PALM **I**nput **D**ata **S**tandard (PIDS) v1.9
(applies to PALM-4U)

General remarks on PIDS

PIDS defines all possible input parameters for PALM which must be provided for a model run in NetCDF format (version 4 or higher). The number and choice of parameters depends on the setup. This file gives an overview over all possible quantities that can be contained in the input files.

There are three different types of input files: *static*, *dynamic*, *radiation*, and *chemistry*, which must ne named (e.g. for a job named my_test_setup):

my_test_setup_static_driver.nc – contains all static information like orography, buildings, and surface classification.

my_test_setup_dynamic_driver.nc – contains dynamic information for the run, such as time-dependent boundary conditions and the initial state of the atmosphere.

my_test_setup_radiation_driver.nc – contains static and dynamic information of radiation properties (trace gas profiles, sky view factors).

my_test_setup_chemistry_driver.nc – contains all information on chemical species and emissions.

The origin of the model is the front left corner of the model domain at (z=0,y=0,x=0).

Missing definitions:

- Virtual measurements
- Multi-agent-system parameters
- Radiation quantities (initial profiles for RRTMG, sky view factors, ...)

General remarks on surface classification

The surface classification in PALM follows a three-step approach. First, a bulk land surface and soil classification is set. In a second step (when more detailed information is available), this classification is partly overwritten for each location (y,x). For buildings it is possible to prescribe explicit properties for single surface elements (step three). Surface types must be specified according to PIDS for each individual pixel with location (y,x). Missing values are not allowed for the bulk classification and mismatch of settings is checked by PALM internally.

For variables with _X, X can be a user-defined string which enables to store multiple datasets in one file.

The bulk parameterization is realized by the following fields:
vegetation_type, ***pavement_type***, ***building_type***, ***soil_type***, and ***water_type***.

Note that **for each location (y,x) at least ONE** of the parameters ***vegetation_type***, ***pavement_type***, ***building_type***, or ***water_type*** must be set to a non-missing value. If more than one type is defined at a given location, a tile approach is used and the distribution of these types is steered via ***surface_fraction***.

Note that a ***soil_type*** is required for each location (y,x) where either ***vegetation_type*** or ***pavement_type*** is a non-missing value.

The bulk classification provides default standard values for a variety of parameters required by PALM. User-defined surface types can be provided by using ****_type*** classes 0 and/or prescribing ****_pars*** fields (see below) at given locations where the ****_type*** settings are the overwritten.

In case more detailed information on vegetation is available (i.e. 3D „resolved“ vegetation), the leaf area density, position of tree trunks, and the root distribution in the soil of this vegetation is needed. This is realized by setting of ***leaf_area_density, basal_area_density,*** and ***root_area_density***. Note that there is a tool which converts arbitrary vegetation information into suitable PALM input data fields. When resolved vegetation is used, the under-tree vegetation type must be set in ***vegetation_type***. It is also possible to prescribe ***water_type*** and ***pavement_type*** surfaces below tree crowns.

In case detailed information on single or all building surface is available, the properties for single (or all) surface elements can be given via ***building_surface_pars***. Note that the size of these surface elements must be the same as the numerical grid used (typically 1m x 1m).

Important: all text strings must be provided as data type „char“ (NC_CHAR), as string variables are currently not supported by the Fortran NetCDF interface.

Global attributes:

(char)	*Conventions*	*“CF-1.7”* - NetCDF convention.
(char)	*data_content*	Text (max. 16 chars, see tables A1,A2)
(char)	*source*	Text
(int)	*version*	Text (1-999)
(char)	*dependencies*	Text
(char)	*history*	Information of data processing, separation by comma, e.g., *“2016-04-22 11:45: updated vegetation”*
(char)	*keywords*	list, separation by comma
(char)	*campaign*	Text (max. 12 chars)

(char)	*creation_time*	File creation date (UTC), format: YYYY-MM-DD hh:mm:ss +00
(char)	*title*	Short description, e.g., *"PALM input file for scenario 1b"*
(char)	*acronym*	Abreviation of insitution according to table A3 (max. 12 chars), e.g., "LUHimuk"
(char)	*institution*	Name of institution according to table A3, e.g., "Leibniz Universitaet Hannover, Institut fuer Meteorologie und Klimatologie"
(char)	*author*	First name, last name, email adress
(char)	*contact_person*	First name, last name, email adress
(char)	*license*	Text
(char)	*origin_time*	Reference point in time (UTC), format: *"YYYY-MM-DD hh:mm:ss +00"*
(char)	*location*	Name of city or region, e.g., *"Berlin",*
(char)	*site*	Name of model domain
(float)	*origin_x*	Reference easting in m (UTM), e.g., *"549020.0"*
(float)	*origin_y*	Reference northing in m (UTM), e.g., *"5802436.0"*
(float)	*origin_z*	*Reference height in m above sea level according to DHHN2016, e.g., "57.f"*
(float)	*origin_lat*	Defines the lower left corner (y, x) of the model domain in degrees north, e.g., *"52.37f"*
(float)	*origin_lon*	Defines the front left corner (y, x) of the model domain in degrees east, e.g., *"9.72f"*
(float)	*rotation_angle*	Clockwise angle of rotation in degrees between North positive y axis and the y axis in the data, e.g., *"0.0f"*
(char)	*references*	Citations if required and useful, separation by comma
(char)	*comment*	Miscellaneous information about the data or methods to produce it
(float)	*palm_version*	e.g., "5.0" to allow compatibility checks

Coordinate variables:

(float) time

time since reference point in seconds.

attributes

(char)	*long_name*	*"time"*
(char)	*standard_name*	*"time"*

(char) *units* *“seconds since 1970-01-01 00:00:00”*

(char) *axis* *“T”*

(int) ***s***

number of building surface elements

attributes

(char) *long_name* *“number of surface element”*

(char) *units* *“1”*

(float) ***z***

height above ground (center)

attributes

(char) *long_name* *“height above origin”*

(char) *standard_name* *“height_above_mean_sea_level”*, must only be set if “*origin_z* = 0”

(char) *units* *“m”*

(char) *axis* *“Z”*

(char) *positive* *“up”*

(float) ***zs***

height above ground of building surface element

attributes

(char) *long_name* *“height above origin”*

(char) *standard_name* *“height_above_mean_sea_level”*, must only be set if “*origin_z* = 0”

(char) *units* *“m”*

(char) *axis* *“Z”*

(char) *positive* *“up”*

(float) ***zw***

height above ground (shifted by +dz/2 in z-direction)

attributes

(char) *long_name* *“height above origin”*

(char) *standard_name* *“height_above_mean_sea_level”*, must only be set if “*origin_z* = 0”

(char) *units* *“m”*

(char) *axis* *“Z”*

(char) *positive* *“up”*

(float) zsoil

depth in the soil

attributes

(char)	*long_name*	*"depth in the soil"*
(char)	*standard_name*	*"height_below_mean_sea_level"*, must only be set if "*origin_z* = 0"
(char)	*units*	*"m"*
(char)	*axis*	*"Z"*
(char)	*positive*	*"down"*

(float) depth ***Subject to change!***

Only used in dynamic driver. Will be removed in future (replaced by zsoil)

depth in the soil

attributes

(char)	*long_name*	*"depth in the soil"*
(char)	*standard_name*	*"height_below_mean_sea_level"*, must only be set if "*origin_z* = 0"
(char)	*units*	*"m"*
(char)	*axis*	*"Z"*
(char)	*positive*	*"down"*

(float) zlad

height above ground of leaf area density and basal area density

attributes

(char)	*long_name*	*"height above origin"*
(char)	*standard_name*	*"height_above_mean_sea_level"*, must only be set if "*origin_z* = 0"
(char)	*units*	*"m"*
(char)	*axis*	*"Z"*
(char)	*positive*	*"up"*

(float) y

y-position

attributes

(char)	*long_name*	*"distance to origin in y-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"Y"*

(float) ys

y-position of building surface element

attributes

(char)	*long_name*	*"distance to origin in y-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"Y"*

(float) yv

y-position (shifted by -dy/2 in y-direction

attributes

(char)	*long_name*	*"distance to origin in y-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"Y"*

(float) x

x-position

attributes

(char)	*long_name*	*"distance to origin in x-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"X"*

(float) xs

x-position of building surface element

attributes

(char)	*long_name*	*"distance to origin in x-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"X"*

(float) xv

x-position (shifted by -2/dx in x-direction

attributes

(char)	*long_name*	*"distance to origin in x-direction"*
(char)	*units*	*"m"*
(char)	*axis*	*"X"*

(float) lat(y, x)

latitude of location (y, x)

attributes

(char) *long_name* *“latitude”*

(char) *standard_name* *“latitude”*

(char) *units* *“degrees_north”*

(float) ***latu(y, xu)***

latitude of location (y, xu)

attributes

(char) *long_name* *“latitude”*

(char) *standard_name* *“latitude”*

(char) *units* *“degrees_north”*

(float) ***latv(yv, x)***

latitude of location (yv, x)

attributes

(char) *long_name* *“latitude”*

(char) *standard_name* *“latitude”*

(char) *units* *“degrees_north”*

(float) ***lats(s)***

latitude of building surface element (s)

attributes

(char) *long_name* *“latitude”*

(char) *standard_name* *“latitude”*

(char) *units* *“degrees_north”*

(float) ***lon(y, x)***

longitude of location (y, x)

attributes

(char) *long_name* *“longitude”*

(char) *standard_name* *“longitude”*

(char) *units* *“degrees_east”*

(float) ***lonu(y, xu)***

longitude of location (y, xu)

attributes

(char) *long_name* *“longitude”*

(char) *standard_name* *“longitude”*

(char) *units* *“degrees_east”*

(float) lonv(yv, x)

longitude of location (yv, x)

attributes

(char)	*long_name*	*“longitude”*
(char)	*standard_name*	*“longitude”*
(char)	*units*	*“degrees_east”*

(float) lons(y, x)

longitude of building surface element (s)

attributes

(char)	*long_name*	*“longitude”*
(char)	*standard_name*	*“longitude”*
(char)	*units*	*“degrees_east”*

(float) azimuth(s)

azimuth angle of building surface element relative to the rotated system, allowed values: 0° (right), 90° (front), 180° (left), 270° (back)

attributes

(char)	*long_name*	*“azimuth angle”*
(char)	*standard_name*	*“surface_azimuth_angle”*
(char)	*units*	*“degrees”*

(float) zenith(s)

zenith angle of building surface element relative to the rotated system, allowed values: 0° (top), 180° (bottom)

attributes

(char)	*long_name*	*“zenith angle”*
(char)	*standard_name*	*“surface_zenith_angle”*
(char)	*units*	*“degrees”*

(float) dt_emission

time step of the emission data

attributes

(char)	*long_name*	*“emission data time step”*
(char)	*standard_name*	*“emission_time_step”*
(char)	*units*	*“s”*

(float) E_UTM([y,] x)

Projection x coordinate, two-dimensional in case of a rotated coordinate

system (rotation_angle > 0)

attributes

(char)	*long_name*	*"easting"*
(char)	*standard_name*	*"projection_x_coordinate"*
(char)	*units*	*"m"*

(float) Es_UTM(s)

Projection x coordinate for surface element s

attributes

(char)	*long_name*	*"easting"*
(char)	*standard_name*	*"projection_x_coordinate"*
(char)	*units*	*"m"*

(float) N_UTM([y,] x)

Projection y coordinate, two-dimensional in case of a rotated coordinate system (rotation_angle > 0)

attributes

(char)	*long_name*	*"northing"*
(char)	*standard_name*	*"projection_y_coordinate"*
(char)	*units*	*"m"*

(float) N_UTM(s)

Projection y coordinate for surface element s

attributes

(char)	*long_name*	*"northing"*
(char)	*standard_name*	*"projection_y_coordinate"*
(char)	*units*	*"m"*

(int) crs

Grid mapping definition

attributes

(char)	*long_name*	*"coordinate reference system"*
(char)	*grid_mapping_name*	e.g., *"transverse mercator"* or *"latitude_longitude"*
(float)	*semi_major_axis*	e.g., *"6378137.0"*
(float)	*inverse_flattening*	e.g., *"298.257222101"*
(float)	*longitude_of_prime_meridian*	e.g., *"0.0"*

(float)	*longitude_of_central_meridian*	e.g., *"9.0"*
(float)	*scale_factor_at_central_meridian*	e.g., *"0.9996"*
(float)	*latitude_of_projection_origin*	e.g., *"0.0"*
(float)	*false_easting*	e.g., *"500000.0"*
(float)	*false_northing*	e.g., *"0.0"*
(char)	*units*	*"m"*
(char)	*epsg_code*	e.g., *"EPSG:25832"* or *"EPSG:4258"*

Dimensions:

(integer) nbuilding_pars ***description:*** number of elements in building_pars
(integer) nsurface_fraction ***description:*** number of elements in surface_fraction
(integer) nvegetation_pars ***description:*** number of elements in vegetation_pars
(integer) nalbedo_pars ***description:*** number of elements in albedo_pars
(integer)* npavement_pars** ***description: number of elements in pavement_pars
(integer) npavement_subsurface_pars ***description:*** number of elements in pavement_pars
(integer)* nwater_pars** ***description: number of elements in water_pars
(integer)* nbuilding_surface_pars** ***description: number of elements in building_surface_pars
(integer)* nwall** ***description: number of elements in init_building_temperature
(integer) nazimuth **description:** number of azimuth angles (azimuth_uv)
(integer) nzenith **description:** number of zenith angles (zenith_uv)
(integer)* ns** ***description: number of elements in
(integer) ncat ***description:*** number of emission categories
(integer) nspecies ***description:*** number of emission species
(integer) nshoursyear ***description:*** number of hours per year
(integer) nmonthdayyear ***description:*** number of required input values for emission time rescaling factors (= 91)
(integer) npm ***description:*** number of PM species
(integer) nvoc ***description:*** number of VOC species
(integer) nsurfaces ***description:*** number of surface elemtents

Topography variables:

Topography can consist of the terrain height, artificial constructs (e.g. buildings), and vegetation on the terrain

[static] ***zt(y, x)***

terrain height in m above mean sea level

type: NC_FLOAT

coordinates

y	y-position (in m)
x	x-position (in m)

attributes

(char)	*long_name*	*"terrain_height"*
(char)	*res_orig*	original resolution of the data in m
(char)	*source*	*data source, e.g., "satellite data"*
(char)	*units*	*"m"*
(float)	*_FillValue*	*-9999.0*
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[static] ***buildings_2d(y, x), buildings_3d(z, y, x)***

building topology or building height, depending on setting of attribute *lod*. z=0 refers to the highest point of terrain height occupied by that building.

type: NC_FLOAT, NC_BYTE

coordinates

z	z-position (in m) (*lod* = *2* only)
y	y-position (in m)
x	x-position (in m)

attributes

(char)	*long_name*	*"building_height"* or *"building_flag"* (*lod* = 2)
(char)	*res_orig*	original resolution of the data in m
(char)	*source*	*data source, e.g., "satellite data"*
(char)	*units*	*"m"* or *"1"* (*lod* = 2)
(float/ byte)	*_FillValue*	*-9999.0* or *-127b* (*lod* = 2)
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*
(int)	*lod*	Level of detail (1 or 2) *lod* = *1* surface-mounted buildings (no holes), the variable provides building heights in m *lod = 2* 3D-topology, the variable provide either 1b (building) or 0b (no building)

(byte)	*valid_range*	*0b, 1b*; valid values (*lod* = *2* only)
(byte)	*flag_values*	*0b, 1b*; available values (*lod* = *2* only)
(char)	*flag_meanings*	„no building, building“ (*lod* = *2* only)

[static] ***obstruction_uv (azimuth_uv, zenith_uv, y, x)***

obstruction of the sky at a pixel location. This array is used for evaluating UV exposure

type: NC_BYTE

coordinates

zenith_uv	zenith angle (in °)
azimuth_uv	azimuth angle (in °)
y	y-position (in m)
x	x-position (in m)

attributes

(char)	*long_name*	*“obstruction”*
(char)	*res_orig*	original resolution of the data in m
(char)	*source*	*data source, e.g., “satellite data”*
(char)	*units*	*“1”*
(byte)	*_FillValue*	*-127b*
(char)	*coordinates*	*“E_UTM N_UTM lon lat”*
(char)	*grid_mapping*	*“crsUTM: E_UTM N_UTM crsETRS: lon lat”*
(byte)	*valid_range*	*0b, 1b*
(byte)	*flag_values*	*0b, 1b*
(char)	*flag_meanings*	„no obstruction, obstruction“

Surface classification data (level 1, unresolved)

[static] ***vegetation_type(y, x)***

classification of natural land surface types. Note that this setting is developed for mesoscale parametrization. In case of resolved vegetation, parameters must be refined. Parameter settings for individual classes are available at https://palm.muk.uni-hannover.de/trac/wiki/doc/app/lsmpar#vegetation_type

A default albedo_type is set according to the list below, but can be overwritten by setting ***albedo_type***

vegetation_type	default albedo_type	description
0	0	user-defined vegetation according to ***vegetation_pars***
1	17	bare soil (no vegetation)
2	2	crops, mixed farming
3	5	short grass
4	6	evergreen needleleaf trees
5	8	deciduous needleleaf trees
6	9	evergreen broadleaf trees
7	8	deciduous broadleaf trees
8	3	tall grass
9	11	desert
10	13	tundra
11	2	irrigated crops
12	11	semidesert
~~13~~	-	~~ice caps and glaciers~~ (not implemented yet)
14	4	bogs and marshes
15	4	evergreen shrubs
16	4	deciduous shrubs
17	7	mixed forest / woodland
18	8	interrupted forest

type: NC_BYTE

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char) long_name *"vegetation type classification"*

(char) res_orig original resolution of the data in m

(char) source *data source, e.g., "satellite data"*

(char) units *"1"*

(float) _FillValue *-127b*

(char) coordinates *"E_UTM N_UTM lon lat"*

(char) *grid_mapping* *“crsUTM: E_UTM N_UTM crsETRS: lon lat”*

[static] ***building_id(y, x)***

Running number from 1-N, where N is the total number of individual buildings. This parameter is used to identify single building envelopes.

type: NC_INT

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char) *long_name* *“building id numbers”*

(char) *res_orig* original resolution of the data in m

(char) *source* *data source, e.g., “satellite data”*

(char) *units* *“1”*

(float) *_FillValue* *-9999*

(char) *coordinates* *“E_UTM N_UTM lon lat”*

(char) *grid_mapping* *“crsUTM: E_UTM N_UTM crsETRS: lon lat”*

[static] ***building_type(y, x)***

Classification of building types. In the first step, building parameters cannot be set individually by ***building_pars***.

A default albedo_type is set according to the list below, but can be overwritten by setting ***albedo_type*** or ***albedo_pars***

building_type	**albedo_type**	**description**		
		name	type	age
0		user-defined type according to ***building_pars***		
1	33	R1	residential	- 1950
2	33	R2	residential	1951 - 2000
3	33	R3	residential	2001 -
4	33	O1	office	- 1950
5	33	O2	office	1951 - 2000
6	33	O3	office	2001 -

type: NC_BYTE

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char)	*long_name*	*"building type classification"*
(char)	*res_orig*	original resolution of the data in m
(char)	*source*	*data source, e.g., "satellite data"*
(char)	*units*	*"1"*
(float)	*_FillValue*	*-127*
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[static] ***pavement_type(y, x)***

Classification of pavements (on soil). The classification follows closely the definitions of OpenStreetMap, but with a reduced number of classes.

A default albedo_type is set according to the list below, but can be overwritten by setting ***albedo_type*** or ***albedo_pars***

pavement_type	albedo_type	description (to be continued)
0	0	user-defined pavement according to ***pavement_pars***
1	18	unknown pavement (asphalt/concrete mixture)
2	19	asphalt (asphalt concrete)
3	20	concrete (Portland concrete)
4	21	sett
5	22	paving stones
6	23	cobblestone
7	24	metal
8	25	wood
9	26	gravel
10	27	fine gravel
11	28	pebblestone
12	29	woodchips
13	30	tartan (sports)
14	31	artifical turf (sports)
15	32	clay (sports)

16	33	building (dummy)

type: NC_BYTE

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char) long_name *"pavement type classification"*

(char) res_orig original resolution of the data in m

(char) source *data source, e.g., "satellite data"*

(char) units *"1"*

(float) _FillValue *-127*

(char) coordinates *"E_UTM N_UTM lon lat"*

(char) grid_mapping *"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[static] ***water_type(y, x)***

Classification of water bodies (y,x).

A default albedo_type is set according to the list below, but can be overwritten by setting ***albedo_type*** or ***albedo_pars***

water_type	albedo_type	description
0	1	user-defined water body according to ***water_pars***
1	1	lake
2	1	river
3	1	ocean
4	1	pond
5	1	fountain

type: NC_BYTE

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char) long_name *"water type classification"*

(char) res_orig original resolution of the data in m

(char) source *data source, e.g., "satellite data"*

(char)	*units*	*"1"*
(float)	*_FillValue*	*-127*
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[static] ***soil_type([zsoil,] y, x)***

Classification of soil in terms of porosity (zsoil,y,x).

A default albedo_type is set according to the list below, but can be overwritten by setting ***albedo_type*** or ***albedo_pars***

soil_type	description
0	user-defined soil according to ***soil_pars***
1	coarse
2	medium
3	medium-fine
4	fine
5	very fine
6	organic

type: NC_BYTE

coordinates

zsoil	z-position (in m) in the soil (*lod* = *2* only)
y	y-position (in m)
x	x-position (in m)

attributes

(char)	*long_name*	*"soil type classification"*
(char)	*res_orig*	original resolution of the data in m
(char)	*source*	*data source, e.g., "satellite data"*
(char)	*units*	*"1"*
(float)	*_FillValue*	*-127*
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*
(int)	*lod*	Level of detail *lod* = 1 uniform soil texture in the vertical direction *lod* = 2 multi-level soil texture in the vertical direction

[static] ***albedo_type(y, x)***

Optional classification of albedo for ***vegetation_type***, ***water_type***, and ***pavement_type*** surfaces. Default values are set by settings of ***vegetation_type***, ***water_type***, and ***pavement_type***. The value of ***albedo_type*** will also overwrite setting via ***vegetation_pars***, ***water_pars***, or ***pavement_pars***.

Default ***albedo_type***:

albedo_type	description
0	user-defined vegetation according to ***albedo_pars***
1	ocean
2	mixed farming, tall grassland
3	tall/medium grassland
4	evergreen shrubland
5	short grassland/meadow/shrubland
6	evergreen needleleaf forest
7	mixed deciduous forest
8	deciduous forest
9	tropical evergreen broadleaved forest
10	medium/tall grassland/woodland
11	desert, sandy
12	desert, rocky
13	tundra
14	land ice
15	sea ice
16	snow
17	unknown pavement (asphalt/concrete mixture)
18	asphalt (asphalt concrete)
19	concrete (Portland concrete)
20	sett
21	paving stones
22	cobblestone
23	metal
24	wood
25	gravel
26	fine gravel

27	pebblestone
28	woodchips
29	tartan (sports)
30	artifical turf (sports)
31	clay (sports)

type: NC_BYTE

coordinates

y y-position (in m)

x x-position (in m)

attributes

(char) long_name *"albedo type classification"*

(char) res_orig original resolution of the data in m

(char) source *data source, e.g., "satellite data"*

(char) units *"1"*

(float) _FillValue *-127*

(char) coordinates *"E_UTM N_UTM lon lat"*

(char) grid_mapping *"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[static] ***surface_fraction(nsurface_fraction, y, x)***

Fraction of the respective surface type given via ***vegetation_type***, ***pavement_type*** and ***water_type***. Note that ***vegetation_type*** itself has also a tile approach (vegetion, bare soil, liquid water on plants). Also pavement has a tile approach to account for liquid water on the pavement. The sum over all fractions must be equal to one for each location. This parameter is only needed at locations (y, x) where more than one surface type (vegetation, pavement, water) is defined.

nsurface_fraction	**description**
0	fraction of vegetation (according to vegetation_type)
1	fraction of pavement (according to pavement_type)
2	fraction of water (according to water_type)

type: NC_FLOAT

coordinates

nsurface_fraction parameter index

y y-position (in m)

x x-position (in m)

attributes

(char)	*long_name*	*"surface tile fraction"*
(char)	*units*	*"1"*
(float)	*_FillValue*	*-9999.0*
(char)	*coordinates*	*"E_UTM N_UTM lon lat"*
(char)	*grid_mapping*	*"crsUTM: E_UTM N_UTM crsETRS: lon lat"*

Surface classification data (level 2, unresolved)

[static] ***vegetation_pars (nvegetation_pars, y, x)***

description: Parameters required for the bulk land surface parameterization. When ***vegetation_type*** = 0, all parameters must be set, otherwise single parameters set by ***vegetation_type*** can be overwritten.

nvegetation_pars	description
0	minimum canopy resistance (s/m)
1	leaf area index
2	vegetation coverage (0-1)
3	canopy resistance coefficient (1/hPa)
4	roughness length for momentum (m)
5	roughness length for heat (m)
6	skin layer heat conductivity (stable conditions) (W/m²/K)
7	skin layer heat conductivity (unstable conditions) (W/m²/K)
8	fraction of incoming shortwave radiation transmitted directly to the soil
9	heat capacity of the surface / skin layer (J/m²/K)
10	albedo type
11	emissivity

type: NC_FLOAT

coordinates:

p parameter index
y y-position (in m)
x x-position (in m)

attributes:

(char) long_name *„vegetation parameters“*
(char) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***soil_pars (nsoil_pars, [zsoil,] y, x)***

description: Parameters required for the soil parameterization.
When ***soil_type*** = 0, all parameters must be set, otherwise single parameters set by ***soil_type*** can be overwritten.

nsoil_pars	description
0	Van Genuchten parameter alpha
1	Van Genuchten parameter l
2	Van Genuchten parameter n
3	saturation hydraulic conductivity (m/s)
4	saturation soil moisture (m³/m³)
5	field capacity (m³/m³)
6	wilting point (m³/m³)
7	residual moisture (m³/m³)

type: NC_FLOAT

coordinates:

p parameter index
zsoil z-position (in m) in the soil (positive downward, *lod* = *2* only)
y y-position (in m)
x x-position (in m)

attributes:

(integer) *lod* Level of detail (1,2)
lod = *1* uniform soil texture in the vertical direction
lod = *2* multi-level soil texture in the vertical direction
(char) long_name *„soil parameters“*
(char) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*

(char) units = „1“
(float) _FillValue = -9999.0f
(char) coordinates = „E_UTM N_UTM lon lat“
(char) grid_mapping = „crsUTM: E_UTM N_UTM crsETRS: lon lat“

[static] ***building_pars (nbuilding_pars, y, x)***

description: Parameters required for the building parameterization when ***building_type*** = 0. In this parameterization, only one value per (y,x) pixel can be set. Parameters 20-37 can be used to specify different building properties for the ground floor level.

nbuilding_pars	description
0	wall fraction (0-1)
1	window fraction (0-1)
2	green fraction on wall (0-1)
3	green fraction on roof (0-1)
4	leaf area index of green fraction (roof)
5	leaf area index of green fraction (wall)
6	heat capacity of wall layer 1
7	heat capacity of wall layer 2
8	heat capacity of wall layer 3
9	thermal conducivity of wall layer 1
10	thermal conducivity of wall layer 2
11	thermal conducivity of wall layer 3
12	indoor target summer temperature (K)
13	indoor target winter temperature (K)
14	emissivity of wall fraction (0-1)
15	emissivity of green fraction (0-1)
16	emissivity o f window fraction (0-1)
17	transmissivity of window fraction (0-1)
18	roughness length for momentum (m)
19	roughness length for heat (m)
20	ground floor height (m)
21	ground floor wall fraction (0-1)
22	ground floor window fraction (0-1)

23	ground floor green fraction on wall (0-1)
24	ground floor green fraction on roof (0-1)
25	ground floor leaf area index of green fraction (wall)
26	ground floor heat capacity of wall layer 1
27	ground floor heat capacity of wall layer 2
28	ground floor heat capacity of wall layer 3
29	ground floor thermal conducivity of wall layer 1
30	ground floor thermal conducivity of wall layer 2
31	ground floor thermal conducivity of wall layer 3
32	ground floor emissivity of wall fraction (0-1)
33	ground floor emissivity of green fraction (0-1)
34	ground floor emissivity of window fraction (0-1)
35	ground floor transmissivity of window fraction (0-1)
36	ground floor roughness length for momentum (m)
37	ground floor roughness length for heat (m)
38	albedo_type of wall fraction
39	albedo_type of green fraction
40	albedo_type of window fraction
41	wall thickness of layer 1 (m)
42	wall thickness of layer 2 (m)
43	wall thickness of layer 3 (m)
44	wall thickness of layer 4 (m)
45	surface conductivity (will be removed in future version)
46	surface heat capacity (will be removed in future version)

type: NC_FLOAT

coordinates:

p parameter index
y y-position (in m)
x x-position (in m)

attributes:

(char) long_name *„building parameters“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***albedo_pars (nalbedo_pars, y, x)***

description: User-defined settings of albedo when ***albedo_type*** = 0. Note that parameters 0-5 are settings for the main surface type (e.g. vegetation for natural surfaces and walls/pavement for urban surfaces).

nalbedo_pars	description
0	Broadband albedo (buildings: wall fraction for building surfaces)
1	Longwave albedo (buildings: wall fraction for building surfaces)
3	Shortwave direct albedo (buildings: wall fraction for building surfaces)
4	Longwave albedo for green fraction (buildings only)
5	Shortwave albedo for green fraction (buildings only)
6	Longwave albedo for window fraction (buildings only)
7	Shortwave albedo for window fraction (buildings only)

type: NC_FLOAT
coordinates:

p parameter index
y y-position (in m)
x x-position (in m)

attributes:

(char) long_name *„albedo parameters“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***pavement_pars (npavement_pars, y, x)***

description: Parameters required for the bulk pavement parameterization in the land surface scheme when ***pavement_type = 0***

npavement_pars	description
0	roughness length for momentum (m)
1	roughness length for heat (m)
2	albedo type
3	emissivity (0-1)

type: NC_FLOAT

coordinates:

p parameter index
y y-position (in m)
x x-position (in m)

attributes:

(char) long_name *„pavement parameters“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***pavement_subsurface_pars (npavement_subsurface_pars, zsoil, y, x)***

description: Parameters required for the bulk pavement parameterization when ***pavement_type = 0.*** For all zsoil levels where ***pavement_subsurface_pars*** is not missing, the settings of the soil model are overwritten, i.e. thermal conductivity and heat capacties are set, and he layers are impermeable for water.

npavement_subsurface_pars	description
0	thermal conductivity (W/m/K) of the layer
1	volumetric heat capacity (J/m³/K) of the layer

type: NC_FLOAT
coordinates:

p parameter index
y y-position (in m)
x x-position (in m)

attributes:

(char) long_name *„pavement parameters“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***water_pars (nwater_pars, y, x)***

description: Parameters required for the parameterization of water surfaces when ***water_type = 0***. or when single parameters shall be overwritten. This list will be extended as soon as a better water scheme is available.

nwater_pars	description
0	water temperature (fixed) (K)
1	roughness length for momentum (Charnock parameterization is used if not set) (m)
2	roughness length for heat (Charnock parameterization is used if not set) (m)
3	heat conductivity between skin layer and water (stable conditions) (W/m²/K) (should not be changed)
4	heat conductivity between skin layer

	and water (unstable conditions) (W/m²/K) (should not be changed)
5	albedo type
6	emissivity (0-1)

type: NC_FLOAT
coordinates:
p parameter index
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„water parameters“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

Surface classification (level 3, resolved vegetation)

[static] ***lad (zlad, y, x)***
description: Vegetation resolved by the canopy model in terms of a three-dimensional leaf area density (LAD). A preprocessor tool is available to convert arbitrary vegetation information to an LAD field.

type: NC_FLOAT
coordinates:
zlad z-position (in m)
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„leaf area density“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„m2 m-3“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***bad (zlad, y, x)***
description: Basal area in m² (i.e. trunk area) per grid volume. A preprocessor tool is available to convert arbitrary vegetation information to an

basal area density field. The dimension of the field must be equal to that of the leaf area density.

type: NC_FLOAT
coordinates:
zlad z-position (in m)
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„basal area density“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„m2 m-3“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***root_area_dens_r (zsoil, y, x)***
description: Root area of the resolved vegetation in the soil.

type: NC_FLOAT
coordinates:
zsoil z-position (in m)
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„root area density of resolved vegetation“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***root_area_dens_s (zsoil, y, x)***
description: Root area of the parameterized vegetation in the soil that is set via ***vegetation_type***. When vegetation_type = 0, the ***root_area_density_lsm*** must be set for all locations where ***vegetation_type*** is not missing.

type: NC_FLOAT
coordinates:
zsoil z-position (in m)
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„root area density of parameterized vegetation“*

(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0f*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***tree_id (zlad, y, x)***

description: Id of the tree which belongs to the specific grid volume.

type: NC_INT
coordinates:
zlad z-position (in m)
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *„tree id“*
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[static] ***building_surface_pars (nbuilding_surface_pars,s)***

description: Detailed information for specific building surfaces. This variable can contain data for the entire domain or for selected areas where detailed data is available.

nbuilding_surface_pars	description
0	wall fraction (0-1)
1	window fraction (0-1)
2	green fraction on wall (0-1)
3	green fraction on roof (0-1)
4	leaf area index of green fraction
5	heat capacity of wall layer 1
6	heat capacity of wall layer 2
7	heat capacity of wall layer 3
8	thermal conducivity of wall layer 1
9	thermal conducivity of wall layer 2

10	thermal conducivity of wall layer 3
11	indoor target summer temperature
12	indoor target winter temperature
13	emissivity of wall fraction (0-1)
14	emissivity of green fraction (0-1)
15	emissivity o f window fraction (0-1)
16	transmissivity of window fraction (0-1)
17	roughness length for momentum (m)
18	roughness length for heat (m)
19	Broadband albedo of wall fraction
20	Longwave albedo of wall fraction
21	Shortwave albedo of wall fraction
22	Broadband albedo of window fraction
23	Longwave albedo of window fraction
24	Shortwave albedo of window fraction
24	Broadband albedo of green fraction
25	Longwave albedo of green fraction
26	Shortwave albedo of green fraction

type: NC_FLOAT
coordinates:

s	number of surface element
zs(s)	z value of surface element s
ys(s)	y value of surface element s
xs(s)	x value of surface element s
lats(s)	lat value of surface element s
lons(s)	lon value of surface element s
azimuth(s)	azimuth of surface element s
zenith(s)	zenith of surface element s
Es_UTM(s)	E_UTM value of surface element s
Ns_UTM(s)	N_UTM value of surface element s

attributes:

(char) long_name „“
(float) res_orig Original resolution of the data in m
(char) source Data source, e.g. *„satellite data“*
(char) units = *„1“*
(float) _FillValue = *-9999.0*
(char) coordinates = *„Es_UTM Ns_UTM lons lats“*

(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

Initialization data

[dynamic] ***init_atmosphere_Y (z, [y], [x])***

description: Initialization of prognostic variables. Y can be:

Y	**description**
pt	air potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)
w	wind component in z-direction (m/s)
s	passive scalar (arbritrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)
hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT

coordinates:

z z-position (in m)
y y-position (in m) (*lod* = *2* only)
x x-position (in m) (*lod* = *2* only)

attributes:

(integer) *lod* Level of detail (1,2)
lod = *1* One profile is used for initialization of the entire model domain (*z*)
lod = *2* Volume data is provided for initialization (*z*,*y*,*x*)
(char) long_name e.g. *„initial profile of potential temperature“*
(char) source Data source, e.g. *„satellite data“*
(char) units String that can be recognized by UDUNITS
(float) _FillValue = *-9999.0*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***init_soil_Y (depth, [y], [x])***

description: Initialization of prognostic variables. Y can be:

Y	description
t	soil temperature (K)
m	volumetric soil moisture (m^3/m^3)

type: NC_FLOAT

coordinates:

zsoil z-position in soil (in m)
y y-position (in m) (*lod* = *2* only)
x x-position (in m) (*lod* = *2* only)

attributes:

(integer) *lod* Level of detail (1,2)
lod = *1* One profile is used for initialization of the entire model domain (*z*)
lod = *2* Volume data is provided for initialization (*zsoil,y,x*)
(char) long_name e.g. *„initial profile of soil temperature“*
(char) source Data source, e.g. *„satellite data“*
(char) units String that can be recognized by UDUNITS
(float) _FillValue = *-9999.0*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***init_building_temperature_X ([s], [nwall,] [y,] [x])***

description: Initialization of walls and indoor temperatures with varying level of detail. For *lod* = *1* the same wall temperature is assumed for all walls of the building at location (y,x). With *lod* = *2* the wall layer temperatures of a building at (y,x) are provided individually for each layer. For *lod* = *3* the wall temperatures are provided individually for each surface element and for each wall layer.

type: NC_FLOAT

coordinates:

s number of surface element (*lod* = *3* only)
nwall wall layer (1-3) (*lod* = *2-3* only)
y y-position (in m) (*lod* = *1-2* only)
x x-position (in m) (*lod* = *1-2* only)

attributes:

(integer) *lod* Level of detail (1,2,3)
lod = *1* One value is used for all wall layers
lod = *2* Different values are provided for the different wall layers
lod = *3* Different values are provided for each individual surface element and wall layer

(char) long_name e.g. *„initial wall temperature“*
(char) source Data source, e.g. *„satellite data“*
(char) units *„K“*
(float) _FillValue *= -9999.0*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***init_pavement_temperature (depth, [y], [x])***

description: Initialization of the pavement temperature
type: NC_FLOAT
coordinates:
zsoil z-position in soil-pavement continuum (in m)
y y-position (in m) (*lod* = *2* only)
x x-position (in m) (*lod* = *2* only)
attributes:
(integer) lod Level of detail (1,2)
lod = *1* One profile is used for initialization of the entire model domain (*z*)
lod = *2* Volume data is provided for initialization (*zsoil,y,x*)
(char) long_name e.g. *„initial profile of pavement temperature“*
(char) source Data source, e.g. *„satellite data“*
(char) units *String that can be recognized by UDUNITS*
(float) _FillValue *= -9999.0*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***init_water_temperature (y, x)***

description: Initialization of the water temperature at location (y,x)
type: NC_FLOAT
coordinates:
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name e.g. *„initial water temperature“*
(char) source Data source, e.g. *„satellite data“*
(char) units *String that can be recognized by UDUNITS*
(float) _FillValue *= -9999.0*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***init_agents_X*** ***(to be defined, e.g. population structure depending on x,y)***

[radiation] ***init_radiation_X*** ***(to be defined, e.g. trace gas profiles for RRTMG, shape view***

factors for urban radiation)

Large-scale forcing data

[dynamic] ***tend_X_nud (time, z)***

description: Nudging data for X (requires cyclic boundary conditions)

X	**part of long_name attribute (Y)**	**units**
theta	potential temperature	K s-1
thetal	liquid water potential temperature	K s-1
q	humidity	s-1
qv	specific humidity	s-1
u	u wind component	m s-2
v	v wind component	m s-2
w	subsidence velocity	m s-2
s	passive scalar	s-1

type: NC_FLOAT
coordinates:
time time
z z-position (in m)
attributes:
(char) long_name "tendency for Y nudging"
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units see table
(float) _FillValue = *-9999.0f*

[dynamic] ***nudging_tau (time, z)***

description: Nudging relaxation time scale (requires cyclic boundary conditions)
type: NC_FLOAT
coordinates:
time time
z z-position (in m)
attributes:
(char) long_name *„nudging relaxation time scale"*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units *„s"*

[dynamic] ***ls_forcing_X (time, z)***

description: Large-scale forcing data via profiles of tendencies (requires cyclic boundary conditions). X can be as follows:

Subject to change! Some of these variables are not implemented as tendencies. Inconsistency! How shall they be defined in the end? This must then also be changed/added in the UC2-DS!

X	**part of long_name attribute (Y)**	**units**
ug	u wind component geostrophic	m s-1
vg	v wind component geostrophic	m s-1
sub_w	subsidence velocity of w	m s-1
thetal_adv	liquid water potential temperature due to advection	K s-1
q_adv	humidity due to advection	s-1
adv_s	advection of scalar (s)	s-1
adv_no	advection of NO	1e-6 s-1
adv_no2	advection of NO2	1e-6 s-1
adv_no3	advection of NO3	1e-6 s-1
adv_pm10	advection of PM10	1e-6 s-1
adv_hno3	advection of HNO3	1e-6 s-1
adv_so4	advection of SO4	1e-6 s-1
adv_yyy	advection of species „YYY“	1e-6 s-1
thetal_sub	liquid water potential temperature due to subsidence	K s-1
q_sub	humidity due to subsidence	s-1
sub_s	subsidence velocity of scalar	s-1

type: NC_FLOAT

coordinates:

time time

z z-position (in m)

attributes:

(char) long_name *“tendency for Y”*

(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*

(char) units see table

[dynamic] ***ls_forcing_left_Y (time, z,y)***

description: Large-scale forcing data (boundary conditions) for left model boundary. Y is as follows

Y	description
pt	potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)
w	subsidence velocity (m/s)
s	passive scalar (arbitrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)
hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT
coordinates:
time time
z/zw z-position (in m)
y/yv y-position (in m)
attributes:
(char) long_name e.g. *„large scale forcing for left model boundary for X“*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*
(char) units String that can be recognized by UDUNITS
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***ls_forcing_right_Y (time, z,y)***
description: Large-scale forcing data (boundary conditions) for right model boundary. Y is as follows

Y	description
pt	potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)

w	subsidence velocity (m/s)
s	passive scalar (arbitrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)
hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT
coordinates:
time time
z/zw z-position (in m)
y/yv y-position (in m)
attributes:
(char) long_name e.g. *„large scale forcing for right model boundary for X“*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*
(char) units String that can be recognized by UDUNITS
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***ls_forcing_north_Y (time, z,x)***
description: Large-scale forcing data (boundary conditions) for back model boundary. Y is as follows
Subject to change! Will be renamed in future!

Y	**description**
pt	potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)
w	subsidence velocity (m/s)
s	passive scalar (arbitrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)

hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT
coordinates:
time time
z/zw z-position (in m)
x/xu x-position (in m)
attributes:
(char) long_name e.g. *„large scale forcing for back model boundary for X“*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*
(char) units String that can be recognized by UDUNITS
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***ls_forcing_south_Y (time, z,x)***
description: Large-scale forcing data (boundary conditions) for front model boundary. Y is as follows
Subject to change! Will be renamed in future!

Y	**description**
pt	potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)
w	subsidence velocity (m/s)
s	passive scalar (arbitrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)
hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT
coordinates:
time time

z/zw z-position (in m)
x/xu x-position (in m)

attributes:

(char) long_name e.g. *„large scale forcing for front model boundary for X“*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*
(char) units String that can be recognized by UDUNITS
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[dynamic] ***ls_forcing_top_Y (time, y,x)***

description: Large-scale forcing data (boundary conditions) for top model boundary. Y is as follows

Y	**description**
pt	potential temperature (K)
qv	specific humidity (kg/kg)
u	wind component in x-direction (m/s)
v	wind component in y-direction (m/s)
w	subsidence velocity (m/s)
s	passive scalar (arbitrary units)
no	NO (ppm)
no2	NO2 (ppm)
no3	NO3 (ppm)
pm10	PM10 (ppm)
hno3	HNO3 (ppm)
so4	SO4 (ppm)
yyy	species „YYY“ (ppm)

type: NC_FLOAT

coordinates:

time time
y/yv y-position (in m)
x/xu x-position (in m)

attributes:

(char) long_name e.g. *„large scale forcing for top model boundary for X“*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00“*
(char) units String that can be recognized by UDUNITS
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

Radiation forcing data

[radiation] ***rad_swd_dif_0 (time, y,x)***

description: Incoming diffuse shortwave radiative flux at the surface
type: NC_FLOAT
coordinates:
time time
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *"incoming diffuse shortwave radiative flux at the surface"*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units *„W m-2"*
(char) coordinates = *„E_UTM N_UTM lon lat"*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[radiation] ***rad_swd_dir_0 (time, y,x)***

description: Incoming direct shortwave radiative flux at the surface
type: NC_FLOAT
coordinates:
time time
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *"incoming direct shortwave radiative flux at the surface"*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units *„W m-2"*
(char) coordinates = *„E_UTM N_UTM lon lat"*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat"*

[radiation] ***rad_swu_dif_0 (time, y,x)***

description: Outgoing diffuse shortwave radiative flux at the surface
type: NC_FLOAT
coordinates:
time time
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *"outgoing diffuse shortwave radiative flux at the surface"*
(char) source Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units *„W m-2"*

(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

[radiation] ***rad_swu_dir_0 (time, y,x)***

description: Outgoing direct shortwave radiative flux at the surface
type: NC_FLOAT
coordinates:
time time
y y-position (in m)
x x-position (in m)
attributes:
(char) long_name *“outgoing direct shortwave radiative flux at the surface”*
(char) source Data source, e.g.
„COSMO analysis from 2003-05-21 13:00“
(char) units *„W m-2“*
(char) coordinates = *„E_UTM N_UTM lon lat“*
(char) grid_mapping = *„crsUTM: E_UTM N_UTM crsETRS: lon lat“*

Emission data

[chemistry] ***emission_category_index(ncat)***

description: Category index of the emission quantity in question
type: NC_BYTE
coordinates:
ncat number of categories

attributes:
(char) long_name “emission category Index”
(char) standard_name “emission_cat_index”
(char) units *„1“*

[chemistry] ***emission_category_name(ncat)***

description: Emission categories names (match to ***emission_category_index*** of the same index element)
type: NC_STRING
coordinates:
ncat number of categories

attributes:
(char) long_name “emission category name”
(char) standard_name “emission_cat_name”
(char) units *„1“*

[chemistry] ***emission_name(nspecies)***

description: List of all names of emitted species. The number of species *nspecies* varies depending on the employed chemistry scheme.
type: NC_STRING
coordinates:

nspecies	number of emission species

attributes:

(char) long_name	"emission species name"
(char) standard_name	"emission_name"
(char) *units*	*„1“*

[chemistry] ***emission_index(nspecies)***

description: Index of the emitted species.
type: NC_USHORT
coordinates:

nspecies	number of emission species

attributes:

(char) long_name	"emission species index"
(char) standard_name	"emission_index"
(char) *units*	*„1“*

[chemistry] ***emission_values(nspecies,[ncat],[dt_emission],[z],y,x)***

description: Emission values of the different species. The coordinates vary based on the chosen level of detail.
type: NC_FLOAT
coordinates:

nspecies	number of emission species
ncat	number of emission category (*lod* = 1 only)
dt_emission	time step (in s) (*lod* = 2 only)
z	z-position (in m) (*lod* = 2 only)
y	y-position (in m)
x	x-position (in m)

attributes:

(integer)	*lod*	level of detail (1,2)
	lod = 1	Emissions are provided in categories and yearly. They are then rescaled by pre-defined re-scaling factors.
	lod = 2	Emissions are provided for each point in

space and at given points in time.

(char) long_name “emission values”
(char) standard_name “emission_values”
(float) dt_emission e.g. “3600.0” (lod = 2 only)
(char) units „kg/grid/yr“ (lod = 1) „kg/m2/dt_emission“ (lod = 2) (tbc)
(float) _FillValue = -9999.0f
(char) coordinates = „E_UTM N_UTM lon lat“
(char) grid_mapping = „crsUTM: E_UTM N_UTM crsETRS: lon lat“

[chemistry] ***emission_time_factors(ncat,[nhoursyear],[nmonthdayhour])***

description: Emission time scaling factors for emission_values (lod = 1). Two different time factors are possible: 1) Scaling according to month-day-hour classification (lod = 1), or scaling for each hour of the year (lod = 2).
type: NC_FLOAT
coordinates:

ncat number of emission categories
nhoursyear number of time scaling factors (lod = 2 only)
nmonthdayhour number of time scaling factors (lod = 1 only)

Index	Description
1-12	scaling factor for the index month of the year (sum must be 1)
13-19	scaling factor for the index day of the week (sum must be 1)
20-43	scaling factor for the index hour of the wordking day (sum must be 1)
44-67	scaling factor for the index hours of a saturday (sum must be 1)
68-91	scaling factor for the index hours of a sunday/public holiday (sum must be 1)

attributes:

(integer) lod Level of detail (1,2)
lod = 1 Classification in month-day-hour
lod = 2 Scaling factors for each hour of the year
(char) long_name “emission time scaling factors”
(char) standard_name “emission_time_scaling_factors”
(char) units „1“

[chemistry] ***composition_nox(ncat,1:2)***

description: Composition of species NOx (NO and NO2). The sum for each ***ncat*** must be equal to one.
type: NC_FLOAT
coordinates:

ncat	number of emission categories
1-2	1: NO, 2: NO2

attributes:

(char) long_name	"composition of NOx"
(char) standard_name	"composition_nox"
(char) *units*	*„1“*

[chemistry] ***composition_sox(ncat,1:2)***

description: Composition of species SOx (SO2 and SO4). The sum for each ***ncat*** must be equal to one.
type: NC_FLOAT
coordinates:

ncat	number of emission categories
1-2	1: SO2, 2: SO4

attributes:

(char) long_name	"composition of SOx"
(char) standard_name	"composition_sox"
(char) *units*	*„1“*

[chemistry] ***emission_pm_name(npm)***

description: List of all PM names.
type: NC_STRING
coordinates:

npm	number of PM species

attributes:

(char) long_name	"PM name"
(char) standard_name	"pm_name"
(char) *units*	*„1“*

[chemistry] ***composition_pm(ncat,npm,1:3)***

description: Composition of PM emission species 1-3 (1: PM10, 2: PM2.5, 3: PM1). The sum for each ***ncat*** must be equal to one.
type: NC_FLOAT
coordinates:

ncat number of emission categories
npm number of PM species

attributes:
(char) long_name “composition of PM”
(char) standard_name “composition_PM”
(char) units „1“

[chemistry] ***emission_voc_name(nvoc)***

description: List of all VOC names.
type: NC_STRING
coordinates:
nvoc number of VOC species

attributes:
(char) long_name “VOC name”
(char) standard_name “voc_name”
(char) units „1“

[chemistry] ***composition_voc(ncat,nvoc)***

description: Composition of VOC emission species. The sum for each ***ncat*** must be equal to one.
type: NC_FLOAT
coordinates:
ncat number of emission categories
nvoc number of VOC species

attributes:
(char) long_name “composition of VOC”
(char) standard_name “composition_voc”
(char) units „1“

[chemistry] ***emission_heat(ncat)***

description: Heat content of emissions
type: NC_FLOAT
coordinates:
ncat number of emission categories

attributes:
(char) long_name “emission heat content”
(char) standard_name “emission_heat_content”
(char) units „J“

[chemistry] ***emission_stack_height(y,x)***

description: Height of the stacks
type: NC_FLOAT
coordinates:

y	y-position (in m)
x	x-position (in m)

attributes:

(char) long_name	"emission stack height"
(char) standard_name	"emission_stack_height"
(char) units	*„m"*
(float) _FillValue	= *-9999.0f*
(char) coordinates	= *„E_UTM N_UTM lon lat"*
(char) grid_mapping	= *„crsUTM: E_UTM N_UTM crsETRS: lon lat"*

General PALM surface forcing data

[dynamic] ***surface_forcing_Y (time)***

description: Large-scale surface forcing data. Surface forcing can be either by Neumann conditions (flux forcing) or by Dirichlet conditions (temperature, humidity forcing). Y can be as follows

Y	**description**
shf	surface sensible heat flux
qsws	surface latent heat flux
pt_surface	surface potential temperature
qv_surface	surface specific humidity
surface_pressure	surface pressure

type: NC_FLOAT
coordinates:

time

attributes:

(char) long_name	e.g. *„large scale forcing data for surface temperature"*
(char) source	Data source, e.g. *„COSMO analysis from 2003-05-21 13:00"*
(char) units	*String that can be recognized by UDUNITS*