = "Z" ;
```

`humidity(p,o,i)` is associated with the coordinate values `time(p)`, `pressure(o)`, `lat(i)`, and `lon(i)`. The number of profiles equals the number of times.

At the cost of some wasted space, the multidimensional array representation also allows one to have a variable number of profiles for different stations, and varying numbers of levels for different profiles. In these cases, any unused elements of the data and auxiliary coordinate variables must contain missing data values (section 9.6).

A9.5.2 Time series of profiles at a single station

If there is only one station in the data variable, there is no need for the station dimension:

Example A9.5.2.1. Time series of atmospheric sounding profiles from a single location stored in a multidimensional array representation.

```
dimensions:
   profile = 30 ;
   z = 42 ;

variables:
   float lon ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   char station_name(name_strlen) ;
       station_name:cf_role = "timeseries_id" ;
       station_name:long_name = "station name" ;
   int station_info;
       station_name:long_name = "some kind of station info" ;

   float alt(profile , z) ;
       alt:standard_name = “altitude”;
       alt:long_name = "height above mean sea level" ;
       alt:units = "km" ;
       alt:axis = "Z" ;
       alt:positive = "up" ;

   double time(profile ) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
       time:missing_value = -999.9;

   float pressure(profile , z) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat alt" ;

   float temperature(profile , z) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
       temperature:units = "Celsius" ;
       temperature:coordinates = "time lon lat alt" ;

   float humidity(profile , z) ;
       humidity:standard_name = "relative_humidity" ;
       humidity:long_name = "relative humidity" ;
       humidity:units = "%" ;
       humidity:coordinates = "time lon lat alt" ;

attributes:
 :featureType = "timeSeriesProfile";
```

The pressure(p,o), temperature(p,o), and humidity(p,o) data for element o of profile p are associated with the coordinate values time(p), alt(p,o), lat, and lon. If all the profiles have the same set of vertical coordinates, the vertical auxiliary coordinate variable could be one-dimensional alt(z), or replaced by a one-dimensional coordinate variable z(z), provided the values are ordered monotonically. In the latter case, listing the vertical coordinate variable in the coordinates attribute is optional.

A9.5.3 Ragged array representation of timeSeriesProfiles

When the number of profiles and levels for each station varies, one can use a ragged array representation. Each of the two element dimensions (time and vertical) could in principle be stored either contiguous or indexed, but this convention supports only one of the four possible choices. This uses the contiguous ragged array representation for each profile (9.3.3), and the indexed ragged array representation to organise the profiles into time series (9.3.4). The canonical use case is when writing real-time data streams that contain profiles from many stations, arriving randomly, with the data for each entire profile written all at once.

Example A9.5.3.1. Time series of atmospheric sounding profiles from a set of locations stored in a ragged array representation.

```
dimensions:
   obs = UNLIMITED ;
   profiles = 1420 ;
   stations = 42;

variables:
   float lon(station) ;
       lon:standard_name = "longitude";
       lon:long_name = "station longitude";
       lon:units = "degrees_east";
   float lat(station) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;
   float alt(station) ;
       alt:long_name = "altitude above MSL" ;
       alt:units = "m" ;
   char station_name(station, name_strlen) ;
       station_name:long_name = "station name" ;
       station_name:cf_role = "timeseries_id";
   int station_info(station) ;
       station_info:long_name = "some kind of station info" ;

   int profile(profile) ;
       profile:cf_role = "profile_id";
   double time(profile);
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   int station_index(profile) ;
       station_index:long_name = "which station this profile is for" ;
       station_index:instance_dimension = "station" ;
   int row_size(profile) ;
       row_size:long_name = "number of obs for this profile " ;
       row_size:sample_dimension = "obs" ;

   float z(obs) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:axis = "Z" ;
        z:positive = "up" ;
   float pressure(obs) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat z" ;

   float temperature(obs) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
       temperature:units = "Celsius" ;
       temperature:coordinates = "time lon lat z" ;

   float humidity(obs) ;
       humidity:standard_name = "relative_humidity" ;
       humidity:long_name = "relative humidity" ;
       humidity:units = "%" ;
       humidity:coordinates = "time lon lat z" ;

attributes:
   :featureType = "timeSeriesProfile";
```

The pressure(o), temperature(o), and humidity(o) data for element o of profile p at station i are associated with the coordinate values time(p), z(o), lat(i), and lon(i).

The index variable (station_index) is identified by having an attribute with name of instance_dimension whose value is the instance dimension name (station in this example). The index variable must have the profile dimension as its sole dimension, and must be type integer. Each value in the index variable is the zero-based station index that the profile belongs to i.e. profile p belongs to station i=station_index(p), as in section A9.2.5.

The count variable (row_size) contains the number of elements for each profile, which must be written contiguously. The count variable is identified by having an attribute with name sample_dimension whose value is the sample dimension (obs in this example) being counted. It must have the profile dimension as its sole dimension, and must be type integer. The number of elements in profile p is recorded in row_size(p), as in section A9.2.4. The sample dimension need not be the netCDF unlimited dimension, though it commonly is.

A9.6 Annotated examples: Trajectory of Profiles

When profiles are taken along a trajectory, one gets a collection of profiles called a trajectoryProfile. A data variable may contain a collection of such trajectoryProfile features, one feature per trajectory. The instance dimension in the case of a trajectoryProfile is also referred to as the **trajectory dimension**. The instance variables, which have just this dimension, are also referred to as **trajectory variables** and are considered to contain information describing the trajectories. The trajectory variables may contain missing values. This allows one to reserve space for additional trajectories that may be added at a later time, as discussed in section 9.6. TrajectoryProfiles are more complicated than trajectories because there are two element dimensions. Each trajectory has a number of profiles as its elements, and each profile has a number of data from various levels as its elements. It is strongly recommended that there always be a variable (of any data type) with the profile dimension and the `cf_role` attribute `"profile_id"`, whose values uniquely identify the profiles.

A9.6.1 Multidimensional array representation of trajectoryProfiles

If there are the same number of profiles for all trajectories, and the same number of vertical levels for every profile, one can use the multidimensional representation:

Example A9.6.1.1. Time series of atmospheric sounding profiles along a set of trajectories stored in a multidimensional array representation.

```
dimensions:
   trajectory = 22 ;
   profile = 33;
   z = 42 ;

variables:
   int trajectory (trajectory ) ;
       trajectory:cf_role = "trajectory_id" ;

   float lon(trajectory, profile) ;
       lon:standard_name = "longitude";
       lon:units = "degrees_east";
   float lat(trajectory, profile) ;
       lat:standard_name = "latitude";
       lat:long_name = "station latitude" ;
       lat:units = "degrees_north" ;

   float alt(trajectory, profile , z) ;
       alt:standard_name = “altitude”;
       alt:long_name = "height above mean sea level" ;
       alt:units = "km" ;
       alt:positive = "up" ;
       alt:axis = "Z" ;

   double time(trajectory, profile ) ;
       time:standard_name = "time";
       time:long_name = "time of measurement" ;
       time:units = "days since 1970-01-01 00:00:00" ;
       time:missing_value = -999.9;

   float pressure(trajectory, profile , z) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat alt" ;

   float temperature(trajectory, profile , z) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
```

```
        temperature:units = "Celsius" ;
        temperature:coordinates = "time lon lat alt" ;

    float humidity(trajectory, profile , z) ;
        humidity:standard_name = "relative_humidity" ;
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat alt" ;

attributes:
 :featureType = "trajectoryProfile";
```

The pressure(i,p,o), temperature(i,p,o), and humidity(i,p,o) data for element o of profile p along trajectory i are associated with the coordinate values time(i,p), alt(i,p,o), lat(i,p), and lon(i,p). Any of the three dimensions could be the netCDF unlimited dimension, if it might be useful to be able enlarge it.

If all of the profiles along any given trajectory have the same set of vertical coordinates values, the vertical auxiliary coordinate variable could be dimensioned alt(trajectory, z). If all the profiles have the same set of vertical coordinates, the vertical auxiliary coordinate variable could be one-dimensional alt(z), or replaced by a one-dimensional coordinate variable z(z), provided the values are ordered monotonically. In the latter case, listing the vertical coordinate variable in the coordinates attribute is optional.

If the profiles are taken along all the trajectories at the same set of times, the time auxiliary coordinate variable could be one-dimensional time(profile), or replaced by a one-dimensional coordinate variable time(time), where the size of the time dimension is now equal to the number of profiles along each trajectory. In the latter case, listing the time coordinate variable in the coordinates attribute is optional.

At the cost of some wasted space, the multidimensional array representation also allows one to have a variable number of profiles for different trajectories, and varying numbers of levels for different profiles. In these cases, any unused elements of the data and auxiliary coordinate variables must contain missing data values (section 9.6).

A9.6.2 Profiles along a single trajectory

If there is only one trajectory in the data variable, there is no need for the trajectory dimension:

Example A9.6.2.1. Time series of atmospheric sounding profiles along a trajectory stored in a multidimensional array representation.

```
dimensions:
   profile = 33;
   z = 42 ;

variables:
   int trajectory;
```

```
        trajectory:cf_role = "trajectory_id" ;

    float lon(profile) ;
        lon:standard_name = "longitude";
        lon:units = "degrees_east";
    float lat(profile) ;
        lat:standard_name = "latitude";
        lat:long_name = "station latitude" ;
        lat:units = "degrees_north" ;

    float alt(profile, z) ;
        alt:standard_name = “altitude”;
        alt:long_name = "height above mean sea level" ;
        alt:units = "km" ;
        alt:positive = "up" ;
         alt:axis = "Z" ;

    double time(profile ) ;
        time:standard_name = "time";
        time:long_name = "time of measurement" ;
        time:units = "days since 1970-01-01 00:00:00" ;
        time:missing_value = -999.9;

    float pressure(profile, z) ;
        pressure:standard_name = "air_pressure" ;
        pressure:long_name = "pressure level" ;
        pressure:units = "hPa" ;
        pressure:coordinates = "time lon lat alt" ;

    float temperature(profile, z) ;
        temperature:standard_name = "surface_temperature" ;
        temperature:long_name = "skin temperature" ;
        temperature:units = "Celsius" ;
        temperature:coordinates = "time lon lat alt" ;

    float humidity(profile, z) ;
        humidity:standard_name = "relative_humidity" ;
        humidity:long_name = "relative humidity" ;
        humidity:units = "%" ;
        humidity:coordinates = "time lon lat alt" ;

attributes:
 :featureType = "trajectoryProfile";
```

The pressure(p,o), temperature(p,o), and humidity(p,o) data for element o of profile p are associated with the coordinate values time(p), alt(p,o), lat(p), and lon(p). If all the profiles have the same set of vertical coordinates, the vertical auxiliary coordinate variable could be one-dimensional alt(z), or replaced by a one-dimensional coordinate variable z(z), provided the values are ordered monotonically. In the latter case, listing the vertical coordinate variable in the coordinates attribute is optional.

A9.6.3 Ragged array representation of trajectoryProfiles

When the number of profiles and levels for each trajectory varies, one can use a ragged array representation. Each of the two element dimensions (along a projectory, within a profile) could in principle be stored either contiguous or indexed, but this convention supports only one of the four possible choices. This uses the contiguous ragged array representation for each profile (9.3.3), and the indexed ragged array representation to organise the profiles into time series (9.3.4). The canonical use case is when writing real-time data streams that contain profiles from many trajectories, arriving randomly, with the data for each entire profile written all at once.

Example A9.6.3.1. Time series of atmospheric sounding profiles along a set of trajectories stored in a ragged array representation.

```
dimensions:
   obs = UNLIMITED ;
   profiles = 142 ;
   section = 3;

variables:
   int trajectory(trajectory) ;
       section:cf_role = "trajectory_id" ;

   double time(profile);
       time:standard_name = "time";
       time:long_name = "time" ;
       time:units = "days since 1970-01-01 00:00:00" ;
   float lon(profile);
       lon:standard_name = "longitude";
       lon:long_name = "longitude" ;
       lon:units = "degrees_east" ;
   float lat(profile);
       lat:standard_name = "latitude";
       lat:long_name = "latitude" ;
       lat:units = "degrees_north" ;
   int row_size(profile) ;
       row_size:long_name = "number of obs for this profile " ;
       row_size:sample_dimension = "obs" ;
   int trajectory_index(profile) ;
       trajectory_index:long_name = "which trajectory this profile is 
for" ;
       trajectory_index:instance_dimension= "trajectory" ;

    float z(obs) ;
       z:standard_name = “altitude”;
       z:long_name = "height above mean sea level" ;
       z:units = "km" ;
       z:positive = "up" ;
       z:axis = "Z" ;

   float pressure(obs) ;
       pressure:standard_name = "air_pressure" ;
       pressure:long_name = "pressure level" ;
       pressure:units = "hPa" ;
       pressure:coordinates = "time lon lat z" ;
   float temperature(obs) ;
       temperature:standard_name = "surface_temperature" ;
       temperature:long_name = "skin temperature" ;
       temperature:units = "Celsius" ;
       temperature:coordinates = "time lon lat z" ;

   float humidity(obs) ;
       humidity:standard_name = "relative_humidity" ;
       humidity:long_name = "relative humidity" ;
       humidity:units = "%" ;
       humidity:coordinates = "time lon lat z" ;

attributes:
   :featureType = "trajectoryProfile";
```

The pressure(o), temperature(o), and humidity(o) data for element o of profile p along trajectory i are associated with the coordinate values time(p), z(o), lat(p), and lon(p).

The index variable (trajectory_index) is identified by having an attribute with name of instance_dimension whose value is the instance dimension name (trajectory in this example). The index variable must have the profile dimension as its sole dimension, and must be type integer. Each value in the index variable is the zero-based trajectory index that the profile belongs to i.e. profile p belongs to trajectory i=trajectory_index(p), as in section A9.2.5.

The count variable (row_size) contains the number of elements for each profile, which must be written contiguously. The count variable is identified by having an attribute with name sample_dimension whose value is the sample dimension (obs in this example) being counted. It must have the profile dimension as its sole dimension, and must be type integer. The number of elements in profile p is recorded in row_size(p), as in section A9.2.4. The sample dimension need not be the netCDF unlimited dimension, though it commonly is.

Other changes wrt CF 1.5

New standard names to be added to the standard name table

- station_description : variable of character type containing a description of a time series station
- station_wmo_id : variable of character or integer type, containing the WMO identifier of an observing station

Changes to Appendix A

Amend the entries for _FillValue and missing_value to have use “C, D” i.e. coordinate or data, and add to their descriptions: “Not allowed for coordinate data except in the case of auxiliary coordinate variables in discrete sampling geometries.”

New attributes to be added:

Attribute	Type	Use	Links	Description
sample_dimension	N	D	**Section 9.1.3, “Representations of collections of features in data variables”**	An attribute which identifies a count variable and names the sample dimension to which it applies. The count variable indicates that the contiguous ragged array representation is being used for a collection of features.
instance_dimension	N	D	Section 9.1.3, “Representations of collections of features in data variables”	An attribute which identifies an index variable and names the instance dimension to which it applies. The index variable indicates that the indexed ragged array representation is being used for a collection of features.
featureType	C	G	Section 9.1.4, “The featureType attribute”	Specifies the type of discrete sampling geometry to which the data in the file belongs, and implies that all data variables in the file contain collections of features of that type.
cf_role	C	C	Section 9.1.5 “Coordinates and metadata” [Need new table that also documents gridspec cf_roles and others – when they are added to CF!]	Identifies the roles of variables that identify features in discrete sampling geometries

New section 4.5, “Discrete axis”

The spatiotemporal coordinates described in sections 4.1-4.4 are continuous variables, and other geophysical quantities may likewise serve as continuous coordinate variables, for instance density, temperature or radiation wavelength. By contrast, for some purposes there is a need for an axis of a data variable which indicates either an ordered list or an unordered collection, and does not correspond to any continuous coordinate variable. Consequently such an axis may be called “discrete”. A discrete axis has a dimension but might not have a coordinate variable. Instead, there might be one or more auxiliary coordinate variables with this dimension (see preamble to section 5). Following sections define various applications of discrete axes, for instance section 6.1.1 “Geographical regions”, section 7.3.3 “Statistics applying to portions of cells”, section 9.3 “Representation of collections of features in data variables”.

Changes to section 5

In section 5, third paragraph, change:

"The dimensions of an auxiliary coordinate variable must be a subset of the dimensions of the variable with which the coordinate is associated (an exception is label coordinates (Section 6.1, “Labels”) which contain a dimension for maximum string length)"

to

"The dimensions of an auxiliary coordinate variable must be a subset of the dimensions of the variable with which the coordinate is associated, with two exceptions. First, string-valued coordinates (see Section 6.1, “Labels”) have a dimension for maximum string length. Second, in the ragged array representations of data (Section 9, “Discrete sampling geometries”), special methods are needed to connect the data and coordinates,” Then begin a new paragraph with “We recommend ...”.

Replace the entire text of section 5.4 with “This section has been superseded by the treatment of time series as a type of discrete sampling geometry in section 9.

Replace the entire text of section 5.5 with “This section has been superseded by the treatment of trajectories as a type of discrete sampling geometry in section 9.

Changes to section 6.1

Replace the first two paragraphs:

The previous section contained several examples in which measurements from scattered sites were grouped using a **single** dimension. Coordinates of the site locations can be provided using auxiliary coordinate variables, but it is often desirable to identify measurement sites by name, or some other unique string. Other purposes for string identifiers are also described in Section 6.1.1, “Geographic Regions”, and Section 7.3.3, “Statistics applying to portions of cells”.

The list of string identifiers plays an analogous role to a coordinate variable, hence we have chosen to use the `coordinates` attribute to provide the name of the variable that contains the string array. An application processing the variables listed in the `coordinates` attribute can recognize a labeled axis by checking whether or not a given variable contains character data. If a character variable has only one dimension (the length of the string), it is regarded as a string-valued scalar coordinate variable, analogous to a numeric scalar coordinate variable (see Chapter 5, *Coordinate Systems* .)

with the following text:

Character strings can be used to provide a name or label for each element of an axis. This is particularly useful for discrete axes (section 4.5). For instance, if a data variable contains time series of observational data from a number of observing stations, it may be convenient to provide the names of the stations as labels for the elements of the station dimension (see section 9.2, *Time Series Data*). Example 9.1 illustrates another application for labels.

Character strings labelling the elements of an axis are regarded as string-valued auxiliary coordinate variables. The `coordinates` attribute of the data variable names the variable that contains the string array. An application processing the variables listed in the `coordinates` attribute can recognize a string-valued auxiliary coordinate variable because it contains an array of character data. The inner dimension (last dimension in CDL terms) is the maximum length of each string, and the other dimensions are axis dimensions. If a character variable has only one dimension (the maximum length of the string), it is regarded as a string-valued scalar coordinate variable, analogous to a numeric scalar coordinate variable (see 5.7, *Scalar Coordinate Variables*.)

Delete Example 6.1 (there are plenty of such examples in section 9). Renumber Example 6.2 as 6.1 and 6.3 as 6.2.