# [UC]$^2$ Data Standard "*Urban Climate under Change*"

## Version 1.5.2

22.11.2022

Dieter Scherer[1], Ute Fehrenbach[1], Tom Grassmann[1], Achim Holtmann[1], Fred Meier[1], Katharina Scherber[1], Dirk Pavlik[2], Thomas Höhne[3], Farah Kanani-Sühring[4], Björn Maronga[4], Felix Ament[5], Sabine Banzhaf[6], Ines Langer[6], Guido Halbig[7], Martin Kohler[8], Ronald Queck[9], Sebastian Stratbücker[10], Matthias Winkler[10], Robert Wegener[11], Matthias Zeeman[12]

[1]Technische Universität Berlin, Fachgebiet Klimatologie

[2]GEO-NET Umweltconsulting GmbH

[3]IDU IT+Umwelt GmbH

[4]Leibniz Universität Hannover, Institut für Meteorologie und Klimatologie

[5]Universität Hamburg, Meteorologisches Institut

[6]Freie Universität Berlin, Institut für Meteorologie

[7]Deutscher Wetterdienst

[8]Karlsruher Institut für Technologie, Institut für Meteorologie und Klimaforschung, Troposphärenforschung

[9]Technische Universität Dresden, Professur für Meteorologie

[10]Fraunhofer-Institut für Bauphysik

[11]Forschungszentrum Jülich GmbH, Institut für Energie- und Klimaforschung, IEK-8: Troposphäre

[12]Karlsruher Institut für Technologie, Institut für Meteorologie und Klimaforschung, Institut für Atmosphärische Umweltforschung

## Working Group "*Data Management*"

of the Research Programme

[UC]$^2$ Urban Climate Under Change
Stadtklima im Wandel

funded by the

Federal Ministry
of Education
and Research

## Foreword

This document was prepared by the working group "*Data Management*" within the framework of the research programme "*Urban Climate Under Change*" [UC]$^2$. The programme meets the challenge of providing large cities and urban regions with a scientifically sound, applicable set of tools for coping with problems associated with current and future weather and climate conditions, and air pollution. Already today, heavy rain and storms, heat and cold waves, droughts, as well as episodes with increased air pollution cause serious economic damage and health problems, including fatalities. In cities, concentration of population on the one hand, and modifications of atmospheric processes caused by urban structures on the other hand, result in a particularly high need for action in this respect. The regional consequences of global climate change will exacerbate these problems in the coming decades, and make additional efforts necessary.

A central objective of the programme is development, validation and application of the building-resolving model *PALM-4U* (*PALM for urban applications*; pronounced: *PALM for you*), designed to simulate entire cities such as Stuttgart or Berlin. *PALM-4U* is based on the Large Eddy Simulation (LES) model *PALM* (*Parallelized Large-Eddy Simulation Model*).

Urban climate models available so far are either to coarse to resolve micro-scale (buildings and street canyons) and local-scale (city quarters) processes, or they cover only smaller urban areas, and cannot be coupled to large-scale numerical models such as those used in weather forecasting or for regional climate projections. However, high-resolution models are of high importance for planning of measures to maintain and improve urban climate, for climate protection, to adapt to climate change, and to improve air pollution control.

In order for *PALM-4U* to be able to contribute to solve the aforementioned problems, its performance must first be tested and evaluated. This requires comprehensive data on weather, climate, and air quality in large cities (Module B). Unfortunately, such data are still only available to a limited extent, especially for multi-year or even multi-decadal long-term atmospheric observations in cities. Therefore, within the framework of the [UC]$^2$ programme, already available data are processed, and missing data are acquired through long-term measurements and intense observation periods. This requires improved concepts and analysis tools, and thus, their development is an important objective of the research programme. In addition, observational data alone should also be usable for specific applications.

Another indispensable requirement for *PALM-4U* is its practical usability. Therefore, on the one hand model results should allow reliable statements for a multitude of specific applications, and on the other hand demands on computer infrastructure and expertise of potential users should be as low as possible. Therefore, a further central goal of the programme is to integrate selected application examples and user groups directly into model development and observational data acquisition in order to ensure practical usability of the urban climate model, as well as of measurement concepts and analysis tools (Module C).

The [UC]$^2$ programme (uc2-program.org/en) consists of four bundle projects with a total of 30 sub-projects performing the above-mentioned tasks in the three modules: Module A: "*Model-based city planning and application in climate change*" (MOSAIK); Module B: "*Three-Dimensional Observation of Atmospheric Processes in Cities*" (3DO); Module C: "*Climate Models for Practice*" (KliMoPrax) and "*Review of practical and user serviceability of an urban climate model to foster climate proof urban development*" (UseUClim).

The working group "*Data Management*" was established to ensure efficient cooperation within the [UC]$^2$ programme by standardising the rules for use of data ([UC]$^2$ Data Policy), storage of data in a standardized format ([UC]$^2$ data standard), and provision of data in a data management system (DMS).

# Content

# 1. Introduction

## 1.1. Objectives

As part of the "*Urban Climate Under Change*" [UC]$^2$ programme (www.uc2-program.org/en), financed by the *Federal Ministry of Education and Research (BMBF)* since June 2016, a programme-wide data management system (DMS) was developed and made available for use by the project partners of the programme [UC]$^2$.

The use of a uniform data format and binding conventions for data and metadata is a prerequisite for effective and efficient management of all data in the DMS, but also for development and provision of innovative tools for analysis and visualization of complex and diverse data. These conventions are explained below and referred to as [UC]$^2$ data standard.

The [UC]$^2$ data standard considers both the requirements of the new urban climate model *PALM-4U*, which is developed by the partners of the MOSAIK bundle project in Module A, and different observational methods applied by the partners of the 3DO bundle project in Module B. It must be ensured that data and metadata are not only complete, but can also be exchanged and used between the two modules without further data conversion. Last but not least, project partners of Module C (KliMoPrax and UseUClim), as well as practice partners shall be able shall be able to use model and observational data for different applications without specific knowledge of details.

Although the [UC]$^2$ data standard was developed for specific requirements of the [UC]$^2$ programme, specifications have been made such that a transfer to other cities, research projects and applications is possible. In addition, data sets following the [UC]$^2$ data standard can be provided not only via the DMS but also via other data repositories.

## 1.2. Conventions

According to the requirements of the BMBF, all data sets that are used across subprojects are stored in the open, self-describing data format *NetCDF* (www.unidata.ucar.edu/software/netcdf) in the programme [UC]$^2$, since this data format is used worldwide and a large number of open, free interfaces and applications exist that handle *NetCDF*. A wide variety of data and metadata can be stored in *NetCD*F files, which are bindingly defined in the [UC]$^2$ data standard.

In addition to the choice of the *NetCDF* data format, the principles of the *NetCDF Climate and Forecast (*CF*) Metadata Conventions version 1.7* (cfconventions.org/Data/cf-conventions/cf-conventions-1.7/cf-conventions.html) are followed, which are thus part of the [UC]$^2$ data standard. In the following, these conventions are referred to as *CF-1.7*. This means that definitions of dimensions, coordinates, data variables and attributes must be *CF-1.7* compliant. Since *CF-1.7* is an extension of the conventions of the *Cooperative Ocean/Atmosphere Research Data Service* (*COARDS*) (ferret.pmel.noaa.gov/Ferret/documentation/coards-netcdf-conventions), they are implicitly considered in the [UC]$^2$ data standard in accordance to the explanations in *CF-1.7*. Some extensions *of CF-1.7* that are also used in the [UC]$^2$ data standard prevent complete backward compatibility to the *COARDS* conventions. To ensure maximum backward compatibility with freely available tools based on the *COARDS* conventions, the [UC]$^2$ data standard does not support all *CF-1.7* options.

For a large number of variables, *CF-1.7* defines uniform standard names and units (cfconventions.org/standard-names.html). Further variables are defined in an analogous manner by the [UC]$^2$ data standard, which also considers the standardizations of the BMBF programme *„High Definition Clouds and Precipitation for Advancing Climate Predictions*“; HD(CP)$^2$; hdcp2.eu; see also: *Standardized Atmospheric Measurement Data (SAMD) product standard* (icdc.cen.uni-hamburg.de/fileadmin/user_upload/samd_docs/samd_product_standard_v1.0.pdf).

The following description of the [UC]$^2$ data standard provides neither a complete description of the *NetCDF* format nor of the *COARDS* and *CF-1.7* conventions based on it, which is why it is recommended to refer to the above documents when creating and using [UC]$^2$ data sets. In addition, it is recommended to read the documentation of the *NetCDF* Application Programming Interface (*NetCDF* API) with utmost care. All examples are given in the Common Data Language (CDL) notation (see www.unidata.ucar.edu/software/netcdf/docs/netcdf_utilities_guide.html). When using other *NetCDF* APIs, it is important to note that there are different conventions for strings or for indexing multidimensional field variables (arrays). Therefore, when creating a *NetCDF* file a check is required whether the files created with a specific *NetCDF* API actually comply with the [UC]$^2$ data standard. In the *CDL* notation, the indexing of arrays begins with index 0 and the last index (furthest to the right) varies most rapidly (row-major ordering like, e.g., in the programming language C).

**Important**: In *NetCDF* files, attributes containing text are treated differently than variables containing text. The former is stored in the form of strings with *UTF-8* encoding, while the latter can only be stored in *NetCDF-3* files as character arrays (data type **`char`**). As of *NetCDF-4*, there is also the data type **`string`** for variables, but this is not supported in the [UC]$^2$ data standard to ensure compatibility with *PALM-4U*. It is therefore specified to use only *ASCII* characters in character arrays! All *NetCDF* files created as part of the [UC]$^2$ programme must be saved in *NetCDF-4* format.

The *NetCDF* data format allows data and metadata to be stored together in one file. Data are stored either as scalar quantities or in one-dimensional or multidimensional arrays. Arrays are defined by dimensions. Metadata are stored in additional variables and attributes. Each *NetCDF* file contains global attributes that relate to the entire data set. These are discussed in Section 2. Section 3 is dedicated to dimensions, coordinate variables and auxiliary coordinate variables, which unambiguously map data in time and space. Section 4 explains the data variables and their attributes, which describe the data in more detail. The **`featureType`** variants supported in the [UC]$^2$ data standard are described in Section 5. Section 6 explains how the DMS automatically generates file names from the contents of a *NetCDF* file. The Appendix contains some examples illustrating creation of *NetCDF* files in *CDL* notation.

The [UC]$^2$ data standard refers to four Tables A1 to A4 which are available as separate files as they need to be updated at shorter intervals to add new variables (Table A1), groups of variables (Table A2), institutions (Table A3) or sites (Table A4). Once published, table entries may not be changed or deleted, so that already created *NetCDF* files remain valid. Version numbering of the tables is therefore independent of the version numbering of the [UC]$^2$ data standard. The current versions of the [UC]$^2$ data standard and tables are made available on the web pages of the [UC]$^2$ programme (www.uc2-program.org/en). These tables are only binding for the [UC]$^2$ programme, but a similar approach is recommended if the [UC]$^2$ data standard is used for other purposes.

# 2. Global attributes

In the [UC]$^2$ data standard, global attributes are defined that are mandatory in each *NetCDF* file, which are listed in Tables 2.1, 2.2 and 2.3, and described in Sections 2.1, 2.2, and 2.3. The global attribute **`featureType`** plays a special role in the description of data structures, and is explained in Section 2.4. Section 2.5 and Table 2.4 describe optional global attributes with special meanings. Examples of how to use global attributes are also given in the Appendix.

Other global attributes not mandatory in the [UC]$^2$ data standard can be added and remain unchanged after uploading to the DMS. They must follow *CF-1.7* or conventions based on it. This applies in particular to upper and lower case of attributes (e.g. **`Conventions`** instead of **`conventions`**).

With exception of proper names and place names, all information in attributes should be in English to facilitate international use of the data.

## 2.1. General global attributes

The global attributes listed in Table 2.1 are characterizing a data set stored in a *NetCDF* file.

**Table 2.1** Mandatory general global attributes. Attributes important for generating file names (see Section 6) are marked with *, while ** indicates attributes whose values are subject to consistency check during upload.

| Name | Designation | Type | Value |
|---|---|---|---|
| **`title`** | Short title of data | S | Text |
| **`data_content`***,** | Title of content | S | Text (max. 16 characters; see Tables A1 and A2) |
| **`source`** | Type of data collection | S | Text (separation of several ways of collecting data with semicolon) |
| **`version`***,** | Version number | I | Integer (**`1`** to **`999`**) |
| **`Conventions`**** | *NetCDF* Conventions | S | **`"CF-1.7"`** |
| **`dependencies`**** | Dependencies | S | Text (separation of several file names with semicolon) |
| **`history`** | Information on data processing | S | Text |
| **`institution`**** | Institution of author or editor | S | Text (see Table A3) |
| **`acronym`***,** | Acronym of institution | S | Text (max. 12 characters; see Table A3) |
| **`author`**** | Author | S | Family name, first name, e-mail address (separation of several authors with semicolon) |
| **`contact_person`**** | Responsible person | S | Family name, first name, e-mail address (separation of several contact persons with semicolon) |
| **`references`** | Publications on data or methods | S | Text (separation of several references with semicolon) |
| **`comment`** | Optional additional information | S | Text |
| **`keywords`** | Keywords for searching in the DMS | S | Text (separation of several keywords with semicolon) |
| **`licence`**** | Licence information | S | Text |

**Type:** I: integer, S: *UTF-8* string

The following of the global attributes listed in Table 2.1 can be assigned individually, and are not subject to any further agreements: **`title`**, **`history`**, **`comment`**, **`keywords`**. If no entries are provided, then empty strings must be used. The other global attributes listed in Table 2.1 are explained below.

### `data_content`

This attribute describes the contents of the file in short form. This can either be the abbreviation for a single variable or a category for a data set consisting of several variables. For all data generated under the [UC]$^2$ programme, the abbreviations and category names are defined by the IOP city coordinators in consultation with the [UC]$^2$ working group "*Data Management*". The current list of abbreviations for individual variables can be found in Table A1, while the category names for variable groups are listed in Table A2. For many variables for which a value for the attribute **`standard_name`** (see Section 4.2) exists in *CF-1.7*, there is also an abbreviation of the *Atmospheric Model Intercomparison Project* (*AMIP*) defined by the *Intergovernmental Panel on Climate Change* (*IPCC*), which is used as the

value for **data_content** according to Table A1 when single variables are stored in a *NetCDF* file (see also www.wcrp-climate.org/images/modelling/WGCM/publications/IPCC_standard_output.pdf). For all other variables, a separate abbreviation has been introduced in Table A1. Analogous, separate groups of variables have been created in Table A2 if there is no suitable definition in *CF-1.7*.

**source**

The method of data acquisition is specified here. If original data are the result of a computer simulation, the model name and model version must be specified here. If data were collected using observations, the type of observation is specified (e.g. **"AWS", "mobile", "bicycle", "UAS", "LIDAR", "RADAR"**). It is recommended to use capital letters only for abbreviations.

**version**

The version number must be assigned in ascending order, starting with **1**, to ensure that differently formatted versions of a data set can be uniquely identified. Higher numbers indicate an updated, improved version of the record. The highest version number in the DMS represents the currently valid data set.

**Conventions**

The value **"CF-1.7"** must be specified as *NetCDF* convention.

**dependencies**

If data do not depend on other data, an empty string is specified here. Otherwise, the file names of the referenced data set must be specified according to Section 6. Several file names are separated by semicolons.

**institution**

Within the framework of the $[UC]^2$ programme, the $[UC]^2$ working group "*Data Management*" defines the full names of the institutions. The current list can be found in Table A3.

**acronym**

Within the framework of the $[UC]^2$ programme, the $[UC]^2$ working group "*Data Management*" defines the abbreviations for the respective institutions. The current list can be found in Table A3.

**author**

If one or more persons are authors of a data set, they are listed here according to the formatting requirements in Table 2.1, otherwise an empty string is given. The e-mail address is optional.

**contact_person**

One or more persons must be listed here as contact persons in accordance with the format specifications in Table 2.1. The e-mail address is optional.

**references**

One or more references in the bibliographic format of the Meteorologische Zeitschrift (see www.schweizerbart.de/journals/metz/instructions), as well as optionally the respective DOI or URL to the abstract or full text can be indicated. Otherwise an empty string is given.

**licence**

This attribute contains information on the license terms of data usage. If there are no such provisions, an empty string can be specified. For data collected under the $[UC]^2$ programme, a common data policy has been agreed, which is generally available at (www.uc2-program.org/uc2_data_policy.pdf). The

data policy defines seven different license types: MOSAIK Licence, 3DO Licence; KliMoPrax Licence, UseUClim Licence, [UC]$^2$ Restricted Licence, [UC]$^2$ Research Licence, and [UC]$^2$ Open Licence. If data are licensed under one of the above license types, the licence must be specified according or analogous to the following example:

**"[UC]2 Open Licence; see [UC]2 data policy available at www.uc2-program.org/uc2_data_policy.pdf"**.

**Important**: Although *UTF-8* supports the character "²" (exponent 2), it is not used here, because not all *NetCDF* APIs offer full support for *UTF-8* strings, and thus may display the license entry incorrectly!

## 2.2. Global attributes for time reference of data

The global attributes listed in Table 2.2 serve to reference data in time. All times are always given as Coordinated Universal Time (*UTC*).

**Table 2.2** Mandatory global attributes for the temporal reference of the data. Attributes that are important for generating the file name (see Section 6) are marked with *, while ** indicates attributes whose values are subject to a consistency check during upload.

| Name | Designation | Type | Value |
|---|---|---|---|
| **campaign***,** | Observation/model campaign | S | Text (max. 12 characters) |
| **origin_time***,** | Reference time (*UTC*) | S | **"YYYY-MM-DD hh:mm:ss +00"** |
| **creation_time**** | Time (*UTC*) of file generation | S | **"YYYY-MM-DD hh:mm:ss +00"** |

**Type:** S: *UTF-8* string

**campaign**

This attribute establishes the temporal relationship of data to measurement or model campaigns. Measurement campaigns in [UC]$^2$ are named according to a defined key. Data from long-term observations (LTO) are given the value **"LTO"**. Data from intense observation periods (IOP) are given the value **"IOP"** and a number with two digits, e.g. **"IOP01"**. Data from special observation periods (SOP) are given the value **"SOP"** and a number with two digits, e.g. **"SOP01"**.

Data sets from wind tunnel experiments or field measurements to be used as reference data for agreed model validation under the [UC]$^2$ programme shall be assigned the value **VALRxx**, where **xx** is a two-digit number defined by the working group *"Model Evaluation"*. Similarly, the results of model simulations for validation runs are called **VALMxx**. If validation runs that have already been simulated are re-simulated, they get version numbers:

**VALMxxvy** (xx=01-99, version ="v", version number y=1-9).

For wind tunnel data to be used for other purposes, it is recommended to mark them with the value **WTE**.

If data are not related to a campaign, a different string can be specified, whereby this string may only contain the ASCII characters **A-Z**, **a-z**, **"-"**, **"."**, and **"_"**, as well as digits **0** to **9**. Spaces or empty strings are not allowed since the value of **campaign** is used for generating a filename (see Section 6). For *PALM-4U* simulations without reference to a campaign, it is recommended to use **"PALM-4U"**.

**Important:** When generating the file name (see Section 6), minus signs **"-"**, which are required to separate the file name components, are replaced by underscores **"_"**; however, the attribute value itself remains unchanged!

**origin_time**

The reference time in *UTC* is specified for the specification of the relative times in the data (see Section 3.1). The specification of the time zone *UTC* by **"+00"** is mandatory. For data without explicit time reference, e.g. for digital elevation models or digital maps of land cover or other properties of surfaces, the time of data collection or map tracking should be specified here.

**creation_time**

Analogous to **origin_time,** the time of file creation in *UTC* is specified here. Specification of the time zone *UTC* by **"+00"** is mandatory.

## 2.3. Global attributes for spatial reference of data

The global attributes listed in Table 2.3 serve to spatially reference the data.

**Table 2.3** Mandatory global attributes for the spatial reference of the data. Attributes that are important for generating the file name (see Section 6) are marked with *****, while ****** indicates attributes whose values are subject to a consistency check during upload.

| Name | Designation | Type | Value |
|---|---|---|---|
| **location***,** | Location of data | S | Name of the region, city or municipality |
| **site***,** | Measurement site or model domain | S | Text (max. 12 characters) |
| **origin_x** | x-coordinate of reference point | F | Easting (= East value) as *UTM* coordinates in meters |
| **origin_y** | y-coordinate of reference point | F | Northing (= North value) as *UTM*-coordinates in meters |
| **origin_lon** | Longitude of reference point | F | In decimal degrees East |
| **origin_lat** | Latitude of reference point | F | In decimal degrees North |
| **origin_z** | Reference height | F | In meters |
| **rotation_angle** | Rotation angle of coordinate system | F | In degrees (**0.0** - **359.99**) |

**Type:** F: floating point number, S: *UTF-8* string

**location**

This attribute indicates the location (regions, cities or municipalities) to which the data set refers. For German cities and districts, official vehicle registration numbers (e.g. **"B"** for Berlin, **"HH"** for Hamburg, **"S"** for Stuttgart) may be used. For all data generated within the framework of the [UC]$^2$ programme, the values are defined by the [UC]$^2$ working group *"Data Management"*. The current list can be found in Table A4.

**site**

This attribute indicates the area in which observations have taken place, or contains the name of a model domain in short form. A site may be a single measuring station, or may include several stations close to each other. Within the framework of the [UC]$^2$ programme, the IOP city coordinators, in consultation with the [UC]$^2$ working group *"Data Management"*, define designations for sites. The current list can be found in Table A4.

**`origin_x`**

This attribute contains the x-coordinate (in metres) of a spatial reference point, which is the origin of a metric coordinate system (see Section 3.3). The mandatory metric coordinate system of the [UC]$^2$ data standard is the *Universal Transverse Mercator System (UTM)*. The parameters of the coordinate reference system (crs) for *UTM* eastings and northings are stored in the grid mapping variable **`crs`** (see Section 3.3).

**Important:** For data generated within the framework of the [UC]$^2$ programme, *UTM* eastings and northings must be specified according to the reference system ETRS89 with the GRS80 ellipsoid!

**`origin_y`**

This attribute contains the y-coordinate (in metres) of the spatial reference point, which is the origin of the metric coordinate system (see description of **`origin_x`** and Section 3.3).

**`origin_lon`**

The longitude of the spatial reference point is given in decimal degrees of Eastern longitude. The [UC]$^2$ data standard specifies that the reference system and ellipsoid for the specification of the geographical longitude and latitude must be identical to those of the UTM coordinate system (see Section 3.3).

**Important:** For data generated within the framework of the [UC]$^2$ programme, geographic longitudes and latitudes must be specified according to the reference system ETRS89 with the GRS80 ellipsoid!

**`origin_lat`**

The latitude of the spatial reference point is given in decimal degrees North latitude (see description of **`origin_lon`** and Section 3.3).

**`origin_z`**

The height of the spatial reference point is given in meters relative to a vertical reference system (vrs) (see Section 3.2). In the case of observation data, or when using a **`featureType`** variant, **`origin_z`** is uniformly set to the value **`0.0`**.

**`rotation_angle`**

This attribute contains the rotation angle of the metric coordinate system in degrees. The angle is mapped clockwise from the North. If the coordinate system is not rotated, i.e., if the y-axis points to North, the value **`0`** is given. An orientation of the y-axis to the East, South or West is displayed with a rotation angle of **`90`**, **`180`** or **`270`**.

## 2.4. The global attribute **`featureType`**

The *NetCDF* data format provides the ability to store data in scalar, one-dimensional or multidimensional variables. According to *CF-1.7*, the global attribute **`featureType`** can be used to define how time and space references are structured using dimensions, coordinate and auxiliary coordinate variables and data variables for different types of temporal and spatial organization of data.

If the data are available in a regular or irregular grid for a metric coordinate system, they are stored as multidimensional variables whose dimensions are given by **`time`**, **`z`**, **`y`**, and **`x`**, and specified by corresponding coordinate variables of the same name.

**Important: `featureType`** must not be defined for multidimensional data!

Dimensions must be listed in the following sequence when defining a variable: time (**`time`**), height above reference point **`(z)`**, distance from reference point in y-direction **`(y)`**, distance from reference point in x-direction **`(x)`**, i.e., data are stored in a *NetCDF* file in such a way that the index varies most

slowly for time. Only this sequence ensures that data are also compatible with *COARDS* and *PALM-4U*. For clarification, assume the definition of a data variable **ta(time,z,y,x)** containing air temperature for each time point **time[m]** of a time period, and for each point **(x[i],y[j],z[k])** of a three-dimensional metric grid. The value of the dimension **time** is the number of time points, and the values of the dimensions **z, y, x** are the respective number of grid points. The index **m** can have values between **0** and **time-1**. Similarly, the indices **i**, **j** and **k** can have values from **0** to **x-1**, **0** to **y-1**, and **0** to **z-1**.

If data sets are not to be stored as multidimensional data, the global attribute **featureType** must be specified according to *CF-1.7*.

**Important:** A *NetCDF* file can only contain data with the same **featureType**!

Since it is possible to save data of a certain **featureType** as multidimensional data or using another **featureType** variant, not all **featureType** variants defined in *CF-1.7* are supported in the [UC]$^2$ data standard. The supported **featureType** variants **"timeSeries"**, **"timeSeriesProfile"**, and **"trajectory"** are explained in detail in Sections 5.1, 5.2 and 5.3.

## 2.5. Optional global attributes

The optional global attributes listed in Table 2.4 serve to further describe a data set stored in a *NetCDF* file.

**Table 2.4** Optional global attributes. Attributes that are important for generating the file name (see Section 6) are marked with *****, while ****** indicates attributes whose values are subject to a consistency check during upload.

| Name | Designation | Type | Value |
|---|---|---|---|
| **data_specifier***,** | Specification of the data | S | Text (max. 16 characters) |

**Type:** S: *UTF-8* string

**data_specifier**

This attribute can be used to further specify the content of the data set. Only the characters **A-Z**, **a-z**, numbers **0** through **9** and **"_"** may be used. If this attribute is defined, its value will be part of the automatically generated file name (see Section 6). Therefore, this attribute can be used to generate files with distinct file names which otherwise would have the same automatically generated file name.

# 3. Dimensions, coordinate variables and auxiliary coordinate variables

Each data variable (see Section 4) is located in time and space as specified by time points, vertical coordinates (heights or depths) and horizontal coordinates, and can therefore be located exactly using these data. A spatial or temporal coordinate variable is a numerical, one-dimensional variable whose name is identical to its dimension and contains monotonically ascending or descending values at times or spatial coordinates.

**Important:** Coordinate variables must not have missing values!

Auxiliary coordinate variables are variables that contain coordinates but are not coordinate variables in the above sense (e.g. multidimensional arrays). Beside spatial or temporal coordinate variables and auxiliary coordinate variables, there are further auxiliary coordinate variables for unique allocation of data (e.g. for station or trajectory names; see Sections 5.1 and 5.2). In contrast to coordinate variables, auxiliary coordinate variables may have missing values, thus the attribute **`_FillValue`** is mandatory, and its value must be either **`-9999`** or **`-9999.0`**.

**Important:** Although *CF-1.7* allows it, the [UC]$^2$ data standard prohibits scalar coordinate variables and auxiliary coordinate variables to ensure compatibility with the *COARDS* conventions; i.e., values must always be entered as arrays, even if they have only one element!

Depending on the **`featureType`**, the definitions of the dimensions, coordinate variables and auxiliary coordinate variables differ according to *CF-1.7* (see examples in the Appendix).

For multidimensional data with simple time and space references, where the global attribute **`featureType`** must not be defined (see Section 2.4), the dimension **`time`**, **`z`**, **`y`**, and **`x`** as well as the one-dimensional coordinate variables of the same name **`time(time)`**, **`z(z)`**, **`y(y)`**, and **`x(x)`** produce the time and space references directly. Otherwise, depending on **`featureType`**, other dimensions are defined (see Sections 5.1, 5.2 and 5.3), and the variables **`time`**, **`z`**, **`y`**, and **`x`** are defined as one- or multidimensional coordinate variables or auxiliary coordinate variables of the dimensions **`ntime`**, **`nx`**, **`ny`**, **`nz`**, **`station`** or **`traj`**.

The following sections explain the corresponding conventions of the [UC]$^2$ data standard with respect to definition, attributes and values of coordinate and auxiliary coordinate variables.

## 3.1. Time

All times are given in seconds in the coordinate or auxiliary coordinate variable **`time`** relative to a reference time (Figure 3.1). The attributes **`long_name`** and **`standard_name`** must have the value **`"time"`**. The **`axis`** attribute is set to **`"T"`**.


**Figure 3.1** Definition of time data in the coordinate variable or auxiliary coordinate variable **`time`** (marked red) in the [UC]$^2$ data standard. All absolute times (here, **`origin_time`**; marked in green) refer to the proleptic Gregorian calendar and the coordinated world time (*UTC*).

In *NetCDF* files it is allowed to define the dimension **`time=UNLIMITED`** allowing later addition of newer data. Since uploading data to the DMS requires the data to be complete, this option is not supported in the [UC]$^2$ data standard.

Reference time is stored as an absolute time in the global attribute **`origin_time`** (see Section 2.2). All absolute times refer to the proleptic Gregorian calendar and the time zone *UTC*.

Time intervals of temporally aggregated quantities are specified in the **`time_bounds`** variable. The rules for the bounds mechanism described in section 3.6 apply. It is also specified that time intervals

are open at the bottom and closed at the top. All time specifications in **time** refer to the end of a time interval. For graphical representations of time-aggregated values it may be useful to refer to the middle of a time interval, but this must be done independently by the respective graphics program.

Aggregation methods of data are specified by the attribute **cell_methods**. The aggregation method can be omitted, if the values are point values (**"point"**). It is recommended to comprehensively use the specifications explained in *CF-1.7* in **cell_methods**. The attribute **bounds** and the variable **time_bounds** are omitted for temporally punctual data.

If there are only values in a file for one point in time or for a single time interval (e.g. daily means), the variable **time** must nevertheless be defined as an array with one element.

In principle, it is also possible to define the variable **time** as **float** or **double** to allow fractions of a second. However, it is recommended to use this option only when the use of integers (**int** or **long**) is not sufficient.

**Important:** Time specifications of *PALM-4U* simulations are always represented as floating-point numbers (data type **float**). Data to be read directly into *PALM-4U* must therefore use the data type **float** for the variable **time**!

The [UC]$^2$ data standard defines further conventions for storage of observational data generated within the [UC]$^2$ programme. For observational data from the intense observation periods (IOP), only the data of a single day may be stored per file. The start of the respective day is defined in **origin_time** as reference time. Thus, the variable **time** contains values up to a maximum of **86400**. Data from long-term observations (LTO) may only contain values for a single month. The beginning of the month is defined in **origin_time** as reference time, and the variable **time** therefore contains values up to a maximum of **31*86400**. The time step for LTO data that is to be provided via the DMS, is also set to a value of at least 1800 seconds (30 minutes).

**Important**: The above specifications do not apply to data sets used for model validation purposes, and thus extend over the entire validation period!

## 3.2. Vertical coordinates

Information on vertical coordinates (heights or depths) are listed in the coordinate or auxiliary coordinate variable **z** as relative values with respect to a reference height (**origin_z**) in meters (Figure 3.2). The attribute **long_name** must have the value **"height above origin"**. The attribute **axis** is set to **"Z"**.

**Important:** The attribute **standard_name** must be **"height_above_mean_sea_level"**, if **origin_z** has the value **0**, and must not be defined in any other case! In case of observational data or when using a **featureType** variant, the reference height in **origin_z** is uniformly set to the value **0**, and thus, **standard_name** must have the value **"height_above_mean_sea_level"**!

The reference height **origin_z** is a global attribute (see Section 2.3) given in meters, relative to a vertical reference system. Positive values for **z** are heights above the reference height. The attribute **positive** must therefore always be set to the value **"up"**. Negative values for **z** are therefore always below the reference height.

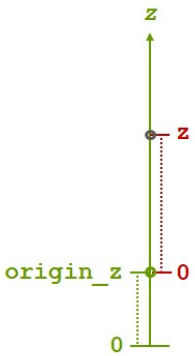


**Figure 3.2** Definition of the vertical coordinate or auxiliary coordinate variable **z** (marked red) in the [UC]$^2$ data standard. Absolute height data (here: **origin_z**; marked in green) refer to the *German height reference system DHHN2016* in case of data generated within the [UC]$^2$ programme. The vertical axis is always pointing upwards.

The vertical reference system is specified by the variable **vrs.** The vertical reference system specified in **vrs** must refer to an average sea level. The attribute **long_name** has the value **"vertical reference system"**, and the attribute **system_name** has the value **"DHHN2016"**. For all data generated within the framework of the [UC]$^2$ programme, the *German height reference system DHHN2016* (see de.wikipedia.org/wiki/Integrierter_Raumbezug_2016) is used as vertical reference system, and the reference height **origin_z** is in meters above standard elevation zero.

**Important:** The attribute **standard_name** must not be defined for **vrs**!

Height intervals of vertically aggregated quantities are specified by the variable **z_bounds.** The rules for the bounds mechanism described in Section 3.6 apply. Furthermore it is defined that height intervals are open at the bottom and closed at the top. Height values in **z** always refer to the centre of a height interval.

Aggregation methods of data are specified by the attribute **cell_methods**. The aggregation method can also be omitted, if the values are point values (**"point"**). It is recommended to comprehensively use the specifications explained in *CF-1.7* in **cell_methods**. For vertical point data, the attribute **bounds** and the variable **z_bounds** are omitted.

If there are only values for one height or for one height interval in a file, the variable **z** must still be defined as an array with an element.

In case of time series data (**featureType="timeSeries"**; see Section 5.1) or time series of vertical profiles (**featureType="timeSeriesProfile"**; see Section 5.2), the station height (height of the surface) must be specified in the variable **station_h**. The height above ground or the depth below ground is thus the difference between **z** and **station_h**.

For trajectories (**featureType="trajectory"**), a variable named **height** according to Section 5.3 must be used to specify the height above ground (**height** ≥ **0**) or the depth below ground **height** < **0**) for each location.

When specifying **station_h** and **height**, please note the following: For files of the **featureType timeSeries** or **timeSeriesProfile**, the variable **station_h** specifies the **height** of the surface above **origin_z** (**origin_z** = 0). The surface can also be a roof surface. So for measurements above roof **station_h** = ground level + roof level, for measurements above bare ground **station_h** = ground level. The height of the measurement points above **origin_z** is given in the variable **z**. The distance of the measurement from the surface is therefore **z** minus **station_h**. For a measurement in 5 m above the roof of a 10 m high building at a ground level of 30 m above sea level, **station_h** = 40 m and **z** = 45 m.

For **featureType trajectory files**, the variable **height** is used to indicate the height of the measurement above the surface. For example, for a backpack measurement, **height** could be constant at 1.5 m. For a trajectory, **height** could start at 0 m, rise to 200 m and end at 0 m again. The variable **z** again indicates the height of the measuring points above **origin_z** (**origin_z** = 0). The difference between **z** and **height** is the height of the surface. If, in the example of a trajectory, the height of the terrain was constant at 30 m above sea level, **z** would vary between 30 and 230 m.

## 3.3. Horizontal coordinates

According to *CF-1.7*, each *NetCD*F file must contain coordinate variables or auxiliary coordinate variables that specify longitude and latitude of data. For this, the variables **lon** and **lat** are used (see below and examples in the Appendix).

In the [UC]$^2$ data standard, however, horizontal coordinates are always represented by metric coordinates **x** and **y** in order to ensure compatibility with *PALM-4U*, and to facilitate spatial references to data. The attribute **long_name** must have the value **"distance to origin in x-direction"** or **"distance to origin in y-direction"**.

| **Important**: The attribute **standard_name** must not be specified for **x** and **y**! |
|---|

*PALM-4U* uses a horizontal regular metric grid, which can be optionally rotated to optimize model performance. For this reason, all model and observational data refer to a horizontal metric coordinate system as defined in Figure 3.3, where its origin is defined in a *UTM* coordinate system via the global attributes **origin_x** and **origin_y**. The rotation angle is specified in degrees by the global attribute **rotation_angle**; the values are between **0** and **359.99**. An unrotated horizontal metric coordinate system is characterized by the value **0** in **rotation_angle**, which indicates that the x-axis points to *UTM* grid east and the y-axis to *UTM* grid north. The value **90** indicates the x-axis pointing to *UTM* grid south and the y-axis to *UTM* grid east.

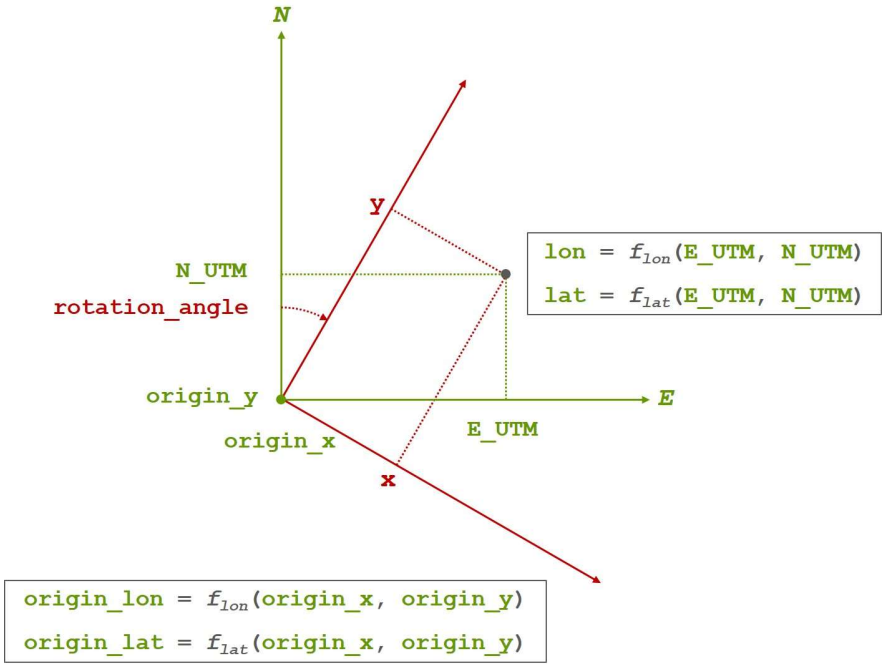


**Figure 3.3** Definition of a rotated horizontal metric coordinate system with the coordinates **x** and **y** (marked in red) in the [UC]$^2$ data standard. All *UTM* coordinates (here: **E_UTM** and **N_UTM**, **origin_x** and **origin_y**; marked in green), as well as the corresponding longitude and latitude (here: **lon** and **lat**, **origin_lon** and **origin_lat**; marked in green) refer to the *European Terrestrial Reference System 1989 (ETRS89)* with the ellipsoid of the Geodetic Reference System 1980 (GRS80). The functions

$f_{lon}$ and $f_{lat}$, used to convert *UTM* coordinates into geographical longitude and latitude, are freely available from program libraries such as PROJ.4.

**Important:** In the case of observational data acquired within the [UC]$^2$ programme, **rotation_angle** is uniformly set to the value **0**!

*UTM* coordinates can be calculated as eastings (**E_UTM**) and northings (**N_UTM**) from the coordinates **x** and **y** and the global attributes **origin_x, origin_y**, and **rotation_angle** using equations Eq. 3.1 and 3.2:

$$E\_UTM = origin\_x + \cos(rotation\_angle) \cdot x + \sin(rotation\_angle) \cdot y \quad \text{(Eq. 3.1)}$$

$$N\_UTM = origin\_y - \sin(rotation\_angle) \cdot x + \cos(rotation\_angle) \cdot y \quad \text{(Eq. 3.2)}$$

The [UC]$^2$ data standard requires the *UTM* coordinates **E_UTM** und **N_UTM** to be stored as auxiliary coordinate variables to avoid conversions according to Eq. 3.1 and 3.2 which may introduce uncertainties (rounding problems etc.), and to store further information about the underlying *UTM* map projection (see below). Standard names must be used according to the examples in the Appendix.

Eastings and northings are one-dimensional variables **E_UTM(x)** and **N_UTM(y)**, as long as none of the **featureType** variants is used. For rotated coordinate systems **E_UTM(y,x)** and **N_UTM(y,x)** must be defined as two-dimensional variables.

In the **featureType** variants, **x** and **y** are auxiliary coordinate variables, too. In **"timeSeries"** and **"timeSeriesProfile"**, horizontal auxiliary coordinate variables are one-dimensionally defined by the dimension **station** (**x(station)**, **y(station)**, **E_UTM(station)**, and **N_UTM(station)**). In **"trajectory"** they are two-dimensional variables of the dimensions **traj** and **ntime** (**x(traj,ntime)**, **y(traj,ntime)**, **E_UTM(traj,ntime)**, and **N_UTM(traj,ntime)**) (see examples in the Appendix).

According to *CF-1.7*, horizontal coordinates must also be provided as longitudes and latitudes. Conversion can be done with freely available tools (e.g. proj.org/apps/cs2cs.html or trac.osgeo.org/osgeo4w/). The corresponding auxiliary coordinate variables must be, analogously to **E_UTM** und **N_UTM**, defined as one- or two-dimensional variables (e.g. **lon(y,x)** and **lat(y,x)** or **lon(station)** and **lat(station)**), and filled with converted geographical longitudes and latitudes. The standard names according to the examples in the Appendix must be used.

The parameters of the coordinate reference system (crs) for *UTM* eastings and northings are stored in the grid mapping variable **crs**. For Germany, there are three *UTM* zones (31, 32 and 33) referring to central meridians 3, 9, and 15 degrees (*EPSG* projections 25831, 25832 and 25833; see e.g. spatialreference.org/ref/epsg/25831/). For urban regions in other countries, appropriate *UTM* zones must be chosen.

**Important:** The attribute **standard_name** must not be specified for **crs**!

Data variables are spatially referenced by absolute coordinates using the attributes **coordinates** and **grid_mapping**, and the underlying coordinate reference system (see also Section 4.1). Thus, each model grid point or observation location can be uniquely located both relative to the origin of the coordinate system (defined by **origin_x** and **origin_y**) via the coordinate variables **x** and **y**, as well as absolutely via the auxiliary coordinate variables for longitudes **lon** and latitudes **lat**, or *UTM* eastings **E_UTM** and northings **N_UTM**.

## 3.4. *PALM-4U* model grid

*PALM-4U* uses a three-dimensional, metric Arakawa-C model grid optimized for numerical models (see Figure 3.4). The model grid with the dimensions and coordinate variables **x**, **y** and **z** comprises $nx \cdot ny \cdot nz$ volume elements with the centres **(x[i],y[j],z[k])** at which the values of the

intensive thermodynamic state variables (potential temperature etc.) and the concentrations of the air constituents (specific humidity, concentrations of air pollutants etc.) are available. At the centres of the six surfaces bounding a volume element, the flow quantities (wind, sensible and latent heat flux densities, etc.) are stored. This requires the further dimensions **xu**, **yv**, and **zw**, as well as the coordinate variables **xu(xu)**, **yv(yv)**, and **zw(zw)**, which are each offset by half a grid spacing in x, y and z directions. The dimensions **xu**, **yv**, and **zw** are each one element larger than the corresponding dimensions **x**, **y**, and **z.**

Analogous to **E_UTM**, **N_UTM**, **lon** und **lat**, further auxiliary coordinate variables are required for the coordinate variables **xu** und **yv**:

Blue points on the x-axis: **Eu_UTM**, **Nu_UTM**, **lonu**, and **latu**.

Blue points on the y-axis: **Ev_UTM**, **Nv_UTM**, **lonv**, and **latv**.

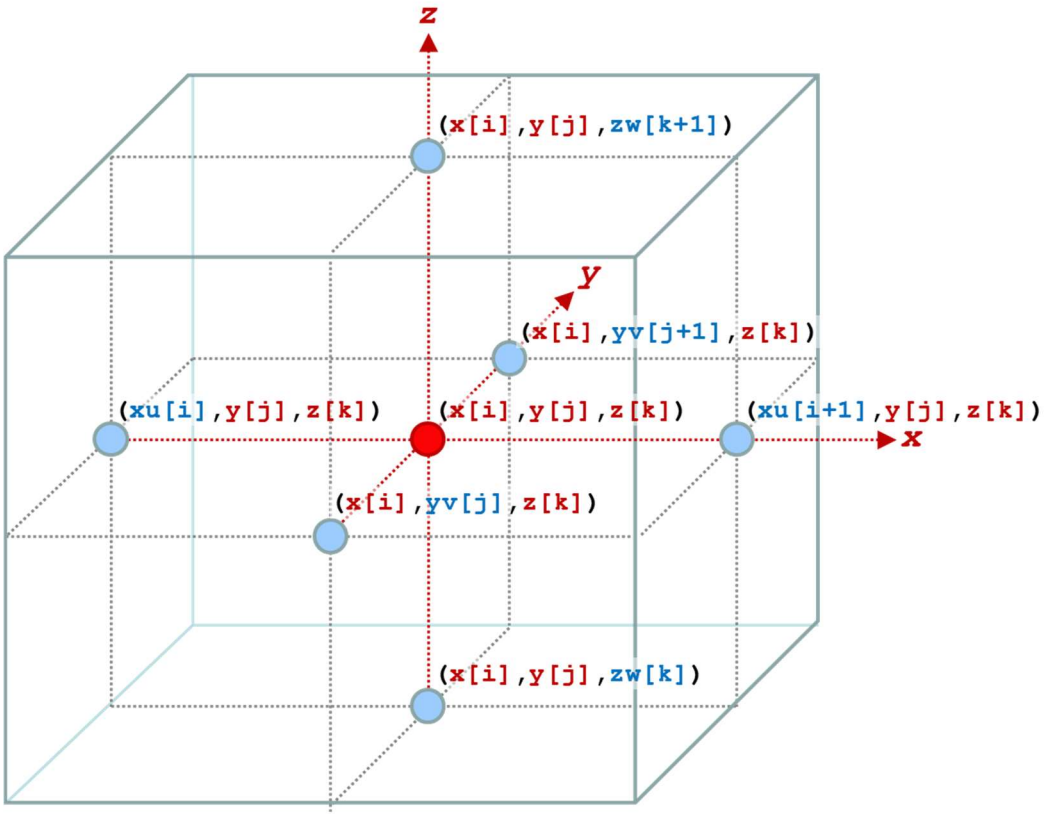


**Figure 3.4** Definition of a spatial three-dimensional, metric model grid for *PALM-4U* (Arakawa-C grid) with coordinate variables **x**, **y**, and **z** (marked in red) and the additional coordinate variables **xu**, **yv**, and **zw** (marked in blue), each offset by half a grid spacing. The red point lies in the centre of a solid element, while blue points represent the midpoints of the six surface elements bounding the solid element.

## 3.5. Surfaces

In order to process data for surfaces (e.g. walls and roofs of buildings), further information is required. Four different cases can be distinguished, each of them coming with specific requirements for additional information.

**Important:** Should data explicitly refer to specific times, then the coordinate variable **time** must be used. The dimension **time** must be the first dimension of the data variables.

**Digital terrain and surface models**

Digital terrain models (DTM) and digital surface models (DSM) are widely used for quantitative description of the topography of surfaces. These are two-dimensional grid data (raster data) in which heights of terrain or surfaces are available as data variables in the general form **zt(y,x)**. Height values apply either only to grid points (i.e., without specifying **cell_methods** for the horizontal dimensions), or are mean values (**cell_methods="area: mean"**) of surface elements (pixels).

In case of topographies with relief, the orientation of each surface element is specified by two angles. Exposure and slope inclination are usually specified for a DTM. In general, the orientation of a surface can be specified by the azimuth and zenith angles of the surface normal vector. For this purpose, the data variables **azimuth** and **zenith** are provided, which in case of DTM and DSM are defined as two- or three-dimensional fields **azimuth(y,x)** or **azimuth(time,y,x)**, and **zenith(y,x)** or **zenith(time,y,x)**. With this definition, the angle **azimuth** corresponds to exposure, and the angle **zenith** to slope inclination. The values of **zenith** are given in decimal degrees, and lie between **0** (surface is oriented upwards) and **180** (surface is oriented downwards). The data variable **azimuth** describes the horizontal orientation of the normal vector in decimal degrees. Values lie between **0** and **359.99**, whereby the following convention applies analogous to wind directions: North=**0**; East=**90**; South=**180**; West=**270**. For a surface oriented upwards or downwards, the value of azimuth is undefined and set to the pseudo value **0**. If the variables **zenith** and **azimuth** are missing, then uniform surfaces oriented upwards are assumed (so-called 'Lego-topography').

**Buildings in the *PALM-4U* model grid**

In the *PALM-4U* model grid (Section 3.4), any volume element with centre point **(x[i],y[j],z[k])** can be part of a building. Since *PALM-4U* also allows overhanging building parts, each of the six area elements delimiting a volume element can thus represent an interface between a building and the atmosphere. The normal vector of an interface surface is always oriented to the outside, i.e., either to the outdoor or indoor atmosphere.

Since in most cases, the number of the actual building surfaces is substantially smaller than the number of the theoretically possible surface elements, it is efficient for data storage reasons to introduce a further dimension **s**, as well as the coordinate variable of the same name **s** (surface), which specifies the number of building surfaces. For each of the **s** building surfaces, the auxiliary coordinate variables **xs(s)**, **ys(s)**, **zs(s)**, and **Es_UTM(s)**, **Ns_UTM(s)**, **lons(s)** and **lats(s)**, as well as the angles **zeniths(s)** and **azimuths(s)** are required. In addition, further data variables are required to characterize building surfaces.

**Pixel-based surface data**

Especially for thermal IR images taken in oblique view geometry, data are available as area integrals (mean values; **cell_methods="area: mean"**) for image elements (pixels), which are stored in two-dimensional variables. Two additional dimensions **row** and **col**, and the corresponding coordinate variables must be defined to specify image size and indexes of pixels. If coordinates **(x[i,j],y[i,j],z[i,j])** of midpoints of pixels are available, and thus the georeference is known, data can be stored together with coordinates in a *NetCDF* file.

**Important:** In future versions of the $[UC]^2$ data standard, coordinates **x**, **y**, and **z** must be provided!

The auxiliary coordinate variables **x(row,col)**, **y(row,col)**, **z(row,col)**, **lon(row,col)**, **lat(row,col)**, **E_UTM(row,col)** and **N_UTM(row,col)**, as well as the data variables, e.g. **ts(row,col)**, are thus two-dimensional arrays.

**Important:** When defining the dimensions **row** und **col**, it must be noted that the index for **col** varies the fastest, and thus specifies the number of image columns!

It is recommended to additionally specify the orientation for each pixel using the data variables **azimuth(row,col)** and **zenith(row,col)**.

If images are time series of temporally stationary surfaces, they can be stored in an analogous manner by using the additional dimension **time**. As always, the time dimension must be the first dimension, e.g. **ts(time,row,col)**. If positions or orientations of surfaces change over time, the spatial auxiliary coordinate variables are three-dimensional variables (**x(time,row,col)**, etc.), too.

As an alternative, pixel-based surface data of stationary surfaces (both single images and image time series) could be stored using the **featureType** variant **"timeSeries"** (see Section 5.1). However, since each pixel must then be listed as separate station, this alternative is inefficient for larger numbers of pixels and is thus not recommended.

**Surfaces arbitrarily oriented in space**

In the cases above, corner points of oriented surfaces are specified by two- or three-dimensional grids. For certain observational data, e.g. thermal image time series of arbitrarily oriented building surfaces and vegetation canopies, or in case of realistic 3D building models from GIS data, it is necessary to describe orientation of surfaces in general form to enable complex, realistic representation of surface.

As in the case of buildings in the *PALM*-4U model grid, a further dimension **s** and a coordinate variable of the same name are required to specify the number of surface elements.

Complex objects (e.g. buildings with inclined roofs that can currently not be represented by a *PALM-4U* model grid) are divided into elementary, interconnected triangles (Figure 3.5). Volumes are represented by their bounding surfaces. Depending on the number of corner points, each of these surfaces is divided into a number of triangles (Figure 3.5) whose number and sizes can be chosen such that surfaces can be represented with sufficient spatial differentiation.

A triangle is defined by three points with regard to its position and extent, as well as its orientation. The additional dimension **npt** has the fixed value **3**.

The sequence of the three points determines the orientation of a triangle (see Figure 3.5). Normal vectors are directed into the atmosphere; and thus, **azimuth** and **zenith** of normal vectors are not required (for simplification of data-usage, they can nonetheless be provided). For each of the **s** surfaces, the auxiliary coordinate variables **(xs(s,npt),ys(s,npt),zs(s,npt))**, as well as **Es_UTM(s,npt)**, **Ns_UTM(s,npt)**, **lons(s,npt)** and **lats(s,npt)** are required. In addition, further data variables for each surface element (e.g. building physics variables, surface temperature, etc.) characterize building surfaces, or are measured or simulated values.

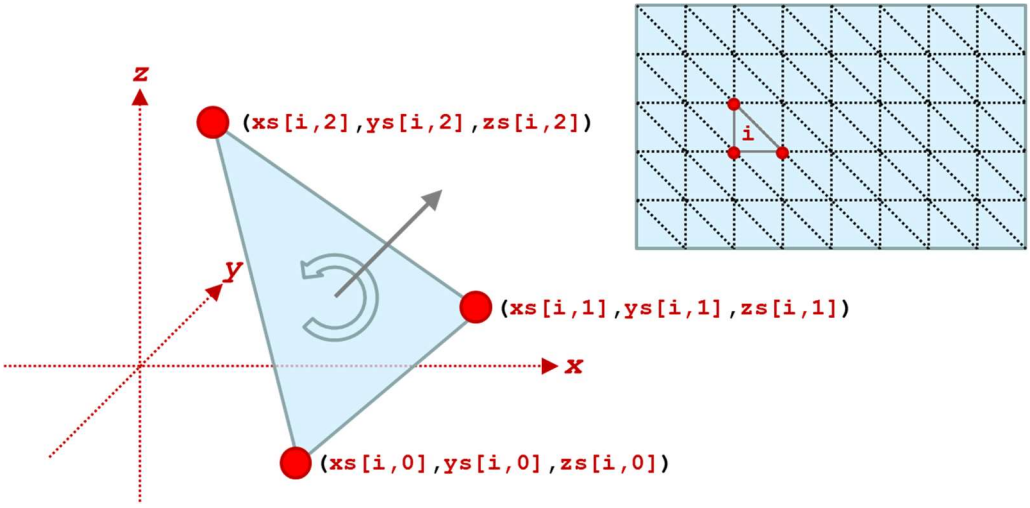


**Figure 3.5** Definition of an oriented triangle (index value **i**) by three points with the auxiliary coordinate variables **xs**, **ys**, and **zs**, whose sequence of the second index (index values **0** to **2**)

defines the orientation of the normal vector in a right-handed system. The normal vector of a triangle points to the atmosphere (exterior walls: outdoor atmosphere; interior walls: indoor atmosphere). Complex or extended surfaces can be represented by decomposition into a multitude of triangles with sufficient spatial differentiation.

## 3.6. Interval boundaries of coordinate and auxiliary coordinate variables

In case of coordinate and auxiliary coordinate variables that refer to intervals, these intervals are specified according to *CF-1.7* using the bounds mechanism. For this purpose, the coordinate or auxiliary coordinate variable is given the attribute **`bounds`**, which contains the name of the variable describing the interval boundaries. This variable is given the name **`xyz_bounds`**, where **`xyz`** is the name of the corresponding coordinate or auxiliary coordinate variable. The variable **`xyz_bounds`** has, in addition to the dimensions of the variable **`xyz`**, another dimension **`nv`** with the value 2. **`nv`** must be the dimension that varies the fastest, i.e. it must be on the far right. **`xyz_bounds`** must be defined in the same data type as **`xyz`**. The variable **`xyz_bounds`** must not have any attributes, because it automatically takes the values of the attributes of **`xyz`** (e.g. **`units`** or **`_FillValue`**).

# 4. Data variables

Data variables in the [UC]$^2$ data standard are variables that are not coordinate variables, auxiliary coordinate variables, or ancillary variables (see Section 4.3), and do not describe reference systems (**vrs, crs**). A data variable thus contains data and variable-specific metadata of quantities acquired by observations or model simulations.

In the [UC]$^2$ data standard, binding specifications are made for the designations of data variables, as is also the case in the data standard of the *SAMD* of the BMBF programme HD(CP)$^2$. Since *PALM-4U* uses fixed names for input and output variables, these must be used so that automated processing of observational and model data is possible. In addition, data variables are to be named as follows:

a) <***variable***>

b) <***variable***>_<***method***>

Here, <***variable***> is the corresponding value for the respective quantity according to Table A1. <***method***> specifies the temporal aggregation method according to *CF-1.7*. The value of **<method>** must be an abbreviation (see Table 4.1) of one of the permitted methods of the attribute **cell_methods**. An exception to this is the value **skew** (skewness; third moment of a distribution), for which there is currently no aggregation method defined in *CF-1.7*.

**Table 4.1** Allowed values (**<*method*>**) of temporal aggregation methods (**cell_methods**) according to *CF-1.7* for naming data variables.

| **<*method*>** | **cell_methods in *CF-1.7*** | **Specification of units** |
|---|---|---|
| max | "maximum" | Quantity (e.g. **"m s-1"**) |
| maxabs | "maximum_absolute_value" | Quantity (e.g. **"m s-1"**) |
| med | "median" | Quantity (e.g. **"m s-1"**) |
| mid | "mid_range" | Quantity (e.g. **"m s-1"**) |
| min | "minimum" | Quantity (e.g. **"m s-1"**) |
| minabs | "minimum_absolute_value" | Quantity (e.g. **"m s-1"**) |
| meanabs | "mean_absolute_value" | Quantity (e.g. **"m s-1"**) |
| meanupd | "mean_of_upper_decile" | Quantity (e.g. **"m s-1"**) |
| mode | "mode" | Quantity (e.g. **"m s-1"**) |
| range | "range" | Quantity (e.g. **"m s-1"**) |
| sigma | "standard_deviation" | Quantity (e.g. **"m s-1"**) |
| sumsqr | "sum_of_sqares" | Square of Quantity (e.g. **"m2 s-2"**) |
| var | "variance" | Square of Quantity (e.g. **"m2 s-2"**) |
| skew | | No dimension (**"1"**) |

Variant a) is used if a variable contains values that are instantaneous (**cell_methods**: **"point"**), averaged over time (**cell_methods**: **"mean"**), or accumulated over time (**cell_methods**: **"sum"**). Otherwise, variant b) is used. Thus, several quantities aggregated with different methods can be stored in a single *NetCDF* file.

**Important:** If a *NetCDF* file contains only one single data variable, the name of the variable according to Table A1 must also be used as the value for the global attribute **data_content**, i.e., without _<***method***>!

Variable-specific metadata is specified by variable attributes. The [UC]$^2$ data standard distinguishes between mandatory and optional attributes for data variables. Mandatory attributes are checked for consistency when files are uploaded to the DMS.

## 4.1. Mandatory attributes

Table 4.2 lists the mandatory attributes for data variables.

**Table 4.2** Mandatory attributes for data variables.

| Name | Explanation | Type | Value |
|---|---|---|---|
| **long_name** | Long name of data variable | S | According to Table A1 |
| **units** | Unit of data variable | S | Text; *UDUNITS*-conform unit |
| **_FillValue** | Fill value in case of missing or invalid data | N | **-9999 or -9999.0** |
| **coordinates** | All coordinate and auxiliary coordinate variables | S | e.g. **"lon lat E_UTM N_UTM x y z time"** |
| **grid_mapping** | Coordinate system reference-variable | S | **"crs"** |

**Type:** N: numerical (integer or floating point number), S: *UTF-8* string

### long_name

A long name is used to explain a variable in more detail, and is also used by some programs for automatically generated graphics. Within the framework of the [UC]² programme, the **long_name** value is assigned to each variable by the working group *"Data Management"*. The current list of defined variables and the corresponding values for **long_name** can be found in Table A1.

### units

According to *CF-1.7*, different units can be used for variables as long as they are UDUNITS-compliant and can be converted into each other (see cfconventions.org/standard-names.html). In Table A1, there is a proposed value for **units** for each variable defined in the [UC]$^2$ data standard. Optionally, an alternative unit can additionally be specified by the optional attribute **units_alt** (see Section 4.2).

### _FillValue

Missing values or invalid values (e.g. values outside a valid value range) are assigned the specified "fill value", which is uniformly set to the value **-9999** or rather **-9999.0**. The attribute **_FillValue** must have the same data type as the corresponding data variable!

### coordinates

This attribute contains the names of all coordinate and auxiliary coordinate variables to which the data variable refers. The value of this attribute differs depending on whether a *PALM-4U* model grid is used, whether oriented surfaces exist, or whether a **featureType** variant is used. The order or number of blank spaces is irrelevant.

Some examples are given below:

```
"lon  lat  E_UTM  N_UTM  x  y  z  time"
"lonu latu Eu_UTM Nu_UTM xu y  z  time"
"lonv latv Ev_UTM Nv_UTM x  yv z  time"
"lon  lat  E_UTM  N_UTM  x  y  zw time"
"lons lats Es_UTM Ns_UTM xs ys zs time"
"lon  lat  E_UTM  N_UTM  x  y  z  time station_name"
"lon  lat  E_UTM  N_UTM  x  y  z  time traj_name"
"lon  lat  E_UTM  N_UTM  x  y  z  time traj_name bands_pm".
```

**grid_mapping**

This attribute contains the assignment of the horizontal auxiliary coordinate variables to the coordinate of the reference system variable **crs** (see examples in the Appendix).

## 4.2. Optional attributes

*CF-1.7* lists a large number of additional attributes that describe the properties of variables in more detail. These can optionally be used, or are required if further properties of data have to be described. The latter applies in particular if data is aggregated in time or space, where the aggregation method must be specified according to *CF-1.7* by the attribute **cell_methods**, since otherwise, graphics or analysis programs may assume that data are valid only in points in time or space. Exceptions are temporal aggregation methods (e.g. **skew**), for which there are currently no values for **cell_methods** defined in *CF-1.7*.

Table 4.3 lists the optional attributes for variables additionally defined in the [UC]$^2$ data standard. It is strongly recommended to use all of the optional attributes listed in Table 4.3 if this is useful and applicable. In particular, the indication of the measurement instruments is required for all observation data. In addition, other optional attributes may be used in accordance with *CF-1.7*.

**Table 4.3** Optional attributes for data variables. The attribute **standard_name** is marked with *, which may only be defined if there is a *CF-1.7* standard name for the variable.

| Name | Explanation | Type | Value |
|---|---|---|---|
| **standard_name*** | Standard name of quantity | S | According to Table A1 (the attribute must not be defined, if there is no *CF-1.7* conform standard name!) |
| **units_alt** | Alternative unit | S | Text |
| **uncertainty_rel** | Relative uncertainty | F | Specification of the respective value in percent |
| **uncertainty_abs** | Absolute uncertainty | N | Specification of the unit of the quantity |
| **processing_level** | Processing level | I | Integer between **0** and **3** |
| **processing_info** | Processing method | S | Text |
| **instrument_name** | Designation of measurement instrument | S | Text |
| **instrument_nr** | Internal number of measurement instrument | S | Text: Numbers and letters |
| **instrument_sn** | Serial number of measuring instrument | S | Text: Numbers and letters |

**Type:** I: integer, N: numerical (integer or floating point number), S: *UTF-8* string

**standard_name**

This attribute is defined in *CF-1.7* for a variety of quantities (see cfconventions.org/standard-names.html), and must be used if available. Otherwise, the attribute may not be defined. The current list of defined variables and the corresponding values for **standard_name** can be found in Table A1.

**units_alt**

This attribute can be used to indicate an alternative unit complementary to the UDUNITS compliant unit (see mandatory attribute **units** in Section 4.1). Note that the units in **units** and **units_alt** must refer to identical numerical values of the data. As an example, the unit **"s-1"** can alternatively be given as **"Hz"**.

**uncertainty_rel**

This attribute specifies the uncertainty of the values in relative form (in percent). For example, a value of **20.0** (= 20 %) means that the uncertainty is at least one fifth of the respective value. If the attribute is omitted, **uncertainty_rel** is set to the value **0.0**, and the uncertainty is then determined by the value of **uncertainty_abs**. If the uncertainty calculated via **uncertainty_rel** is smaller than the value in **uncertainty_abs**, the latter value is used to determine the uncertainty. If both **uncertainty_rel** and **uncertainty_abs** are missing, then values are regarded as being exact.

**uncertainty_abs**

This attribute specifies the uncertainty of the values in absolute form. Thus, the absolute uncertainty is constant regardless of the respective value. If the attribute is omitted, **uncertainty_abs** is set to the value **0** (or **0.0**), and the uncertainty is then determined by the value of **uncertainty_rel**. If the uncertainty calculated via **uncertainty_rel** is smaller than the value in **uncertainty_abs**, the latter value is used to determine the uncertainty. If both **uncertainty_rel** and **uncertainty_abs** are missing, then values are regarded as being exact.

**Important:** This attribute must have the same data type as the variable!

**processing_level**

This attribute contains a numeric code that has the following meaning: **0**: raw data; **1**: calibrated data; **2**: automatically filtered or processed data; **3**: manually filtered or processed data. Observational data to be provided via the DMS must be quality-tested, and therefore at least be calibrated. Whenever possible, data shall be filtered or further processed using automatic procedures, e.g. by checking the valid value range of a variable and marking all values outside this range by the value specified in **_FillValue**.

**processing_info**

This attribute allows to make additional specifications for data processing. For example, information on data filtering or other quality assurance methods can be stored as free text.

**instrument_name**

Since each measured variable can be measured with different instruments, the name or designation of the measuring instrument (or sensor) is given here. This attribute does not apply to model simulations.

**instrument_number**

Several measuring instruments of the same type can, for example, be freely numbered or be designated with a combination of numbers and letters to distinguish them. The numbering may consist of numbers, letters or combinations thereof. This attribute does not apply to model simulations.

**instrument_sn**

To identify the measuring instrument, the serial number must be indicated as numbers, letters or combinations thereof. This attribute does not apply to model simulations.

## 4.3. Ancillary variables and flags

In *CF-1.7* it is possible to define ancillary variables with additional data (ancillary data) for a data variable, and to link them with the data variable. This is useful, for instance, if each value of a data variable is to be assigned its own additional data. One or more ancillary variables can be defined for a data variable. Conversely, an ancillary variable can be assigned to several data variables.

Ancillary variables must have the same dimensions as the data variable, but may have different data types. Data variables are linked to ancillary variables via the **ancillary_variables** attribute, in which a data variable lists the names of all ancillary variables assigned to it. Several ancillary variables are separated by spaces.

Special cases of ancillary variables are so-called flags. According to *CF-1.7*, ancillary variables that contain flags must use the **flag_meanings** attribute and one of the **flag_values** or **flag_masks** attributes. Although *CF-1.7* also allows simultaneous use of **flag_values** and **flag_masks**, this is not recommended as two flag variables can be used alternatively. In case of missing values flags may also be used to indicate the reason for missing data in the data variable. For this purpose, corresponding **flag_meanings** must be created. If the reason is unknown, the **flag_meanings** can also contain "NA". Therefore, flags should not have the attribute **_FillValue** themselves.

For ancillary variables, the $[UC]^2$ data standard defines a naming convention. On the one hand, a data variable can also act as an ancillary variable for other data variables. On the other hand, additional ancillary variables can be introduced. These are named as follows: **ancillary_xyz**. The value for **xyz** can be freely chosen. An example of data with ancillary variables and flags is given in Appendix A5 Example "Ancillary variables and flags".

## 4.4. Spectral data

Spectral data are data for which several values for a variable are available at every time and every place. A common example of spectral data is satellite data, where for each pixel there are radiance or reflectance values for different wavelength intervals (spectral channels or spectral bands). Another example are simultaneous measurements of number or mass concentrations of aerosol particles for different size classes with aerosol spectrometers.

Instead of storing the values of spectral data in separate variables, they can be combined into a single data variable. This requires a further dimension and a coordinate variable of the same name, which are named **bands_xyz** in the $[UC]^2$ data standard. The value for **xyz** can be chosen arbitrarily, which allows to save different spectra together in a single *NetCDF* file. For example, the dimensions and coordinate variables **bands_landsat_tm** and **bands_pm** could be used if satellite data of the Landsat Thematic Mapper (TM) sensor should be combined with aerosol spectra for fine particles in one file.

According to *CF-1.7*, the coordinate variable **bands_xyz** can have different numerical data types (**int**, **float**, etc.) and contains values characterizing the spectral bands. For example, values can be integer numbers of spectral bands, mean wavelengths of wavelength intervals, or mean diameters of size classes. In case of a continuous geophysical variable, the interval limits can be defined in a further variable using the attribute **bounds**, as e.g. in the case of the variable **time**. The rules for the bounds mechanism described in Section 3.6 apply.

If further information on spectral channels is available (e.g., numbers or names of spectral channels), these can be defined as auxiliary coordinate variables with the dimension **bands_xyz**. Such auxiliary

coordinate variables must be named **bands_xyz_abc** where the value of **abc** may be freely chosen. Via the **bounds** attribute these auxiliary coordinate variables can reference a variable with interval limits which has to be named **bands_xyz_abc_bounds**. The rules for the bounds mechanism described in Section 3.6 apply.

**Important:** As generally required, all coordinate and auxiliary coordinate variables must be specified in the attribute **coordinates** of the respective data variable!

Finally, data variables are defined as multidimensional arrays of all coordinate variables, where the dimension **bands_xyz** must be on the far left (slowest varying) side according to *CF-1.7*. An example of spectral data is given in Appendix A6 Example "Spectral data".

# 5. Supported **featureType** variants

As described in Section 2.4, *CF-1.7* defines different **featureType** variants to support specific discrete sampling geometries. The three **featureType** variants supported in the [UC]$^2$ data standard are described below.

## 5.1. Time series (**featureType="timeSeries"**)

Use of time series is intended when time series data to be stored in a *NetCDF* file are available at one or more locations within an area specified by the global attributes **location** and **site**.

The dimension **station** specifies the number of locations with different horizontal or vertical coordinates. The name **station** was chosen according to *CF-1.7*, but can also be used in a general way to distinguish several time series at different heights at a single site. In addition, data must not be exclusively data from observations, but could also result from numerical model simulations at so-called virtual stations.

**Important:** For all data generated within the framework of the [UC]$^2$ programme, the **featureType "timeSeries"** must be used even if there is only one (virtual) station! Then, the dimension **station** is set to the value **1**!

The auxiliary coordinate variable **station_name(station,max_name_len)** containing the station names must be defined according to the example in Appendix A2 Example "Time series". The dimension **max_name_len** is set to 32 characters, and thus limits the number of characters of the longest valid name. Station names can be freely chosen, whereby only ASCII characters may be used. According to *CF-1.7*, station names must be unique. The attribute **long_name** must have the value **"station name"**, the attribute **standard_name** the value **"platform_name"**, and the attribute **cf_role** must have the value **"timeseries_id"**.

Each data variable must state the additional auxiliary coordinate variable **station_name** in addition to the auxiliary coordinate variables for space and time in the **coordinates** attribute in order to assign data uniquely to stations. Moreover, a variable **station_h(station)** must be defined, which contains the respective height of the surface for each station (see Section 3.2 and Appendix A2 Example "Time series"). Surface may be either the terrain or a roof surface.

The dimension **ntime** defines the largest number of time points or time intervals that exist in one of the time series. The auxiliary coordinate variable **time** is a two-dimensional array **time(station,ntime)**, so that times can be defined independently for each time series. For time series of different lengths, the times of the shorter time series must be filled with fill values.

**Important:** In case of temporally aggregated quantities, the variable **time_bounds(station,ntime,nv)**, which specifies the limits of the time intervals, has, besides **station** and **ntime**, the additional dimension **nv** with the value **2** (see Appendix A2 Example "Time series")!

Point data can be regarded as time series which only refers to one point in time or one time interval. Then, the dimension **ntime** has the value **1**. Therefore, the **featureType "point"** is not supported in the [UC]$^2$ data standard.

**Important:** Time series are ideal for storing data along profiles that are not exclusively vertically aligned, and thus, the values of **x** or **y** may change at any height **z**!

## 5.2. Time series of vertical profiles (**featureType="timeSeriesProfile"**)

Use of time series of vertical profiles is intended when there are time series of vertical profiles at one or more locations for an area specified by the global attributes **location** and **site** to be stored in a single *NetCDF* file. All values of a vertical profile are valid for the same time or time interval.

The dimension **station** specifies the number of locations with different horizontal or vertical coordinates. The name **station** was chosen according to *CF-1.7*, but can also be used in a general way to distinguish several time series of vertical profiles at different heights (e.g. from two different, height-shifted profiler systems). In addition, data must not be exclusively data from observations, but could also result from numerical model simulations at so-called virtual stations.

**Important:** For all data generated within the framework of the [UC]$^2$ programme, it is specified that the **featureType "timeSeriesProfile"** must be used even if there is only one station! Then, the dimension **station** has the value **1**!

The auxiliary coordinate variable **station_name(station,max_name_len)** containing the station names must be defined according to the example in Appendix A3 Example "Time series of vertical profiles". The dimension **max_name_len** is set to 32 characters, and thus limits the number of characters of the longest valid name. Station names can be freely chosen, whereby only ASCII characters may be used. According to *CF-1.7*, station names must be unique. The attribute **long_name** must have the value **"station name"**, the attribute **standard_name** the value **"platform_name"**, and the attribute **cf_role** must have the value **"timeseries_id"**.

Each data variable must define the additional auxiliary coordinate variable **station_name** in addition to the auxiliary coordinate variables for space and time in the **coordinates** attribute in order to assign data uniquely to stations. In addition, a variable **station_h(station)** must be defined, which contains the respective height of the surface for each station (see Section 3.2 and Appendix A3 Example "Time series of vertical profiles"). The surface can be either the terrain or a roof surface.

The dimension **ntime** defines the largest number of time points or time intervals present in one of the time series of vertical profiles. The auxiliary coordinate variable **time** is a two-dimensional array **time(station,ntime)**, so that for each time series of vertical profiles the times at the individual stations can be defined independently of each other. For time series of different lengths, the times of the shorter time series must be filled with fill values.

**Important:** In the case of temporally aggregated quantities, the variable **time_bounds(station,ntime,nv)**, which specifies the limits of the time intervals, has the additional dimension **nv** with the value **2** in addition to **station** and **ntime**!

The dimension **nz** defines the largest number of heights or height intervals that exists in one of the time series of vertical profiles. The auxiliary coordinate variable **z** is a three-dimensional array **z(station,ntime,nz)**, so that for each time series of vertical profiles the heights at the different stations and times can be independently defined. In the case of time series of vertical profiles with different numbers of heights or height intervals, non-existing values must be filled with fill values.

Vertical profile data that are only available at one time can be regarded as time series of vertical profiles that refer to only one time or time interval. The dimension **ntime** then has the value **1**. Therefore, the **featureType "profile"** is not supported in the [UC]$^2$ data standard.

## 5.3. Trajectories (**featureType="trajectory"**)

Use of trajectories is intended if for an area specified by the global attributes **location** and **site** there is one or more series of locations at which data of one or more quantities are available for successive time points or time intervals to be stored in a single *NetCDF* file. Trajectories can be arbitrarily oriented in space.

**Important:** In contrast to a profile, for which the data are available at different heights simultaneously, the data along a trajectory are valid at different, strictly monotonously increasing points in time or time intervals, but may be available at the same points in space!

The dimension **traj** specifies the number of trajectories. With this **featureType** also time series of trajectories, i.e. repeated measurements or simulations along the same locations, can be stored. In this case, each repetition represents its own trajectory.

**Important:** For all data generated within the framework of the [UC]$^2$ programme, it is specified that the **featureType "trajectory"** must be used even if there is only one trajectory! Then, the dimension **traj** has the value **1**!

The auxiliary coordinate variable **traj_name(traj,max_name_len)** containing the trajectory names must be defined according to the example in Appendix A4 Example "Trajectories". The dimension **max_name_len** is set to 32 characters, and thus limits the number of characters of the longest valid name. The trajectory names can be freely selected, whereby only ASCII characters may be used. According to *CF-1.7*, trajectory names must be unique. The attribute **long_name** must have the value **"trajectory name"**, the attribute **standard_name** the value **"platform_name"**, and the attribute **cf_role** must have the value **"trajectory_id"**.

In addition, the variable **height** must be defined according to the example in Appendix A4 Example "Trajectories", which for each location contains the respective height above or below the surface (see Section 3.2). Height may either be a scalar variable (uniform height above or below the surface) or an array with the same structure as **z**. The height of the surface thus results from the difference of **z** and **height**.

The dimension **ntime** defines the largest number of time points or time intervals that are present in one of the trajectories. The auxiliary coordinate variable **time** is a two-dimensional array **time(traj,ntime)**, so that for each trajectory the number of times can be defined independently of each other. For trajectories with different number of times, non-existing values must be filled with fill values.

**Important:** In case of temporally aggregated values, the variable **time_bounds(traj,ntime,nv)**, which specifies the limits of the time intervals, has the additional dimension **nv** with the value **2** besides **traj** and **ntime**!

# 6. File names in the DMS

Data is provided in the DMS via *NetCDF* files, which are subject to a predefined naming convention. Any partner providing data via the DMS can name files as desired before uploading them to the DMS (however, the naming below is recommended). When uploading files to the DMS, the files are renamed after successful consistency check of the file contents, according to the following convention based on the values of the global attributes stored in the file:

***<campaign>-<location>-<site>-<acronym>-<data_content>-<origin_date>-<version>*.nc**

If the optional global attribute **data_specifier** is used, its value will be additionally included in the file name after ***<data_content>***:

***<campaign>-<location>-<site>-<acronym>-<data_content>*-<data_specifier>-<origin_date>-<version>.nc**

For example, a file name might look like this:

`IOP01-B-rothab1lawn-TUBklima-meteo-20170117-001.nc`

The following conversion rules apply:

1. The values for the file name components ***<campaign>***, ***<location>***, ***<site>***, ***<acronym>*** und ***<data_content>*** are taken directly from the values of the corresponding global attributes. Minus signs **"-"** within attributes are converted into underscores **"_"** when the file names are generated, since minus signs are required to separate the file name components.

2. The value for ***<origin_date>*** is derived from the global attribute **origin_time** and displayed in the format **"YYYYMMDD"** (**"YYYY"** = year with four digits; **"MM"** = month with two digits; **"DD"** = day with two digits).

3. The value for ***<version>*** is derived from the version number (1-999) contained in the global attribute of the same name and represented by three digits (**"001"** to **"999"**).

The names of files listed or downloaded by the DMS thus indicate their data content in a simple, human-readable manner.

When compiling a data set to be uploaded to the DMS in a *NetCDF* file, it must be considered that data that would lead to the same file name must be combined within a single file. To ensure distinct file names for similar data set the optional global attribute **data_specifier** can be used (see Section 2.5). If that attribute is not used, two potential types of problems may lead to file name conflicts:

1. If a ***<site>*** of a ***<location>*** comprises several individual stations operated by an institution (identified via ***<acronym>***) during a period of time (identified via ***<campaign>***) simultaneously (identified via ***<origin_date>***) , all affected station data with the same ***<data_content>*** must be stored in the same file. This means that a uniform **featureType** variant (e.g. **"timeSeries"**; see Section 5.1) must be used.

2. Data for different variables can either be saved individually in separate *NetCDF* files, or be combined within a single *NetCDF* file as data set. The respective values for the global attribute **data_content** can be found in Tables A1 and A2. When using collective names such as **"meteo"** or **"air"** (see Table A2), all variables belonging to this variable group must be stored in the same *NetCDF* file.

**Important**: If a *NetCDF* file with an incomplete data set has already been uploaded to the DMS such that extension of the data set would lead to a file name conflict, the entire data set must be recreated and uploaded to the DMS with a higher version number!

When compiling data sets, it should also be noted that saving individual variables in separate *NetCDF* files improves the identifiability of data via file names using ***<data_content>***, but also increases the number of *NetCDF* files. In addition, it is strongly discouraged to provide the same data in different *NetCDF* files (e.g. both individually and in a group, or multiple times in different groups).

# Appendix

## A1 Example "Multidimensional data"

```
// Global attributes:

:title          = "Multidimensional data example";
:data_content   = "ta";
:source         = "model";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de;
                   Fehrenbach, Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "";
:comment        = "demo data for hourly instantaneous values";
:keywords       = "air temperature; Berlin; gridded data";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at
                   www.uc2-program.org/uc2_data_policy.pdf";
:campaign       = "IOP01";
:origin_time    = "2017-01-17 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:05 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 42.;
:rotation_angle = 0.f;

dimensions:

  time = 24;
  z    = 20;
  y    = 50;
  x    = 40;

variables:

  int time(time);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-17 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(z);
    z:long_name = "height above origin";
    z:units     = "m";
    z:axis      = "Z";
    z:positive  = "up";

  int crs;
    crs:long_name                       = "coordinate reference system";
    crs:grid_mapping_name               = "transverse_mercator";
    crs:semi_major_axis                 = 6378137.f;
    crs:inverse_flattening              = 298.257222101;
```

```
    crs:longitude_of_prime_meridian      = 0.f;
    crs:longitude_of_central_meridian    = 15.f;
    crs:scale_factor_at_central_meridian = 0.9996f;
    crs:latitude_of_projection_origin    = 0.f;
    crs:false_easting                    = 500000.f;
    crs:false_northing                   = 0.f;
    crs:units                            = "m";
    crs:epsg_code                        = "EPSG:25833";

  float y(y);
    y:long_name = "distance to origin in y-direction";
    y:units     = "m";
    y:axis      = "Y";

  float x(x);
    x:long_name = "distance to origin in x-direction";
    x:units     = "m";
    x:axis      = "X";

  double N_UTM(y);
    N_UTM:long_name     = "northing";
    N_UTM:standard_name = "projection_y_coordinate";
    N_UTM:units         = "m";

  double E_UTM(x);
    E_UTM:long_name     = "easting";
    E_UTM:standard_name = "projection_x_coordinate";
    E_UTM:units         = "m";

  double lat(y, x);
    lat:long_name     = "latitude";
    lat:standard_name = "latitude";
    lat:units         = "degrees_north";

  double lon(y, x);
    lon:long_name     = "longitude";
    lon:standard_name = "longitude";
    lon:units         = "degrees_east";

  float ta(time, z, y, x);
    ta:long_name     = "air temperature";
    ta:standard_name = "air_temperature";
    ta:units         = "K";
    ta:_FillValue    = -9999.f;
    ta:coordinates   = "lon lat E_UTM N_UTM x y z time";
    ta:grid_mapping  = "crs";

data:
  time = 3600, 7200, …, 82800, 86400;

  vrs = _; // no value assigned

  z = 0.00000, 1.00000, …, 18.0000, 19.0000;

  crs = _; // no value assigned

  y = 0.00000, 1.00000, …, 48.0000, 49.0000;

  x = 0.00000, 1.00000, …, 38.0000, 39.0000;

  N_UTM = 5813054.0, 5813055.0, …, 5813102.0, 5813103.0;

  E_UTM = 385412.00, 385413.00, …, 385450.00, 385451.00;

  lat = 52.455631, 52.455631, …, 52.455639, 52.455639,
        52.455640, 52.455640, …, 52.455648, 52.455648,
                …,         …, …,         …,         …,
        52.456063, 52.456063, …, 52.456071, 52.456071,
        52.456072, 52.456072, …, 52.456080, 52.456080;
```

```
lon = 13.313614, 13.313629, …, 13.314173, 13.314188,
      13.313614, 13.313629, …, 13.314173, 13.314188,
               …,         …, …,         …,         …,
      13.313598, 13.313613, …, 13.314157, 13.314172,
      13.313597, 13.313612, …, 13.314156, 13.314171;

ta = 0.0226138, -0.0270021, …, 0.547265, 0.647889;
```

## A2 Example "Time series"

```
// Global attributes:

:title          = "featureType example for timeSeries data";
:data_content   = "ta";
:source         = "AWS";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de; Fehrenbach,
                   Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "JOHANNSON, E., S. THORSSON, R. EMMANUEL, E. KRÜGER 2014:
                   Instruments and methods in outdoor thermal comfort studies –
                   The need for standardization - Urban Climate 10, 346-366.
                   DOI: 10.1016/j.uclim.2013.12.002.
                   http://link.springer.com/article/10.1007/s12665-014-3068-1";
:comment        = "demo data for hourly mean values";
:keywords       = "air temperature; Berlin; time series data";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at
                   www.uc2-program.org/uc2_data_policy.pdf";
:campaign       = "IOP01";
:origin_time    = "2017-01-17 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:05 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 0.;
:rotation_angle = 0.f;
:featureType    = "timeSeries";

dimensions:

  station      = 2;
  ntime        = 24;
  nv           = 2;
  max_name_len = 32;

variables:

  char station_name(station, max_name_len);
    station_name:long_name     = "station name";
    station_name:standard_name = "platform_name";
    station_name:cf_role       = "timeseries_id";

  int time(station, ntime);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-17 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";
    time:_FillValue    = -9999;
    time:bounds        = "time_bounds";

  int time_bounds(station, ntime, nv);

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(station);
```

```
    z:long_name     = "height above origin";
    z:standard_name = "height_above_mean_sea_level"; // origin_z = 0.0!
    z:units         = "m";
    z:axis          = "Z";
    z:positive      = "up";

  float station_h(station);
    station_h:long_name     = "surface altitude";
    station_h:standard_name = "surface_altitude";
    station_h:units         = "m";

  int crs;
    crs:long_name                        = "coordinate reference system";
    crs:grid_mapping_name                = "transverse_mercator";
    crs:semi_major_axis                  = 6378137.f;
    crs:inverse_flattening               = 298.257222101;
    crs:longitude_of_prime_meridian      = 0.f;
    crs:longitude_of_central_meridian    = 15.f;
    crs:scale_factor_at_central_meridian = 0.9996f;
    crs:latitude_of_projection_origin    = 0.f;
    crs:false_easting                    = 500000.f;
    crs:false_northing                   = 0.f;
    crs:units                            = "m";
    crs:epsg_code                        = "EPSG:25833";

  float y(station);
    y:long_name = "distance to origin in y-direction";
    y:units     = "m";
    y:axis      = "Y";

  float x(station);
    x:long_name = "distance to origin in x-direction";
    x:units     = "m";
    x:axis      = "X";

  double N_UTM(station);
    N_UTM:long_name     = "northing";
    N_UTM:standard_name = "projection_y_coordinate";
    N_UTM:units         = "m";

  double E_UTM(station);
    E_UTM:long_name     = "easting";
    E_UTM:standard_name = "projection_x_coordinate";
    E_UTM:units         = "m";

  double lat(station);
    lat:long_name     = "latitude";
    lat:standard_name = "latitude";
    lat:units         = "degrees_north";

  double lon(station);
    lon:long_name     = "longitude";
    lon:standard_name = "longitude";
    lon:units         = "degrees_east";

  float ta(station, ntime);
    ta:long_name        = "air temperature";
    ta:standard_name    = "air_temperature";
    ta:units            = "K";
    ta:_FillValue       = -9999.f;
    ta:coordinates      = "lon lat E_UTM N_UTM x y z time station_name";
    ta:grid_mapping     = "crs";
    ta:cell_methods     = "time: mean";
    ta:uncertainty_abs  = 0.2f;
    ta:processing_level = 1;
    ta:processing_info  = "Calibrated data";

data:
  station_name = "AWS 1", "AWS 2";
```

```
time = 3600, 7200, …, 82800, 86400,  // last second of hours
       3600, 7200, …, -9999, -9999; // shorter time series at 2nd station

time_bounds =     0,  3600, // boundaries of hours
               3600,  7200,
                  …,     …,
              79200, 82800,
              82800, 86400,

                  0,  3600,
               3600,  7200,
                  …,     …,
              -9999, -9999,
              -9999, -9999; // shorter time series at 2nd station

vrs = _; // no value assigned

z = 44.0000, 44.5000; // 2 m above ground

station_h = 42.0000, 42.5000;

crs = _; // no value assigned

y = 0.00000, -1.00000;

x = 0.00000, 2.00000;

N_UTM = 5813054.0, 5813053.0;

E_UTM = 385412.00, 385414.00;

lat = 52.455631, 52.455623;

lon = 13.313614, 13.313644;

ta =   0.152663, -0.630299, …, 0.674597, 0.970390,
     -0.0636457, -0.211619, …, -9999.00, -9999.00;
```

## A3 Example "Time series of vertical profiles"

```
// Global attributes:

:title          = "featureType example for timeSeriesProfile data";
:data_content   = "ta";
:source         = "AWS";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de; Fehrenbach,
                   Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "";
:comment        = "demo data for daily mean vertical profile values";
:keywords       = "air temperature; Berlin; vertical profile";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at
                   www.uc2-program.org/uc2_data_policy.pdf";
:campaign       = "IOP01";
:origin_time    = "2017-01-17 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:06 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 0.;
:rotation_angle = 0.f;
:featureType    = "timeSeriesProfile";

dimensions:

  station      = 1;
  ntime        = 1;
  nv           = 2;
  max_name_len = 32;
  nz           = 3;

variables:

  char station_name(station, max_name_len);
    station_name:long_name     = "station name";
    station_name:standard_name = "platform_name";
    station_name:cf_role       = "timeseries_id";

  int time(station, ntime);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-17 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";
    time:bounds        = "time_bounds";

  int time_bounds(station, ntime, nv);

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(station, ntime, nz);
    z:long_name     = "height above origin";
    z:standard_name = "height_above_mean_sea_level"; // origin_z = 0.0!
    z:units         = "m";
    z:axis          = "Z";
```

```
    z:positive      = "up";
    z:bounds        = "z_bounds";

  float z_bounds(station, ntime, nz, nv);

  float station_h(station);
    station_h:long_name     = "surface altitude";
    station_h:standard_name = "surface_altitude";
    station_h:units         = "m";

  int crs;
    crs:long_name                        = "coordinate reference system";
    crs:grid_mapping_name                = "transverse_mercator";
    crs:semi_major_axis                  = 6378137.f;
    crs:inverse_flattening               = 298.257222101;
    crs:longitude_of_prime_meridian      = 0.f;
    crs:longitude_of_central_meridian    = 15.f;
    crs:scale_factor_at_central_meridian = 0.9996f;
    crs:latitude_of_projection_origin    = 0.f;
    crs:false_easting                    = 500000.f;
    crs:false_northing                   = 0.f;
    crs:units                            = "m";
    crs:epsg_code                        = "EPSG:25833";

  float y(station);
    y:long_name = "distance to origin in y-direction";
    y:units     = "m";
    y:axis      = "Y";

  float x(station);
    x:long_name = "distance to origin in x-direction";
    x:units     = "m";
    x:axis      = "X";

  double N_UTM(station);
    N_UTM:long_name     = "northing";
    N_UTM:standard_name = "projection_y_coordinate";
    N_UTM:units         = "m";

  double E_UTM(station);
    E_UTM:long_name     = "easting";
    E_UTM:standard_name = "projection_x_coordinate";
    E_UTM:units         = "m";

  double lat(station);
    lat:long_name     = "latitude";
    lat:standard_name = "latitude";
    lat:units         = "degrees_north";

  double lon(station);
    lon:long_name     = "longitude";
    lon:standard_name = "longitude";
    lon:units         = "degrees_east";

  float ta(station, ntime, nz);
    ta:long_name     = "air temperature";
    ta:standard_name = "air_temperature";
    ta:units         = "K";
    ta:_FillValue    = -9999.f;
    ta:coordinates   = "lon lat E_UTM N_UTM x y z time station_name";
    ta:grid_mapping  = "crs";
    ta:cell_methods  = "time: mean nz: mean";

data:
  station_name = "AWS 1";

  time = 86400; // last second of 17.01.2017

  time_bounds = 0, 86400; // time boundaries for 17.01.2017
```

```
vrs = _; // no value assigned

z = 44.0000, 46.0000, 48.0000; // mid heights of vertical intervals

z_bounds = 43.9000, 44.1000, // vertical intervals (0.2 m extent)
           45.9000, 46.1000,
           47.9000, 48.1000;

station_h = 42.0000;

crs = _; // no value assigned

y = 0.00000;

x = 0.00000;

N_UTM = 5813054.0;

E_UTM = 385412.00;

lat = 52.455631;

lon = 13.313614;

ta = -0.254583, 1.74172, 0.660524;
```

## A4 Example "Trajectories"

```
// Global attributes:

:title          = "featureType example for trajectory data";
:data_content   = "ta";
:source         = "mobile";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de;
                   Fehrenbach, Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "";
:comment        = "demo data for instantaneous trajectory values";
:keywords       = "air temperature; Berlin; trajectory data";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at
                   www.uc2-program.org/uc2_data_policy.pdf";
:campaign       = "IOP01";
:origin_time    = "2017-01-17 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:06 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 0.;
:rotation_angle = 0.f;
:featureType    = "trajectory";

dimensions:

  traj         = 1;
  ntime        = 5;
  max_name_len = 32;

variables:

  char traj_name(traj, max_name_len);
    traj_name:long_name     = "trajectory name";
    traj_name:standard_name = "platform_name";
    traj_name:cf_role       = "trajectory_id";

  int time(traj, ntime);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-17 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(traj, ntime);
    z:long_name     = "height above origin";
    z:standard_name = "height_above_mean_sea_level"; // origin_z = 0.0!
    z:units         = "m";
    z:axis          = "Z";
    z:positive      = "up";

  float height;
    height:long_name     = "height above surface";
    height:standard_name = "height";
```

```
    height:units         = "m";

  int crs;
    crs:long_name                       = "coordinate reference system";
    crs:grid_mapping_name               = "transverse_mercator";
    crs:semi_major_axis                 = 6378137.f;
    crs:inverse_flattening              = 298.257222101;
    crs:longitude_of_prime_meridian     = 0.f;
    crs:longitude_of_central_meridian   = 15.f;
    crs:scale_factor_at_central_meridian = 0.9996f;
    crs:latitude_of_projection_origin   = 0.f;
    crs:false_easting                   = 500000.f;
    crs:false_northing                  = 0.f;
    crs:units                           = "m";
    crs:epsg_code                       = "EPSG:25833";

  float y(traj, ntime);
    y:long_name = "distance to origin in y-direction";
    y:units     = "m";
    y:axis      = "Y";

  float x(traj, ntime);
    x:long_name = "distance to origin in x-direction";
    x:units     = "m";
    x:axis      = "X";

  double N_UTM(traj, ntime);
    N_UTM:long_name     = "northing";
    N_UTM:standard_name = "projection_y_coordinate";
    N_UTM:units         = "m";

  double E_UTM(traj, ntime);
    E_UTM:long_name     = "easting";
    E_UTM:standard_name = "projection_x_coordinate";
    E_UTM:units         = "m";

  double lat(traj, ntime);
    lat:long_name     = "latitude";
    lat:standard_name = "latitude";
    lat:units         = "degrees_north";

  double lon(traj, ntime);
    lon:long_name     = "longitude";
    lon:standard_name = "longitude";
    lon:units         = "degrees_east";

  float ta(traj, ntime);
    ta:long_name     = "air temperature";
    ta:standard_name = "air_temperature";
    ta:units         = "K";
    ta:_FillValue    = -9999.f;
    ta:coordinates   = "lon lat E_UTM N_UTM x y z time traj_name";
    ta:grid_mapping  = "crs";

data:
  traj_name = "Trajectory 1";

  time = 43200, 43800, 44400, 45000, 45600; // 12 UTC, 10 min time steps

  vrs = _; // no value assigned

  z = 43.1000, 43.2000, 43.9000, 43.5000, 43.4000; // altitudes of measurements

  height = 1.10000; // constant height above ground

  crs = _; // no value assigned

  y = 0.00000, 1.00000, 0.200000, 2.20000, 3.00000;

  x = 0.00000, 1.30000, 2.50000, 4.70000, 5.80000;
```

```
N_UTM = 5813054.0, 5813055.0, 5813054.2, 5813056.2, 5813057.0;
E_UTM = 385412.00, 385413.30, 385414.50, 385416.70, 385417.80;
lat = 52.455631, 52.455640, 52.455634, 52.455652, 52.455659;
lon = 13.313614, 13.313633, 13.313651, 13.313683, 13.313699;
ta = 0.861230, -1.53924, 0.710852, 1.28526, -0.224212;
```

## A5 Example "Ancillary variables and flags"

```
// Global attributes:

:title          = "example for ancillary variables and flags";
:data_content   = "meteo";
:source         = "AWS";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de;
                   Fehrenbach, Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "";
:comment        = "demo data for ancillary variables and flags";
:keywords       = "air temperature; Berlin; ancillary variables; flags";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at
                   www.uc2-program.org/uc2_data_policy.pdf";
:campaign       = "LTO";
:origin_time    = "2017-01-01 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:06 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 0.;
:rotation_angle = 0.f;
:featureType    = "timeSeries";

dimensions:

  station      = 1;
  ntime        = 24;
  nv           = 2;
  max_name_len = 32;

variables:

  char station_name(station, max_name_len);
    station_name:long_name     = "station name";
    station_name:standard_name = "platform_name";
    station_name:cf_role       = "timeseries_id";

  int time(station, ntime);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-01 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";
    time:bounds        = "time_bounds";

  int time_bounds(station, ntime, nv);

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(station);
    z:long_name     = "height above origin";
    z:standard_name = "height_above_mean_sea_level";
    z:units         = "m";
    z:axis          = "Z";
    z:positive      = "up";
```

```
float station_h(station);
  station_h:long_name     = "surface altitude";
  station_h:standard_name = "surface_altitude";
  station_h:units         = "m";

int crs;
  crs:long_name                        = "coordinate reference system";
  crs:grid_mapping_name                = "transverse_mercator";
  crs:semi_major_axis                  = 6378137.f;
  crs:inverse_flattening               = 298.257222101;
  crs:longitude_of_prime_meridian      = 0.f;
  crs:longitude_of_central_meridian    = 15.f;
  crs:scale_factor_at_central_meridian = 0.9996f;
  crs:latitude_of_projection_origin    = 0.f;
  crs:false_easting                    = 500000.f;
  crs:false_northing                   = 0.f;
  crs:units                            = "m";
  crs:epsg_code                        = "EPSG:25833";

float y(station);
  y:long_name = "distance to origin in y-direction";
  y:units     = "m";
  y:axis      = "Y";

float x(station);
  x:long_name = "distance to origin in x-direction";
  x:units     = "m";
  x:axis      = "X";

double N_UTM(station);
  N_UTM:long_name     = "northing";
  N_UTM:standard_name = "projection_y_coordinate";
  N_UTM:units         = "m";

double E_UTM(station);
  E_UTM:long_name     = "easting";
  E_UTM:standard_name = "projection_x_coordinate";
  E_UTM:units         = "m";

double lat(station);
  lat:long_name     = "latitude";
  lat:standard_name = "latitude";
  lat:units         = "degrees_north";

double lon(station);
  lon:long_name     = "longitude";
  lon:standard_name = "longitude";
  lon:units         = "degrees_east";

float ta(station, ntime);
  ta:long_name           = "air temperature";
  ta:standard_name       = "air_temperature";
  ta:units               = "degree_C";
  ta:_FillValue          = -9999.f;
  ta:coordinates         = "lon lat E_UTM N_UTM x y z time station_name";
  ta:grid_mapping        = "crs";
  ta:cell_methods        = "time: mean";
  ta:ancillary_variables = "ancillary_ta_flags ancillary_instrument_status";

float hur(station, ntime);
  hur:long_name           = "relative humidity";
  hur:standard_name       = "relative_humidity";
  hur:units               = "1";
  hur:_FillValue          = -9999.f;
  hur:coordinates         = "lon lat E_UTM N_UTM x y z time station_name";
  hur:grid_mapping        = "crs";
  hur:cell_methods        = "time: mean";
  hur:ancillary_variables = "ancillary_instrument_status";
```

```
  byte ancillary_instrument_status(station, ntime);
    ancillary_instrument_status:long_name     = "instrument status";
    ancillary_instrument_status:flag_meanings = "low_battery ventilation_on";
    ancillary_instrument_status:flag_masks    = 1b, 2b;

  byte ancillary_ta_flags(station, ntime);
    ancillary_ta_flags:long_name     = "air temperature flags";
    ancillary_ta_flags:flag_meanings = "okay out_of_range NA";
    ancillary_ta_flags:flag_values   = 0b, 1b, 2b;

data:
  station_name = "AWS 1";

  time = 1386000, 1389600, …, 1465200, 1468800; // sec after beginning of month

  time_bounds = 1382400, 1386000, // boundaries of hours
                1386000, 1389600,
                      …,       …,
                1461600, 1465200,
                1465200, 1468800;

  vrs = _; // no value assigned

  z = 44.0000; // 2 m above ground

  station_h = 42.0000;

  crs = _; // no value assigned

  y = 0.00000;

  x = 0.00000;

  N_UTM = 5813054.0;

  E_UTM = 385412.00;

  lat = 52.455631;

  lon = 13.313614;

  ta = -9999.00, -0.628962, …, 30.0000, 0.774297; // first is fill, 30 too large

  hur = 0.610804, 0.653351, …, 0.503260, 0.517553;

  ancillary_instrument_status = 0b, 1b, …, 3b, 2b; // 1=low bat, 2=vent, 3=both

  ancillary_ta_flags = 2b, 0b, …, 1b, 0b; // first is NA, 30 out of range
```

## A6 Example "Spectral data"

```
// Global attributes:

:title          = "example for spectral data";
:data_content   = "ncaa";
:source         = "APSS";
:version        = 1s;
:Conventions    = "CF-1.7";
:institution    = "Technische Universität Berlin, Fachgebiet Klimatologie";
:acronym        = "TUBklima";
:author         = "Scherer, Dieter, dieter.scherer@tu-berlin.de;
                   Holtmann, Achim, achim.holtmann@tu-berlin.de";
:contact_person = "Meier, Fred, fred.meier@tu-berlin.de;
                   Fehrenbach, Ute, ute.fehrenbach@tu-berlin.de";
:dependencies   = "";
:history        = "";
:references     = "";
:comment        = "demo data for spectral data";
:keywords       = "particles; Berlin; spectral data";
:licence        = "[UC]2 Open Licence; see [UC]2 data policy available at www.uc2-
program.org/uc2_data_policy.pdf";
:campaign       = "LTO";
:origin_time    = "2017-01-01 00:00:00 +00";
:creation_time  = "2020-05-07 15:15:06 +00";
:location       = "B";
:site           = "rothab1lawn";
:origin_x       = 385412.;
:origin_y       = 5813054.;
:origin_lon     = 13.3136143016031;
:origin_lat     = 52.4556312383161;
:origin_z       = 0.;
:rotation_angle = 0.f;
:featureType    = "timeSeries";

dimensions:

  station      = 1;
  ntime        = 24;
  nv           = 2;
  max_name_len = 32;
  bands_pm     = 3;

variables:

  char station_name(station, max_name_len);
    station_name:long_name     = "station name";
    station_name:standard_name = "platform_name";
    station_name:cf_role       = "timeseries_id";

  int time(station, ntime);
    time:long_name     = "time";
    time:standard_name = "time";
    time:units         = "seconds since 2017-01-01 00:00:00 +00";
    time:calendar      = "proleptic_gregorian";
    time:axis          = "T";
    time:bounds        = "time_bounds";

  int time_bounds(station, ntime, nv);

  int vrs;
    vrs:long_name   = "vertical reference system";
    vrs:system_name = "DHHN2016";

  float z(station);
    z:long_name     = "height above origin";
    z:standard_name = "height_above_mean_sea_level";
    z:units         = "m";
    z:axis          = "Z";
```

```
  z:positive      = "up";

float station_h(station);
  station_h:long_name     = "surface altitude";
  station_h:standard_name = "surface_altitude";
  station_h:units         = "m";

int crs;
  crs:long_name                       = "coordinate reference system";
  crs:grid_mapping_name               = "transverse_mercator";
  crs:semi_major_axis                 = 6378137.f;
  crs:inverse_flattening              = 298.257222101;
  crs:longitude_of_prime_meridian     = 0.f;
  crs:longitude_of_central_meridian   = 15.f;
  crs:scale_factor_at_central_meridian = 0.9996f;
  crs:latitude_of_projection_origin   = 0.f;
  crs:false_easting                   = 500000.f;
  crs:false_northing                  = 0.f;
  crs:units                           = "m";
  crs:epsg_code                       = "EPSG:25833";

float y(station);
  y:long_name = "distance to origin in y-direction";
  y:units     = "m";
  y:axis      = "Y";

float x(station);
  x:long_name = "distance to origin in x-direction";
  x:units     = "m";
  x:axis      = "X";

double N_UTM(station);
  N_UTM:long_name     = "northing";
  N_UTM:standard_name = "projection_y_coordinate";
  N_UTM:units         = "m";

double E_UTM(station);
  E_UTM:long_name     = "easting";
  E_UTM:standard_name = "projection_x_coordinate";
  E_UTM:units         = "m";

double lat(station);
  lat:long_name     = "latitude";
  lat:standard_name = "latitude";
  lat:units         = "degrees_north";

double lon(station);
  lon:long_name     = "longitude";
  lon:standard_name = "longitude";
  lon:units         = "degrees_east";

float ncaa(bands_pm, station, ntime);
  ncaa:long_name     = "number concentration of ambient aerosol
                        particles in air";
  ncaa:standard_name = "number_concentration_of_ambient_aerosol_
                        particles_in_air";
  ncaa:units         = "m-3"; // particles per cubic metre
  ncaa:_FillValue    = -9999.f;
  ncaa:coordinates   = "lon lat E_UTM N_UTM x y z time station_name
                        bands_pm bands_pm_size";
  ncaa:grid_mapping  = "crs";
  ncaa:cell_methods  = "time: mean";

int bands_pm(bands_pm);
  bands_pm:long_name = "particle size class number";
  bands_pm:units     = "1";

float bands_pm_size(bands_pm);
  bands_pm_size:long_name = "particle diameter";
  bands_pm_size:units     = "m-6";
```

```
    bands_pm_size:bounds    = "bands_pm_size_bounds";

  float bands_pm_size_bounds(bands_pm, nv);

data:
  station_name = "APSS 1";

  time = 1386000, 1389600, …, 1465200, 1468800; // sec since beginning of month

  time_bounds = 1382400, 1386000, // boundaries of hours
                1386000, 1389600,
                      …,       …,
                1461600, 1465200,
                1465200, 1468800;

  vrs = _; // no value assigned

  z = 44.0000; // 2 m above ground

  station_h = 42.0000;

  crs = _; // no value assigned

  y = 0.00000;

  x = 0.00000;

  N_UTM = 5813054.0;

  E_UTM = 385412.00;

  lat = 52.455631;

  lon = 13.313614;

  ncaa =  1.08040,  1.10315, …, 0.707215, 0.651837,
         0.158736, 0.604397, …, 0.279908, 0.421663,
          1.30170, 0.739476, …,  1.29708, 0.628731;

  bands_pm = 1, 2, 3; // size class number

  bands_pm_size = 0.600000, 1.75000, 6.25000; // central diameter of size class

  bands_pm_size_bounds = 0.200000, 1.00000, // bounds of size classes
                         1.00000, 2.50000,
                         2.50000, 10.0000;
```