CERC hub reference manual

Release 0.3.0.5

CERC Next-Generation Cities

Dec 12, 2024

CONTENTS:

CHAPTER

ONE

CERC HUB' REFERENCE MANUAL

1.1 Authors

- Guillermo Gutierrez Morote
- Pilar Monsalvete Alvarez de Uribarri

1.2 Contributors

- Seyedehrabeeh Hosseinihaghighi
- Milad Aghamohamadnia
- Peter Yefi
- Koa Wells
- Sanam Dabirian
- Soroush Samareh Abolhassani

1.3 About the HUB

This document contains the essential documentation for the CERC HUB, a set of classes, factories, and helpers that simplifies the research at urban scale in multiples domains; these components are designed around three central axes, **extensibility**, **code clarity** and **consistency** as we intend to allow domain experts to perform urban scale simulations with multiple programs and enrich the city from several data sources. HUB is composed of four main components: **city model structure**, **factories**, **catalogs** and, **helpers**.

- **City model structure** is the set of classes designed to be familiar to the domain experts; this familiarity will be possible thanks to using a *standard-based* approach in order to flatten the learning curve. These classes compose the CERC *data model* that provides a simple way to study cities at an urban scale after the enriching process.
- **Factories** are pieces of code in charge of import and export information in and out of the **data model** they will perform the needed conversions to read or write different formats such as epw weather files, insel files or citygml. these factories are mean to be extended, allowing the HUB ecosystem to expand with new formats.
- **Catalogs** are datasets used in the enrichment of the city that can also be used by third party consumers like researchers or simulations software.
- **Helpers** are sets of general tools used by any of the other parts and does not fit in any of the previous categories.

CHAPTER

TWO

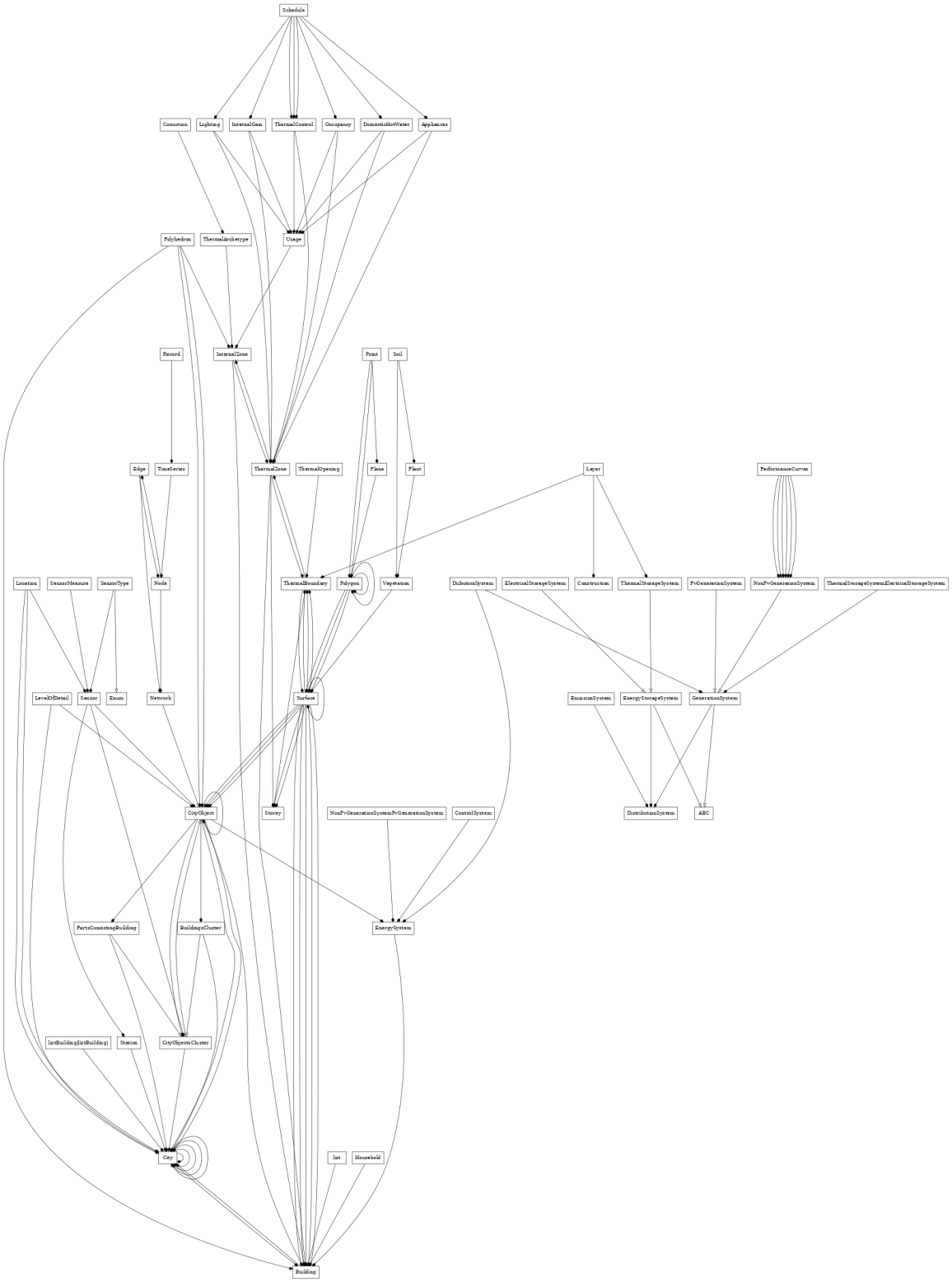
CITY MODEL STRUCTURE

The **city model structure** contains the common data model intended to represent a city digital twin, CERC team and contributors, will further extend these classes to include other domains, in the following sections, researchers and developers could find information about the methods and properties exposed by the city model structure classes.

Important: Please take a look to HUB tutorial to see how to use HUB for your own research

[how to use the hub]

2.1 City model structure UML



2.2 Folder structure

2.2.1 city_model_structure

Main city objects.

```
../hub/hub/city_model_structure/
├── building.py
├── buildings_cluster.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── __init__.py
├── level_of_detail.py
├── network.py
└── parts_consisting_building.py

2 directories, 13 files
```

2.2.2 attributes

Geometrical and non physical components of the city.

```
../hub/hub/city_model_structure/attributes/
├── edge.py
├── __init__.py
├── node.py
├── plane.py
├── point.py
├── polygon.py
├── polyhedron.py
├── record.py
├── schedule.py
└── time_series.py

1 directory, 10 files
```

2.2.3 building_demand

Main classes to model building energy demand.

```
../hub/hub/city_model_structure/building_demand/
├── appliances.py
├── construction.py
├── domestic_hot_water.py
├── household.py
├── __init__.py
├── internal_gain.py
├── internal_zone.py
├── layer.py
├── lighting.py
├── occupancy.py
├── storey.py
├── surface.py
├── thermal_archetype.py
├── thermal_boundary.py
├── thermal_control.py
├── thermal_opening.py
├── thermal_zone.py
└── usage.py

1 directory, 18 files
```

2.2.4 energy_systems

Main classes to model energy systems.

```
../hub/hub/city_model_structure/energy_systems/
├── control_system.py
├── distribution_system.py
├── electrical_storage_system.py
├── emission_system.py
├── energy_storage_system.py
├── energy_system.py
├── generation_system.py
├── __init__.py
├── non_pv_generation_system.py
├── performance_curve.py
├── pv_generation_system.py
└── thermal_storage_system.py

1 directory, 12 files
```

2.2.5 iot

Classes to model IoT devices.

```
../hub/hub/city_model_structure/iot/
├── __init__.py
├── sensor_measure.py
├── sensor.py
├── sensor_type.py
└── station.py

1 directory, 5 files
```

2.2.6 full schema

```
../hub/hub/city_model_structure/
├── attributes
│   ├── edge.py
│   ├── __init__.py
│   ├── node.py
│   ├── plane.py
│   ├── point.py
│   ├── polygon.py
│   ├── polyhedron.py
│   ├── record.py
│   ├── schedule.py
│   └── time_series.py
├── building_demand
│   ├── appliances.py
│   ├── construction.py
│   ├── domestic_hot_water.py
│   ├── household.py
│   ├── __init__.py
│   ├── internal_gain.py
│   ├── internal_zone.py
│   ├── layer.py
│   ├── lighting.py
│   ├── occupancy.py
│   ├── storey.py
│   ├── surface.py
│   ├── thermal_archetype.py
│   ├── thermal_boundary.py
│   ├── thermal_control.py
│   ├── thermal_opening.py
│   ├── thermal_zone.py
│   └── usage.py
├── energy_systems
│   ├── control_system.py
│   ├── distribution_system.py
│   ├── electrical_storage_system.py
│   ├── emission_system.py
│   ├── energy_storage_system.py
│   ├── energy_system.py
│   ├── generation_system.py
│   ├── __init__.py
│   ├── non_pv_generation_system.py
│   ├── performance_curve.py
│   ├── pv_generation_system.py
│   └── thermal_storage_system.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── iot
│   ├── __init__.py
│   ├── sensor_measure.py
│   ├── sensor.py
│   ├── sensor_type.py
│   └── station.py
├── building.py
├── buildings_cluster.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── __init__.py
├── level_of_detail.py
├── network.py
└── parts_consisting_building.py

6 directories, 58 files
```

2.3 Classes

2.3.1 City

class City(*lower_corner*, *upper_corner*, *srs_name*)

Inherit: :py:class:object

City class

add_building_alias(*building*, *alias*)

Add an alias to the building

add_city_object(*new_city_object*)

Add a CityObject to the city

Parameter new_city_object CityObject

Returns None or not implemented error

add_city_objects_cluster(*new_city_objects_cluster*)

Add a CityObject to the city

Parameter new_city_objects_cluster CityObjectsCluster

Returns None or NotImplementedError

building_alias(*alias*) → Building]] | None

Retrieve the city CityObject with the given alias alias

⚠ Building alias is not guaranteed to be unique

Parameter alias str

Returns None or [CityObject]

property buildings → Optional[[*Building*]]

Get the buildings belonging to the city

Returns None or [Building]

property buildings_clusters → Optional[[*BuildingsCluster*]]

Get buildings clusters belonging to the city

Returns None or [BuildingsCluster]

city_object(*name*) → OptionalCityObject]

Retrieve the city CityObject with the given name

Parameter name str

Returns None or CityObject

property city_objects → Optional[[*CityObject*]]

Get the city objects belonging to the city

Returns None or [CityObject]

property city_objects_clusters → Optional[[*CityObjectsCluster*]]

Get city objects clusters belonging to the city

Returns None or [CityObjectsCluster]

property climate_file → Optional[Path]

Get the climate file full path

Returns None or Path

property climate_reference_city → Optional[str]

Get the name for the climatic information reference city

Returns None or str

property copy → *City*

Get a copy of the current city

property country_code

Get city country code

Returns str

property generic_energy_systems → dict

Get dictionary with generic energy systems installed in the city

Returns dict

property latitude → Optional[float]

Get city latitude in degrees

Returns None or float

property level_of_detail → *LevelOfDetail*

Get level of detail of different aspects of the city: geometry, construction and usage

Returns LevelOfDetail

static load(*city_filename*) → *City*

Load a city saved with city.save(city_filename)

Parameter city_filename city filename

Returns City

static load_compressed(*compressed_city_filename*, *destination_filename*) → *City*

Load a city from compressed_city_filename

Parameter compressed_city_filename Compressed pickle as source

Parameter destination_filename Pickle file as destination

Returns City

property location → *Location*

Get city location

Returns Location

property longitude → Optional[float]

Get city longitude in degrees

Returns None or float

property lower_corner → [float]

Get city lower corner

Returns [x,y,z]

merge(*city*) → *City*

Return a merged city combining the current city and the given one

Returns City

property name

Get city name

Returns str

property parts_consisting_buildings → Optional[[*PartsConsistingBuilding*]]

Get parts consisting buildings belonging to the city

Returns None or [PartsConsistingBuilding]

region(*center*, *radius*) → *City*

Get a region from the city

Parameter center specific point in space [x, y, z]

Parameter radius distance to center of the sphere selected in meters

Returns selected_region_city

property region_code

Get city region name

Returns str

remove_city_object(*city_object*)

Remove a CityObject from the city

Parameter city_object CityObject

Returns None

save(*city_filename*)

Save a city into the given filename

Parameter city_filename destination city filename

Returns None

save_compressed(*city_filename*)

Save a city into the given filename

Parameter city_filename destination city filename

Returns None

property srs_name → Optional[str]

Get city srs name

Returns None or str

property stations → [*Station*]

Get the sensors stations belonging to the city

Returns [Station]

property time_zone → Optional[float]

Get city time_zone

Returns None or float

property upper_corner → [float]

Get city upper corner

Returns [x,y,z]

2.3.2 CityObject

class CityObject(*name*, *surfaces*)

Inherit: :py:class:object

class CityObject

property beam → dict

Get beam radiation surrounding the city object in J/m2

Returns dict{dict{[float]}}

property centroid → [float]

Get city object centroid

Returns [x,y,z]

property detailed_polyhedron → *Polyhedron*

Get city object polyhedron including details such as holes

Returns Polyhedron

property diffuse → dict

Get diffuse radiation surrounding the city object in J/m2

Returns dict{dict{[float]}}

property direct_normal → dict

Get beam radiation surrounding the city object in J/m2

Returns dict{dict{[float]}}

property external_temperature → {float}

Get external temperature surrounding the city object in Celsius

Returns dict{dict{[float]}}

property global_horizontal → dict

Get global horizontal radiation surrounding the city object in J/m2

Returns dict{dict{[float]}}

property ground_temperature → dict

Get ground temperature under the city object in Celsius at different depths in meters for different time steps

example of use: {month: {0.5: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]}}

Returns dict{dict{[float]}}

property level_of_detail → *LevelOfDetail*

Get level of detail of different aspects of the city: geometry, construction and usage

Returns LevelOfDetail

property lower_corner

Get city object lower corner coordinates [x, y, z]

Returns [x,y,z]

property max_height → float

Get city object maximal height in meters

Returns float

property name

Get city object name

Returns str

property neighbours → Optional[[*CityObject*]]

Get the list of neighbour_objects and their properties associated to the current city_object

property sensors → [*Sensor*]

Get sensors belonging to the city object

Returns [Sensor]

property simplified_polyhedron → *Polyhedron*

Get city object polyhedron, just the simple lod2 representation

Returns Polyhedron

surface(*name*) → OptionalSurface]

Get the city object surface with a given name

Parameter name str

Returns None or Surface

surface_by_id(*identification_number*) → OptionalSurface]

Get the city object surface with a given name

Parameter identification_number str

Returns None or Surface

property surfaces → [*Surface*]

Get city object surfaces

Returns [Surface]

property type → str

Get city object type

Returns str

property upper_corner

Get city object upper corner coordinates [x, y, z]

Returns [x,y,z]

property volume → float

Get city object volume in cubic meters

Returns float

2.3.3 City Objects Cluster

class CityObjectsCluster(*name*, *cluster_type*, *city_objects*)

Inherit: ABC, CityObject

CityObjectsCluster(ABC) class

add_city_object(*city_object*) → CityObject]

add new object to the cluster

Returns [CityObjects]

property city_objects

raises not implemented error

property name

Get cluster name

Returns str

property sensors → [*Sensor*]

Get sensors belonging to the city objects cluster

Returns [Sensor]

property type

raises not implemented error

2.3.4 Building

class Building(*name*, *surfaces*, *year_of_construction*, *function*, *usages=None*, *terrains=None*, *city=None*)

Inherit: CityObject

Building(CityObject) class

add_alias(*value*)

Add a new alias for the building

property aliases

Get the alias name for the building

Returns str

property appliances_electrical_demand → dict

Get appliances electrical demand in J

Returns dict{[float]}

property appliances_peak_load → Optional[dict]

Get appliances peak load in W

Returns dict{[float]}

property attic_heated → Optional[int]

Get if the city object attic is heated

0: no attic in the building

1: attic exists but is not heated

2: attic exists and is heated

Returns None or int

property average_storey_height → Optional[float]

Get building average storey height in meters

Returns None or float

property basement_heated → Optional[int]

Get if the city object basement is heated

0: no basement in the building

1: basement exists but is not heated

2: basement exists and is heated

Returns None or int

property city → *City*

Get the city containing the building

Returns City

property cold_water_temperature → {float}

Get cold water temperature in degrees Celsius

Returns dict{[float]}

property cooling_consumption

Get energy consumption for cooling according to the cooling system installed in J

return: dict

property cooling_demand → dict

Get cooling demand in J

Returns dict{[float]}

property cooling_peak_load → Optional[dict]

Get cooling peak load in W

Returns dict{[float]}

property distribution_systems_electrical_consumption

Get total electricity consumption for distribution and emission systems in J

return: dict

property domestic_hot_water_consumption

Get energy consumption for domestic according to the domestic hot water system installed in J

return: dict

property domestic_hot_water_heat_demand → dict

Get domestic hot water heat demand in J

Returns dict{[float]}

property domestic_hot_water_peak_load → Optional[dict]

Get cooling peak load in W

Returns dict{[float]}

property eave_height

Get building eave height in meters

Returns float

property energy_consumption_breakdown → dict

Get energy consumption of different sectors

return: dict

property energy_systems → Optional[[EnergySystem]]

Get list of energy systems installed to cover the building demands

Returns [EnergySystem]

property energy_systems_archetype_cluster_id

Get energy systems archetype id

Returns str

property energy_systems_archetype_name

Get energy systems archetype name

Returns str

property floor_area

Get building floor area in square meters

Returns float

property function → Optional[str]

Get building function

Returns None or str

property grounds → [*Surface*]

Get building ground surfaces

Returns [Surface]

property heating_consumption

Get energy consumption for heating according to the heating system installed in J

return: dict

property heating_demand → dict

Get heating demand in J

Returns dict{[float]}

property heating_peak_load → Optional[dict]

Get heating peak load in W

Returns dict{[float]}

property households → [*Household*]

Get the list of households inside the building

Returns List[Household]

property internal_walls → [*Surface*]

Get building internal wall surfaces

Returns [Surface]

property internal_zones → [*InternalZone*]

Get building internal zones

For Lod up to 3, there is only one internal zone which corresponds to the building shell.

In LoD 4 there can be more than one. In this case the definition of surfaces and floor area must be redefined.

Returns [InternalZone]

property is_conditioned

Get building heated flag

Returns Boolean

property lighting_electrical_demand → dict

Get lighting electrical demand in J

Returns dict{[float]}

property lighting_peak_load → Optional[dict]

Get lighting peak load in W

Returns dict{[float]}

property lower_corner

Get building lower corner.

property onsite_electrical_production

Get total electricity produced onsite in J

return: dict

property pv_generation

temporary attribute to get the onsite pv generation in W

Returns dict

property roof_type

Get roof type for the building flat or pitch

Returns str

property roofs → [*Surface*]

Get building roof surfaces

Returns [Surface]

property shell → *Polyhedron*

Get building's external polyhedron

Returns [Polyhedron]

property storeys_above_ground → Optional[int]

Get building storeys number above ground

Returns None or int

property terrains → Optional[[*Surface*]]
Get city object terrain surfaces

Returns [Surface]

property thermal_zones_from_internal_zones → Optional[[*ThermalZone*]]
Get building thermal zones

Returns [ThermalZone]

property upper_corner
Get building upper corner.

property usages → Optional[list]
Get building usages, if none, assume usage is function

Returns None or list of functions

property walls → [*Surface*]
Get building wall surfaces

Returns [Surface]

property year_of_construction
Get building year of construction

Returns int

2.3.5 Parts Consisting Building

class PartsConsistingBuilding(*name*, *city_objects*)
Inherit: CityObjectsCluster
PartsConsistingBuilding(CityObjectsCluster) class

property city_objects → [*CityObject*]
Get city objects that compose the cluster

Returns [CityObject]

property type
Get type of cluster

Returns str

2.3.6 Buildings Cluster

class BuildingsCluster(*name*, *city_objects*)
Inherit: CityObjectsCluster
BuildingsCluster(CityObjectsCluster) class

property city_objects → [*CityObject*]
Get the list of city objects conforming the cluster

Returns [CityObject]

property type
Get cluster type

Returns str

2.3.7 Network

class Network(*name*, *edges=None*, *nodes=None*)

Inherit: CityObject

Generic network class to be used as base for the network models

property edges → [*Edge*]

Get network edges

Returns [Edge]

property id

Get network id, a universally unique identifier randomly generated

Returns str

property nodes → [*Node*]

Get network nodes

Returns [Node]

2.3.8 Level of detail

class LevelOfDetail

Inherit: :py:class:object

Level of detail for the city class

property construction

Get the city minimal construction level of detail, 1 or 2

Returns int

property energy_systems

Get the city minimal energy systems level of detail, 1 or 2

Returns int

property geometry

Get the city minimal geometry level of detail from 0 to 4

Returns int

property surface_radiation

Get the city minimal surface radiation level of detail, 0 (yearly), 1 (monthly), 2 (hourly)

Returns int

property usage

Get the city minimal usage level of detail, 1 or 2

Returns int

property weather

Get the city minimal weather level of detail, 0 (yearly), 1 (monthly), 2 (hourly)

Returns int

2.3.9 Edge

class Edge(*name*, *nodes=None*)
Inherit: :py:class:object

Generic edge class to be used as base for the network edges

property id

Get edge id, a universally unique identifier randomly generated

Returns str

property name

Get edge name

Returns str

property nodes → [*Node*]

Get delimiting nodes for the edge

Returns [Node]

2.3.10 Node

class Node(*name*, *edges=None*)
Inherit: :py:class:object

Generic node class to be used as base for the network nodes

property edges → [*Edge*]

Get edges delimited by the node

Returns [Edge]

property id

Get node id, a universally unique identifier randomly generated

Returns str

property name

Get node name

Returns str

property time_series → *TimeSeries*

Add explanation here

Returns add type of variable here

2.3.11 Plane

class Plane(*origin*, *normal*)
Inherit: :py:class:object

Plane class

distance_to_point(*point*)

Distance between the given point and the plane

Returns float

property equation()

Get the plane equation components Ax + By + Cz + D = 0

Returns (A, B, C, D)

property normal

Get plane normal [x, y, z]

Returns np.ndarray

property opposite_normal

get plane normal in the opposite direction [x, y, z]

Returns np.ndarray

property origin → *Point*

Get plane origin point

Returns Point

2.3.12 Point

class Point(*coordinates*)

Inherit: :py:class:object

Point class

property coordinates

Get point coordinates

Returns [ndarray]

distance_to_point(*other_point*)

Calculates distance between points in an n-D Euclidean space

Parameter other_point point or vertex

Returns float

2.3.13 Polygon

class Polygon(*coordinates*)

Inherit: :py:class:object

Polygon class

property area

Get surface area in square meters

Returns float

property coordinates → [ndarray]

Get the points in the shape of its coordinates belonging to the polygon [[x, y, z],...]

Returns [np.ndarray]

divide(*plane*)

Divides the polygon in two by a plane

Parameter plane plane that intersects with self to divide it in two parts (Plane)

Returns Polygon, Polygon, [Point]

property edges → [[*Point*]]

Get polygon edges list

Returns [[Point]]

property faces → [[int]]

Get polygon triangular faces

Returns [face]

property inverse

Get the polygon coordinates in reversed order

Returns [np.ndarray]

property normal → ndarray

Get surface normal vector

Returns np.ndarray

property plane → *Plane*

Get the polygon plane

Returns Plane

property points → [*Point*]

Get the points belonging to the polygon [[x, y, z],...]

Returns [Point]

property points_list → ndarray

Get the solid surface point coordinates list [x, y, z, x, y, z,...]

Returns np.ndarray

static triangle_mesh(*vertices*, *normal*) → Trimesh

Get the triangulated mesh for the polygon

Returns Trimesh

property triangles → [*Polygon*]

Triangulate the polygon and return a list of triangular polygons

Returns [Polygon]

property vertices → ndarray

Get polygon vertices

Returns np.ndarray(int)

2.3.14 Polyhedron

class Polyhedron(*polygons*)

Inherit: :py:class:object

Polyhedron class

property centroid → Optional[[float]]

Get polyhedron centroid

Returns [x,y,z]

property faces → [[int]]

Get polyhedron triangular faces

Returns [face]

property max_x

Get polyhedron maximal x value in meters

Returns float

property max_y

Get polyhedron maximal y value in meters

Returns float

property max_z

Get polyhedron maximal z value in meters

Returns float

property min_x

Get polyhedron minimal x value in meters

Returns float

property min_y

Get polyhedron minimal y value in meters

Returns float

property min_z

Get polyhedron minimal z value in meters

Returns float

obj_export(*full_path*)

Export the polyhedron to obj given file

Parameter full_path str

Returns None

show()

Auxiliary function to render the polyhedron

Returns None

`stl_export`(*full_path*)

Export the polyhedron to stl given file

Parameter full_path str

Returns None

`property trimesh` → Optional[Trimesh]

Get polyhedron trimesh

Returns Trimesh

`property vertices` → ndarray

Get polyhedron vertices

Returns np.ndarray(int)

`property volume`

Get polyhedron volume in cubic meters

Returns float

2.3.15 Record

`class Record`(*time=None*, *value=None*, *flag=None*)

`Inherit: :py:class:object`

Record class

`property flag`

Add explanation here

Returns add type of variable here

`property time`

Add explanation here

Returns add type of variable here

`property value`

Add explanation here

Returns add type of variable here

2.3.16 Schedule

`class Schedule`

`Inherit: :py:class:object`

Schedule class

`property data_type` → Optional[str]

Get schedule data type from:

['any_number', 'fraction', 'on_off', 'temperature', 'humidity', 'control_type']

Returns None or str

property day_types → Optional[[str]]

Get schedule day types, as many as needed from:

['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'holiday', 'winter_design_day',

'summer_design_day']

Returns None or [str]

property id

Get schedule id, a universally unique identifier randomly generated

Returns str

property time_range → Optional[str]

Get schedule time range from:

['minute', 'hour', 'day', 'week', 'month', 'year']

Returns None or str

property time_step → Optional[str]

Get schedule time step from:

['second', 'minute', 'hour', 'day', 'week', 'month']

Returns None or str

property type → Optional[str]

Get schedule type

Returns None or str

property values

Get schedule values

Returns [Any]

2.3.17 Time Series

class TimeSeries(*time_series_type=None*, *records=None*)

Inherit: :py:class:object

TimeSeries class

property records → [*Record*]

Add explanation here

Returns List[Record]

property time_series_type

Add explanation here

Returns add type of variable here

2.3.18 Appliances

class Appliances

Inherit: :py:class:object

Appliances class

property convective_fraction → Optional[float]

Get convective fraction

Returns None or float

property density → Optional[float]

Get appliances density in Watts per m2

Returns None or float

property latent_fraction → Optional[float]

Get latent fraction

Returns None or float

property radiative_fraction → Optional[float]

Get radiant fraction

Returns None or float

property schedules → Optional[[*Schedule*]]

Get schedules

dataType = fraction

Returns None or [Schedule]

2.3.19 Household

class Household

Inherit: :py:class:object

Household class

property number_of_cars

Get number of cars owned by the householders

Returns int

property number_of_people

Get number of people leaving in the household

Returns int

2.3.20 Internal Gain

class InternalGain

Inherit: :py:class:object

InternalGain class

property average_internal_gain → Optional[float]

Get internal gains average internal gain in W/m2

Returns None or float

property convective_fraction → Optional[float]

Get internal gains convective fraction

Returns None or float

property latent_fraction → Optional[float]

Get internal gains latent fraction

Returns None or float

property radiative_fraction → Optional[float]

Get internal gains radiative fraction

Returns None or float

property schedules → Optional[[*Schedule*]]

Get internal gain schedules

data type = any number

time step = 1 hour

time range = 1 day

Returns [Schedule]

property type → Optional[str]

Get internal gains type

Returns None or string

2.3.21 Internal Zone

Note: The internal zone class represents each of the internal zones described in the geometry when imported.

This imported geometry can be later on divided in different thermal zones in a workflow (e.g. if the building with no interiors defined (LoD up to 3), it will produce one interior zone. Later on, this can be divided by storey and create one thermal zone per each).

Also, several usages can be associated with that internal zone. This usages are described in the Usage class, which has not only the parameters that describe each usage, but also the percentage of the internal zone volume that is affected by that specific use.

---

class InternalZone(*surfaces*, *area*, *volume*)

Inherit: :py:class:object

InternalZone class

property area

Get internal zone area in square meters

Returns float

property geometry → *Polyhedron*

Get internal zone geometry

Returns Polyhedron

property id

Get internal zone id, a universally unique identifier randomly generated

Returns str

property mean_height

Get internal zone mean height in meters

Returns float

property surfaces

Get internal zone surfaces

Returns [Surface]

property thermal_archetype → ThermalArchetype

Get thermal archetype parameters

Returns ThermalArchetype

property thermal_zones_from_internal_zones → Optional[[*ThermalZone*]]

Get building thermal zones as one per internal zone

Returns [ThermalZone]

property usages → [*Usage*]

Get usage archetypes

Returns [Usage]

property volume

Get internal zone volume in cubic meters

Returns float

2.3.22 Layer

class Layer

Inherit: :py:class:object

Layer class

property conductivity → Optional[float]

Get material conductivity in W/mK

Returns None or float

property density → Optional[float]

Get material density in kg/m3

Returns None or float

property id
Get layer id, a universally unique identifier randomly generated
Returns str

property material_name
Get material name
Returns str

property no_mass → Optional[bool]
Get material no mass flag
Returns None or Boolean

property solar_absorptance → Optional[float]
Get material solar absorptance
Returns None or float

property specific_heat → Optional[float]
Get material conductivity in J/kgK
Returns None or float

property thermal_absorptance → Optional[float]
Get material thermal absorptance
Returns None or float

property thermal_resistance → Optional[float]
Get material thermal resistance in m2K/W
Returns None or float

property thickness → Optional[float]
Get layer thickness in meters
Returns None or float

property visible_absorptance → Optional[float]
Get material visible absorptance
Returns None or float

2.3.23 Lighting

class Lighting
Inherit: :py:class:object
Lighting class

property convective_fraction → Optional[float]
Get convective fraction
Returns None or float

property density → Optional[float]
Get lighting density in Watts per m2
Returns None or float

property latent_fraction → Optional[float]

Get latent fraction

Returns None or float

property radiative_fraction → Optional[float]

Get radiant fraction

Returns None or float

property schedules → Optional[[*Schedule*]]

Get schedules

dataType = fraction

Returns None or [Schedule]

2.3.24 Occupancy

class Occupancy

Inherit: :py:class:object

Occupancy class

property latent_internal_gain → Optional[float]

Get latent internal gain in Watts per m2

Returns None or float

property occupancy_density → Optional[float]

Get density in persons per m2

Returns None or float

property occupancy_schedules → Optional[[*Schedule*]]

Get occupancy schedules

dataType = fraction

Returns None or [Schedule]

property sensible_convective_internal_gain → Optional[float]

Get sensible convective internal gain in Watts per m2

Returns None or float

property sensible_radiative_internal_gain → Optional[float]

Get sensible radiant internal gain in Watts per m2

Returns None or float

2.3.25 Storey

class Storey(*name*, *storey_surfaces*, *neighbours*, *volume*, *internal_zone*, *floor_area*)

Inherit: :py:class:object

Storey class

property floor_area

Get storey's floor area in square meters

Returns float

property name

Get storey's name

Returns str

property neighbours

Get the neighbour storeys' names

Returns [str]

property surfaces → [*Surface*]

Get external surfaces enclosing the storey

Returns [Surface]

property thermal_boundaries → [*ThermalBoundary*]

Get the thermal boundaries bounding the thermal zone

Returns [ThermalBoundary]

property thermal_zone → *ThermalZone*

Get the thermal zone inside the storey

Returns ThermalZone

property virtual_surfaces → [*Surface*]

Get the internal surfaces enclosing the thermal zone

Returns [Surface]

property volume

Get storey's volume in cubic meters

Returns float

2.3.26 Surface

class Surface(*solid_polygon*, *perimeter_polygon*, *holes_polygons=None*, *name=None*, *surface_type=None*)

Inherit: :py:class:object

Surface class

property associated_thermal_boundaries → Optional[[*ThermalBoundary*]]

Get the list of thermal boundaries that has this surface as external face

Returns None or [ThermalBoundary]

property azimuth

Get surface azimuth in radians (north = 0)

Returns float

divide(*z*)

Divides a surface at Z plane

Returns Surface, Surface, Any

property global_irradiance → dict

Get global irradiance on surface in W/m2

Returns dict

property global_irradiance_tilted → dict

Get global irradiance on a tilted surface in W/m2

Returns dict

property holes_polygons → Optional[[*Polygon*]]

Get hole surfaces, a list of hole polygons found in the surface

Returns None, [] or [Polygon]

None -> not known whether holes exist in reality or not due to low level of detail of input data

[] -> no holes in the surface

[Polygon] -> one or more holes in the surface

property id

Get the surface id

Returns str

property inclination

Get surface inclination in radians (zenith = 0, horizon = pi/2)

Returns float

property installed_solar_collector_area

Get installed solar collector area in m2

Returns dict

property inverse → *Surface*

Get the inverse surface (the same surface pointing backwards)

Returns Surface

property long_wave_emittance

Get the long wave emittance af the surface

The thermal absorptance can be calculated as 1-long_wave_emittance

Returns float

property lower_corner

Get surface's lower corner [x, y, z]

Returns [float]

property name

Get the surface name

Returns str

property percentage_shared

Get percentage of the wall shared with other walls

Returns float

property perimeter_area

Get perimeter surface area in square meters (opaque + transparent)

Returns float

property perimeter_polygon → *Polygon*

Get a polygon surface defined by the perimeter, merging solid and holes

Returns Polygon

property short_wave_reflectance

Get the short wave reflectance, this includes all solar spectrum, visible and not visible

The absorptance as an opaque surface, can be calculated as 1-short_wave_reflectance

Returns float

property solar_collectors_area_reduction_factor

Get factor area collector per surface area if set or calculate using Romero Rodriguez, L. et al (2017) model if not

Returns float

property solid_polygon → *Polygon*

Get the solid surface

Returns Polygon

property type

Get surface type Ground, Ground wall, Wall, Attic floor, Interior slab, Interior wall, Roof or Virtual internal

If the geometrical LoD is lower than 4,

the surfaces' types are not defined in the importer and can only be Ground, Wall or Roof

Returns str

property upper_corner

Get surface's upper corner [x, y, z]

Returns [float]

property vegetation → Optional[*Vegetation*]

Get the vegetation construction at the external surface of the thermal boundary

Returns None or Vegetation

2.3.27 Thermal Boundary

class ThermalBoundary(*parent_surface*, *opaque_area*, *windows_areas*)

Inherit: :py:class:object

ThermalBoundary class

property construction_name

Get construction name

Returns str

property he → Optional[float]

Get external convective heat transfer coefficient (W/m2K)

Returns None or float

property hi → Optional[float]

Get internal convective heat transfer coefficient (W/m2K)

Returns None or float

property id

Get thermal zone id, a universally unique identifier randomly generated

Returns str

property internal_surface → *Surface*

Get the internal surface of the thermal boundary

Returns Surface

property layers → [*Layer*]

Get thermal boundary layers

Returns [Layers]

property opaque_area

Get the thermal boundary area in square meters

Returns float

property parent_surface → *Surface*

Get the surface that belongs to the thermal boundary, considered the external surface of that boundary

Returns Surface

property thermal_openings → Optional[[*ThermalOpening*]]

Get thermal boundary thermal openings

Returns None or [ThermalOpening]

property thermal_zones → [*ThermalZone*]

Get the thermal zones delimited by the thermal boundary

Returns [ThermalZone]

property thickness

Get the thermal boundary thickness in meters

Returns float

property type

Get thermal boundary surface type

Returns str

property u_value → Optional[float]

Get thermal boundary U-value in W/m2K

internal and external convective coefficient in W/m2K values, can be configured at configuration.ini

Returns None or float

property window_ratio → Optional[float]

Get thermal boundary window ratio

It returns the window ratio calculated as the total windows' areas in a wall divided by

the total (opaque + transparent) area of that wall if windows are defined in the geometry imported.

If not, it returns the window ratio imported from an external source (e.g. construction library, manually assigned).

If none of those sources are available, it returns None.

Returns float

property windows_areas → [float]

Get windows areas

Returns [float]

2.3.28 Thermal Control

class ThermalControl

Inherit: :py:class:object

ThermalControl class

property cooling_set_point_schedules → Optional[[*Schedule*]]

Get cooling set point schedule defined for a thermal zone in Celsius

dataType = temperature

Returns None or [Schedule]

property heating_set_back → Optional[float]

Get heating set back defined for a thermal zone in Celsius

Returns None or float

property heating_set_point_schedules → Optional[[*Schedule*]]

Get heating set point schedule defined for a thermal zone in Celsius

dataType = temperature

Returns None or [Schedule]

property hvac_availability_schedules → Optional[[*Schedule*]]

Get the availability of the conditioning system defined for a thermal zone

dataType = on/off

Returns None or [Schedule]

property mean_cooling_set_point → Optional[float]

Get cooling set point defined for a thermal zone in Celsius

Returns None or float

property mean_heating_set_point → Optional[float]

Get heating set point defined for a thermal zone in Celsius

Returns None or float

2.3.29 Thermal Opening

class ThermalOpening

Inherit: :py:class:object

ThermalOpening class

property area → Optional[float]

Get thermal opening area in square meters

Returns None or float

property conductivity → Optional[float]

Get thermal opening conductivity in W/mK

Returns None or float

property construction_name

Get thermal opening construction name

property frame_ratio → Optional[float]

Get thermal opening frame ratio

Returns None or float

property g_value → Optional[float]

Get thermal opening transmittance at normal incidence

Returns None or float

property he → Optional[float]

Get external convective heat transfer coefficient (W/m2K)

Returns None or float

property hi → Optional[float]

Get internal convective heat transfer coefficient (W/m2K)

Returns None or float

property id

Get thermal zone id, a universally unique identifier randomly generated

Returns str

property overall_u_value → Optional[float]

Get thermal opening overall U-value in W/m2K

Returns None or float

property thickness → Optional[float]

Get thermal opening thickness in meters

Returns None or float

2.3.30 Thermal Zone

class ThermalZone(*thermal_boundaries*, *parent_internal_zone*, *volume*, *footprint_area*, *number_of_storeys*, *usages=None*)

Inherit: :py:class:object

ThermalZone class

property additional_thermal_bridge_u_value → Optional[float]

Get thermal zone additional thermal bridge u value per footprint area W/m2K

Returns None or float

property appliances → Optional[*Appliances*]

Get appliances information

Returns None or Appliances

property days_year → Optional[float]

Get thermal zone usage days per year

Returns None or float

property domestic_hot_water → Optional[*DomesticHotWater*]

Get domestic hot water information of this thermal zone

Returns None or DomesticHotWater

property effective_thermal_capacity → Optional[float]

Get thermal zone effective thermal capacity in J/m3K

Returns None or float

property footprint_area → float

Get thermal zone footprint area in m2

Returns float

property hours_day → Optional[float]

Get thermal zone usage hours per day

Returns None or float

property id

Get thermal zone id, a universally unique identifier randomly generated

Returns str

property indirectly_heated_area_ratio → Optional[float]

Get thermal zone indirectly heated area ratio

Returns None or float

property infiltration_rate_area_system_off

Get thermal zone infiltration rate system off in air changes per second (1/s)

Returns None or float

property infiltration_rate_area_system_on

Get thermal zone infiltration rate system on in air changes per second (1/s)

Returns None or float

property infiltration_rate_system_off

Get thermal zone infiltration rate system off in air changes per second (1/s)

Returns None or float

property infiltration_rate_system_on

Get thermal zone infiltration rate system on in air changes per second (1/s)

Returns None or float

property internal_gains → Optional[[*InternalGain*]]

Calculates and returns the list of all internal gains defined

Returns [InternalGain]

property lighting → Optional[*Lighting*]

Get lighting information

Returns None or Lighting

property mechanical_air_change → Optional[float]

Get thermal zone mechanical air change in air change per second (1/s)

Returns None or float

property occupancy → Optional[*Occupancy*]

Get occupancy in the thermal zone

Returns None or Occupancy

property ordinate_number → Optional[int]

Get the order in which the thermal_zones need to be enumerated

Returns None or int

property parent_internal_zone → *InternalZone*

Get the internal zone to which this thermal zone belongs

Returns InternalZone

property thermal_boundaries → [*ThermalBoundary*]

Get thermal boundaries bounding with the thermal zone

Returns [ThermalBoundary]

property thermal_control → Optional[*ThermalControl*]

Get thermal control of this thermal zone

Returns None or ThermalControl

property total_floor_area

Get the total floor area of this thermal zone in m2

Returns float

property usage_name → Optional[str]

Get thermal zone usage name

Returns None or str

property usages

Get the thermal zone usages

Returns str

property view_factors_matrix

Get thermal zone view factors matrix

Returns [[float]]

property volume

Get thermal zone volume

Returns float

2.3.31 Usage

class Usage

Inherit: :py:class:object

Usage class

property appliances → Optional[*Appliances*]

Get appliances information

Returns None or Appliances

property days_year → Optional[float]

Get usage zone usage days per year

Returns None or float

property domestic_hot_water → Optional[*DomesticHotWater*]

Get domestic hot water information

Returns None or ThermalControl

property hours_day → Optional[float]

Get usage zone usage hours per day

Returns None or float

property id

Get usage zone id, a universally unique identifier randomly generated

Returns str

property internal_gains → [*InternalGain*]

Calculates and returns the list of all internal gains defined

Returns InternalGains

property lighting → Optional[*Lighting*]

Get lighting information

Returns None or Lighting

property mechanical_air_change → Optional[float]

Get usage zone mechanical air change in air change per second (1/s)

Returns None or float

property name → Optional[str]

Get usage zone usage

Returns None or str

property occupancy → Optional[*Occupancy*]

Get occupancy in the usage zone

Returns None or Occupancy

property percentage

Get usage zone percentage in range[0,1]

Returns float

property thermal_control → Optional[*ThermalControl*]

Get thermal control information

Returns None or ThermalControl

2.3.32 Plant

class Plant(*name*, *height*, *leaf_area_index*, *leaf_reflectivity*, *leaf_emissivity*, *minimal_stomatal_resistance*, *co2_sequestration*, *grows_on_soils*)

Inherit: :py:class:object

Plant class

property co2_sequestration

Get plant co2 sequestration capacity in kg CO2 equivalent

Returns float

property grows_on → [*Soil*]

Get plant compatible soils

Returns [Soil]

property height

Get plant height in m

Returns float

property leaf_area_index

Get plant leaf area index

Returns float

property leaf_emissivity

Get plant leaf emissivity

Returns float

property leaf_reflectivity

Get plant leaf area index

Returns float

property minimal_stomatal_resistance

Get plant minimal stomatal resistance in s/m

Returns float

property name

Get plant name

Returns string

property percentage

Get percentage of plant in vegetation

Returns float

2.3.33 Soil

class Soil(*name*, *roughness*, *dry_conductivity*, *dry_density*, *dry_specific_heat*, *thermal_absorptance*, *solar_absorptance*, *visible_absorptance*, *saturation_volumetric_moisture_content*, *residual_volumetric_moisture_content*)

Inherit: :py:class:object

Soil class

property dry_conductivity

Get soil dry conductivity in W/mK

Returns float

property dry_density

Get soil dry density in kg/m3

Returns float

property dry_specific_heat

Get soil dry specific heat in J/kgK

Returns float

property initial_volumetric_moisture_content

Get soil initial volumetric moisture content

Returns None or float

property name

Get soil name

Returns string

property residual_volumetric_moisture_content

Get soil residual volumetric moisture content

Returns None or float

property roughness

Get soil roughness

Returns string

property saturation_volumetric_moisture_content

Get soil saturation volumetric moisture content

Returns float

property solar_absorptance

Get soil solar absortance

Returns float

property thermal_absorptance

Get soil thermal absortance

Returns float

property visible_absorptance

Get soil visible absortance

Returns float

2.3.34 Vegetation

class Vegetation(*name*, *soil*, *soil_thickness*, *plants*)

Inherit: :py:class:object

Vegetation class

property air_gap

Get air gap in m

Returns float

property management

Get management

Returns string

property name

Get vegetation name

Returns string

property plants → [*Plant*]

Get list plants in the vegetation

Returns List[Plant]

property soil → *Soil*

Get soil

Returns Soil

property soil_thickness

Get soil thickness in m

Returns float

2.3.35 Sensor

class Sensor

Inherit: :py:class:object

Sensor abstract class

property location → *Location*

Get sensor location

Returns Location

property measures → [*SensorMeasure*]

Raises not implemented error

property name

Get sensor name

Returns str

property type → <enum 'Sensor'>

Get sensor type

Returns str

property units

Get sensor units

Returns str

2.3.36 Sensor Measure

class SensorMeasure(*latitude*, *longitude*, *utc_timestamp*, *value*)

Inherit: :py:class:object

Sensor measure class

property latitude

Get measure latitude

property longitude

Get measure longitude

property utc_timestamp

Get measure timestamp in utc

property value

Get sensor measure value

2.3.37 Sensor Type

class SensorType(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Inherit: Enum

Sensor type enumeration

2.3.38 Station

class Station(*station_id=None*, *_mobile=False*)

Inherit: :py:class:object

Station class

property id

Get the station id a random uuid will be assigned if no ID was provided to the constructor

Returns ID

property mobile

Get if the station is mobile or not

Returns bool

property sensors → [*Sensor*]

Get the sensors belonging to the station

Returns [Sensor]

CHAPTER

THREE

FACTORIES

Factories are divided into Imports and Exports, depending on if they are used to enrich (Import) the *city model structure* or to deliver third party defined formats (Export) such as **INSEL** or **IDF** file, the factories could be extended to include new imports and outputs providing an additional level of abstraction to researchers.

⚠ Please, note that the private methods, the ones starting with an underscore character (_), documented in the factories are mean to be called by using the **handler** parameter; this parameter must contain the method name without the _ character.

Note: For instance, to use _citygml handler in the Geometry factory, the handler parameter value needs to be 'citygml'

⚠ **This documentation includes only the base factories classes as these are the intended entry points for the Import/Export functionality.**

Important: Please refer to the development manual if you want to create your own factories.

[development manual]

3.1 Folder structure

3.1.1 Imports

```
../hub/hub/imports/
├── energy_systems
│   ├── __init__.py
│   ├── montreal_custom_energy_system_parameters.py
│   ├── montreal_future_energy_systems_parameters.py
│   ├── north_america_custom_energy_system_parameters.py
│   └── palma_energy_systems_parameters.py
├── results
│   ├── energy_plus.py
│   ├── energy_plus_single_building.py
│   ├── ep_multiple_buildings.py
│   ├── __init__.py
│   ├── insel_monthly_energy_balance.py
│   └── simplified_radiosity_algorithm.py
├── construction_factory.py
├── energy_systems_factory.py
├── geometry_factory.py
├── __init__.py
├── results_factory.py
├── usage_factory.py
└── weather_factory.py

3 directories, 18 files
```

3.1.2 Exports

```
../hub/hub/exports/
├── building_energy
│   ├── idf_files
│   │   ├── base.idf
│   │   ├── Energy+.idd
│   │   ├── Minimal.idf
│   │   └── outputs.idf
│   ├── idf_helper
│   │   ├── idf_appliance.py
│   │   ├── idf_base.py
│   │   ├── idf_construction.py
│   │   ├── idf_dhw.py
│   │   ├── idf_file_schedule.py
│   │   ├── idf_heating_system.py
│   │   ├── idf_infiltration.py
│   │   ├── idf_lighting.py
│   │   ├── idf_material.py
│   │   ├── idf_occupancy.py
│   │   ├── idf_schedule.py
│   │   ├── idf_shading.py
│   │   ├── idf_surfaces.py
│   │   ├── idf_thermostat.py
│   │   ├── idf_ventilation.py
│   │   ├── idf_window.py
│   │   ├── idf_windows_constructions.py
│   │   ├── idf_windows_material.py
│   │   ├── idf_zone.py
│   │   └── __init__.py
│   ├── insel
│   │   ├── __init__.py
│   │   └── insel_monthly_energy_balance.py
│   ├── cerc_idf.py
│   ├── energy_ade.py
│   ├── idf.py
│   └── __init__.py
├── energy_systems
│   ├── air_source_hp_export.py
│   ├── heat_pump_export.py
│   └── water_to_water_hp_export.py
├── energy_building_exports_factory.py
├── energy_systems_factory.py
├── exports_factory.py
└── __init__.py

6 directories, 37 files
```

3.2 Imports Classes

3.2.1 Construction Factory

class ConstructionFactory(*handler*, *city*)

Inherit: :py:class:object

ConstructionFactory class

_eilat()

Enrich the city by using Eilat information

_nrcan()

Enrich the city by using NRCAN information

_nrel()

Enrich the city by using NREL information

_palma()

Enrich the city by using Palma information

enrich()

Enrich the city given to the class using the class given handler

Returns None

3.2.2 Geometry Factory

class GeometryFactory(*file_type*, *path=None*, *aliases_field=None*, *height_field=None*, *year_of_construction_field=None*, *function_field=None*, *function_to_hub=None*, *usages_field=None*, *usages_to_hub=None*, *hub_crs=None*)

Inherit: :py:class:object

GeometryFactory class

property _citygml → *City*

Enrich the city by using CityGML information as data source

Returns City

property _geojson → *City*

Enrich the city by using Geojson information as data source

Returns City

property _obj → *City*

Enrich the city by using OBJ information as data source

Returns City

property city → *City*

Enrich the city given to the class using the class given handler

Returns City

3.2.3 Usage Factory

class UsageFactory(*handler*, *city*)

Inherit: :py:class:object

UsageFactory class

_comnet()

Enrich the city with COMNET usage library

_eilat()

Enrich the city with Eilat usage library

_nrcan()

Enrich the city with NRCAN usage library

_palma()

Enrich the city with Palma usage library

enrich()

Enrich the city given to the class using the usage factory given handler

Returns None

3.2.4 Weather Factory

class WeatherFactory(*handler*, *city: City*)

Inherit: :py:class:object

WeatherFactory class

_epw()

Enrich the city with energy plus weather file

enrich()

Enrich the city given to the class using the given weather handler

Returns None

3.3 Exports

3.3.1 Export Factory

class ExportsFactory(*handler*, *city*, *path*, *target_buildings=None*, *adjacent_buildings=None*, *base_uri=None*)

Inherit: :py:class:object

Exports factory class

property _cesiumjs_tileset

Export the city to a cesiumJs tileset format

Returns None

property _obj

Export the city geometry to obj

Returns None

property _sra

Export the city to Simplified Radiosity Algorithm xml format

Returns None

property _stl

Export the city geometry to stl

Returns None

export()

Export the city given to the class using the given export type handler

Returns None

CHAPTER

FOUR

CATALOGS

In its simplest form, a catalogue is a file or group of files that provide technical and/or commercial information regarding components that form a system within any domain. The components are listed with relevant details and associated data is tabulated. Also listed are the dominant/standard configurations in which the components may be used to satisfy use-cases/output requirements (as supplied by component manufacturer/standard organisations).

Note: Examples, Heat Pump catalogue should consist of the heat pump models produced, heat pump type, manufacturer name, output temperatures, nominal capacities, typical configurations for the heat pumps (e.g., configurations when used for space heating only, Domestic Hot Water/DHW purposes only, both space heating and DHW, combinations with solar thermal/PV), storage tank data, circulation pump data, compressor type and associated technical data, valve types etc.

Important: Please refer to the catalogs manual if you want to create or extend your own catalogs.

[catalogs manual]

4.1 Folder structure

4.1.1 Catalogs

```
../hub/hub/catalog_factories/
├── construction
│   ├── construction_helper.py
│   ├── eilat_catalog.py
│   ├── __init__.py
│   ├── nrcan_catalog.py
│   ├── nrel_catalog.py
│   └── palma_catalog.py
├── cost
│   ├── __init__.py
│   └── montreal_custom_catalog.py
├── data_models
│   ├── construction
│   │   ├── archetype.py
│   │   ├── construction.py
│   │   ├── content.py
│   │   ├── __init__.py
│   │   ├── layer.py
│   │   ├── material.py
│   │   └── window.py
│   ├── cost
│   │   ├── archetype.py
│   │   ├── capital_cost.py
│   │   ├── chapter.py
│   │   ├── content.py
│   │   ├── fuel.py
│   │   ├── income.py
│   │   ├── __init__.py
│   │   ├── item_description.py
│   │   └── operational_cost.py
│   ├── energy_systems
│   │   ├── archetype.py
│   │   ├── content.py
│   │   ├── distribution_system.py
│   │   ├── electrical_storage_system.py
│   │   ├── emission_system.py
│   │   ├── energy_storage_system.py
│   │   ├── generation_system.py
│   │   ├── __init__.py
│   │   ├── non_pv_generation_system.py
│   │   ├── performance_curves.py
│   │   ├── pv_generation_system.py
│   │   ├── system.py
│   │   └── thermal_storage_system.py
│   ├── greenery
│   │   ├── content.py
│   │   ├── __init__.py
│   │   ├── plant_percentage.py
│   │   ├── plant.py
│   │   ├── soil.py
│   │   └── vegetation.py
│   ├── usages
│   │   ├── appliances.py
│   │   ├── content.py
│   │   ├── domestic_hot_water.py
│   │   ├── __init__.py
│   │   ├── lighting.py
│   │   ├── occupancy.py
│   │   ├── schedule.py
│   │   ├── thermal_control.py
│   │   └── usage.py
│   └── __init__.py
├── energy_systems
│   ├── __init__.py
│   ├── montreal_custom_catalog.py
│   ├── montreal_future_system_catalogue.py
│   └── palma_system_catalgue.py
├── greenery
│   ├── ecore_greenery
│   │   ├── greenerycatalog.ecore
│   │   ├── greenerycatalog_no_quantities.ecore
│   │   └── greenerycatalog.py
│   ├── greenery_catalog.py
│   └── __init__.py
├── usage
│   ├── comnet_catalog.py
│   ├── eilat_catalog.py
│   ├── __init__.py
│   ├── nrcan_catalog.py
│   ├── palma_catalog.py
│   └── usage_helper.py
├── catalog.py
├── construction_catalog_factory.py
├── costs_catalog_factory.py
├── energy_systems_catalog_factory.py
├── greenery_catalog_factory.py
├── __init__.py
└── usage_catalog_factory.py

13 directories, 75 files
```

4.2 Catalog Base Class

4.2.1 Catalog

class Catalog

Inherit: :py:class:object

Catalogs base class catalog_factories will inherit from this class.

entries(*category=None*)

Base property to return the catalog entries

Returns Catalog content filter by category if provided

get_entry(*name*)

Base property to return the catalog entry matching the given name

Returns Catalog entry with the matching name

names(*category=None*)

Base property to return the catalog entries names.

Returns Catalog names filter by category if provided

4.3 Greenery

4.3.1 Greenery Catalog Factory

class GreeneryCatalogFactory(*handler*, *base_path=None*)

Inherit: :py:class:object

GreeneryCatalogFactory class

property _nrel

Return a greenery catalog based in NREL using ecore as datasource

Returns GreeneryCatalog

property catalog → *Catalog*

Enrich the city given to the class using the class given handler

Returns Catalog

4.3.2 Greenery Content Data Model

class Content(*vegetations*, *plants*, *soils*)

Inherit: :py:class:object

Content class

property plants

All plants in the catalog

property soils

All soils in the catalog

to_dictionary()

Class content to dictionary

property vegetations

All vegetation in the catalog

4.3.3 Greenery Plant Data Model

class Plant(*category*, *plant*)

Inherit: :py:class:object

Plant class

property category

Get plant category name

Returns string

property co2_sequestration

Get plant co2 sequestration capacity in kg CO2 equivalent

Returns float

property grows_on → [*Soil*]

Get plant compatible soils

Returns [Soil]

property height

Get plant height in m

Returns float

property leaf_area_index

Get plant leaf area index

Returns float

property leaf_emissivity

Get plant leaf emissivity

Returns float

property leaf_reflectivity

Get plant leaf area index

Returns float

property minimal_stomatal_resistance

Get plant minimal stomatal resistance in s/m

Returns float

property name

Get plant name

Returns string

to_dictionary()

Class content to dictionary

4.3.4 Greenery Plant Percentage Data Model

class PlantPercentage(*percentage*, *plant_category*, *plant*)

Inherit: Plant

Plant percentage class

property percentage

Get plant percentage

Returns float

to_dictionary()

Class content to dictionary

4.3.5 Greenery Plant Soil Data Model

class Soil(*soil*)
Inherit: :py:class:object

Soil class

property dry_conductivity

Get soil dry conductivity in W/mK

Returns float

property dry_density

Get soil dry density in kg/m3

Returns float

property dry_specific_heat

Get soil dry specific heat in J/kgK

Returns float

property initial_volumetric_moisture_content

Get soil initial volumetric moisture content

Returns float

property name

Get soil name

Returns string

property residual_volumetric_moisture_content

Get soil residual volumetric moisture content

Returns float

property roughness

Get soil roughness

Returns string

property saturation_volumetric_moisture_content

Get soil saturation volumetric moisture content

Returns float

property solar_absorptance

Get soil solar absortance

Returns float

property thermal_absorptance

Get soil thermal absortance

Returns float

to_dictionary()

Class content to dictionary

property visible_absorptance

Get soil visible absortance

Returns float

4.3.6 Greenery Vegetation Data Model

class Vegetation(*category*, *vegetation*, *plant_percentages*)

Inherit: :py:class:object

Vegetation class

property air_gap

Get air gap in m

Returns float

property category

Get vegetation category

Returns string

property dry_soil_conductivity

Get soil dry conductivity in W/mK

Returns float

property dry_soil_density

Get soil dry density in kg/m3

Returns float

property dry_soil_specific_heat

Get soil dry specific heat in J/kgK

Returns float

property management

Get management

Returns string

property name

Get vegetation name

Returns string

property plant_percentages → [*PlantPercentage*]

Get plant percentages

Returns [PlantPercentage]

property soil_initial_volumetric_moisture_content

Get soil initial volumetric moisture content

Returns float

property soil_name

Get soil name

Returns string

property soil_residual_volumetric_moisture_content

Get soil residual volumetric moisture content

Returns float

property soil_roughness

Get soil roughness

Returns float

property soil_saturation_volumetric_moisture_content

Get soil saturation volumetric moisture content

Returns float

property soil_solar_absorptance

Get soil solar absortance

Returns float

property soil_thermal_absorptance

Get soil thermal absortance

Returns float

property soil_thickness

Get soil thickness in m

Returns float

property soil_visible_absorptance

Get soil visible absortance

Returns float

to_dictionary()

Class content to dictionary

4.4 Construction

4.4.1 Construction Catalog Factory

class ConstructionCatalogFactory(*handler*, *base_path=None*)

Inherit: :py:class:object

Construction catalog factory class

property _eilat

Retrieve Eilat catalog

property _nrcan

Retrieve NRCAN catalog

property _nrel

Retrieve NREL catalog

property _palma

Retrieve Palma catalog

property catalog → *Catalog*

Enrich the city given to the class using the class given handler

Returns Catalog

4.4.2 Construction Content Data Model

class Content(*archetypes*, *constructions*, *materials*, *windows*)

Inherit: :py:class:object

Content class

property archetypes

All archetypes in the catalog

property constructions

All constructions in the catalog

property materials

All materials in the catalog

to_dictionary()

Class content to dictionary

property windows

All windows in the catalog

4.4.3 Construction Archetype Data Model

class Archetype(*archetype_id*, *name*, *function*, *climate_zone*, *construction_period*, *constructions*, *average_storey_height*, *thermal_capacity*, *extra_loses_due_to_thermal_bridges*, *indirect_heated_ratio*, *infiltration_rate_for_ventilation_system_off*, *infiltration_rate_for_ventilation_system_on*, *infiltration_rate_area_for_ventilation_system_off*, *infiltration_rate_area_for_ventilation_system_on*)

Inherit: :py:class:object

Archetype class

property average_storey_height

Get archetype average storey height in m

Returns float

property climate_zone

Get archetype climate zone

Returns str

property construction_period

Get archetype construction period

Returns str

property constructions → [*Construction*]

Get archetype constructions

Returns [Construction]

property extra_loses_due_to_thermal_bridges

Get archetype extra loses due to thermal bridges in W/m2K

Returns float

property function

Get archetype function

Returns str

property id

Get archetype id

Returns str

property indirect_heated_ratio

Get archetype indirect heated area ratio

Returns float

property infiltration_rate_area_for_ventilation_system_off

Get archetype infiltration rate for ventilation system off in m3/sm2

Returns float

property infiltration_rate_area_for_ventilation_system_on

Get archetype infiltration rate for ventilation system on in m3/sm2

Returns float

property infiltration_rate_for_ventilation_system_off
Get archetype infiltration rate for ventilation system off in 1/s

Returns float

property infiltration_rate_for_ventilation_system_on
Get archetype infiltration rate for ventilation system on in 1/s

Returns float

property name
Get archetype name

Returns str

property thermal_capacity
Get archetype thermal capacity in J/m3K

Returns float

to_dictionary()
Class content to dictionary

4.4.4 Construction Construction Data Model

class Construction(*construction_id*, *construction_type*, *name*, *layers*, *window_ratio=None*, *window=None*)
Inherit: :py:class:object
Construction class

property id
Get construction id

Returns str

property layers → [*Layer*]
Get construction layers

Returns [layer]

property name
Get construction name

Returns str

to_dictionary()
Class content to dictionary

property type
Get construction type

Returns str

property window → *Window*
Get construction window

Returns Window

property window_ratio
Get construction window ratio

Returns dict

4.4.5 Construction Layer Data Model

class Layer(*layer_id*, *name*, *material*, *thickness*)

Inherit: :py:class:object

Layer class

property id

Get layer id

Returns str

property material → *Material*

Get layer material

Returns Material

property name

Get layer name

Returns str

property thickness

Get layer thickness in meters

Returns None or float

to_dictionary()

Class content to dictionary

4.4.6 Construction Material Data Model

class Material(*material_id*, *name*, *solar_absorptance*, *thermal_absorptance*, *visible_absorptance*, *no_mass=False*, *thermal_resistance=None*, *conductivity=None*, *density=None*, *specific_heat=None*)

Inherit: :py:class:object

Material class

property conductivity

Get material conductivity in W/mK

Returns None or float

property density

Get material density in kg/m3

Returns None or float

property id

Get material id

Returns str

property name

Get material name

Returns str

property no_mass
Get material no mass flag
Returns None or Boolean

property solar_absorptance
Get material solar absorptance
Returns None or float

property specific_heat
Get material conductivity in J/kgK
Returns None or float

property thermal_absorptance
Get material thermal absorptance
Returns None or float

property thermal_resistance
Get material thermal resistance in m2K/W
Returns None or float

to_dictionary()
Class content to dictionary

property visible_absorptance
Get material visible absorptance
Returns None or float

4.4.7 Construction Window Data Model

class Window(*window_id*, *frame_ratio*, *g_value*, *overall_u_value*, *name*, *window_type=None*)
Inherit: :py:class:object
Window class

property frame_ratio
Get window frame ratio
Returns float

property g_value
Get thermal opening g-value
Returns float

property id
Get window id
Returns str

property name
Get window name
Returns str

property overall_u_value

Get thermal opening overall U-value in W/m2K

Returns float

to_dictionary()

Class content to dictionary

property type

Get transparent surface type, 'window' or 'skylight'

Returns str

4.5 Costs

4.5.1 Costs Catalog Factory

class CostsCatalogFactory(*file_type*, *base_path=None*)

Inherit: :py:class:object

CostsCatalogFactory class

property _montreal_custom

Retrieve Montreal Custom catalog

property catalog → *Catalog*

Return a cost catalog

Returns CostCatalog

4.5.2 Costs Content Data Model

class Content(*archetypes*)

Inherit: :py:class:object

Content class

property archetypes

All archetypes in the catalog

to_dictionary()

Class content to dictionary

4.5.3 Costs Archetype Data Model

class Archetype(*lod*, *function*, *municipality*, *country*, *currency*, *capital_cost*, *operational_cost*, *end_of_life_cost*, *income*)

Inherit: :py:class:object

Archetype class

property capital_cost → *CapitalCost*

Get capital cost

Returns CapitalCost

property country

Get country

Returns string

property currency

Get currency

Returns string

property end_of_life_cost

Get end of life cost in given currency per m2

Returns float

property function
Get function

Returns string

property income → *Income*
Get income

Returns Income

property lod
Get level of detail of the catalog

Returns string

property municipality
Get municipality

Returns string

property name
Get name

Returns string

property operational_cost → *OperationalCost*
Get operational cost

Returns OperationalCost

to_dictionary()
Class content to dictionary

4.5.4 Costs Capital Cost Data Model

class CapitalCost(*general_chapters*, *design_allowance*, *overhead_and_profit*)
Inherit: :py:class:object
Capital cost class

chapter(*name*) → *Chapter*
Get specific chapter by name

Returns Chapter

property design_allowance
Get design allowance in percentage (-)

Returns float

property general_chapters → [*Chapter*]
Get general chapters in capital costs

Returns [Chapter]

property overhead_and_profit
Get overhead profit in percentage (-)

Returns float

to_dictionary()
Class content to dictionary

4.5.5 Costs Chapter Data Model

class Chapter(*chapter_type*, *items*)

Inherit: :py:class:object

Chapter class

property chapter_type

Get chapter type

Returns str

item(*name*) → *ItemDescription*

Get specific item by name

Returns ItemDescription

property items → [*ItemDescription*]

Get list of items contained in the chapter

Returns [str]

to_dictionary()

Class content to dictionary

4.5.6 Costs Fuel Data Model

class Fuel(*fuel_type*, *fixed_monthly=None*, *fixed_power=None*, *variable=None*, *variable_units=None*)

Inherit: :py:class:object

Fuel class

property fixed_monthly → Optional[float]

Get fixed operational costs in currency per month

Returns None or float

property fixed_power → Optional[float]

Get fixed operational costs depending on the peak power consumed in currency per month per W

Returns None or float

to_dictionary()

Class content to dictionary

property type

Get fuel type

Returns str

property variable → Optional[tuple[None], tuple[float, str]]

Get variable costs in given units

Returns None, None or float, str

4.5.7 Costs Income Data Model

class Income(*construction_subsidy=None*, *hvac_subsidy=None*, *photovoltaic_subsidy=None*, *electricity_export=None*, *reductions_tax=None*)

Inherit: :py:class:object

Income class

property construction_subsidy → Optional[float]

Get subsidy for construction in percentage %

Returns None or float

property electricity_export → Optional[float]

Get electricity export incomes in currency per J

Returns None or float

property hvac_subsidy → Optional[float]

Get subsidy for HVAC system in percentage %

Returns None or float

property photovoltaic_subsidy → Optional[float]

Get subsidy PV systems in percentage

Returns None or float

property reductions_tax → Optional[float]

Get reduction in taxes in percentage (-)

Returns None or float

to_dictionary()

Class content to dictionary

4.5.8 Costs Item Description Data Model

class ItemDescription(*item_type*, *initial_investment=None*, *initial_investment_unit=None*, *refurbishment=None*, *refurbishment_unit=None*, *reposition=None*, *reposition_unit=None*, *lifetime=None*)

Inherit: :py:class:object

Item description class

property initial_investment → Optional[tuple[None], tuple[float, str]]

Get initial investment of the specific item in given units

Returns None, None or float, str

property lifetime → Optional[float]

Get lifetime in years

Returns None or float

property refurbishment → Optional[tuple[None], tuple[float, str]]

Get refurbishment costs of the specific item in given units

Returns None, None or float, str

property reposition → Optional[tuple[None], tuple[float, str]]

Get reposition costs of the specific item in given units

Returns None, None or float, str

to_dictionary()

Class content to dictionary

property type

Get item type

Returns str

4.5.9 Costs Operational Cost Data Model

class OperationalCost(*fuels*, *maintenance_heating*, *maintenance_cooling*, *maintenance_pv*, *co2*)

Inherit: :py:class:object

Operational cost class

property co2

Get cost of CO2 emissions in currency/kgCO2

Returns float

property fuels → [*Fuel*]

Get fuels listed in capital costs

Returns [Fuel]

property maintenance_cooling

Get cost of maintaining the cooling system in currency/W

Returns float

property maintenance_heating

Get cost of maintaining the heating system in currency/W

Returns float

property maintenance_pv

Get cost of maintaining the PV system in currency/m2

Returns float

to_dictionary()

Class content to dictionary

4.6 Energy Systems

4.6.1 Energy Systems Catalog Factory

class EnergySystemsCatalogFactory(*handler*, *base_path=None*)
Inherit: :py:class:object

Energy system catalog factory class

property _montreal_custom

Retrieve NRCAN catalog

property _montreal_future

Retrieve North American catalog

property _palma

Retrieve Palma catalog

property catalog → *Catalog*

Enrich the city given to the class using the class given handler

Returns Catalog

4.6.2 Energy Systems Content Data Model

class Content(*archetypes*, *systems*, *generations=None*, *distributions=None*)
Inherit: :py:class:object

Content class

property archetypes

All archetype system clusters in the catalog

property distribution_equipments

All distribution equipments in the catalog

property generation_equipments

All generation equipments in the catalog

property systems

All systems in the catalog

to_dictionary()

Class content to dictionary

4.6.3 Energy Systems Archetype Data Model

class Archetype(*name*, *systems*, *archetype_cluster_id=None*)
Inherit: :py:class:object

Archetype class

property cluster_id

Get id

Returns string

property name

Get name

Returns string

property systems → [*System*]

Get list of equipments that compose the total energy system

Returns [Equipment]

to_dictionary()

Class content to dictionary

4.6.4 Energy Systems Distribution System Data Model

class DistributionSystem(*system_id*, *model_name=None*, *system_type=None*, *supply_temperature=None*, *distribution_consumption_fix_flow=None*, *distribution_consumption_variable_flow=None*, *heat_losses=None*, *generation_systems=None*, *energy_storage_systems=None*, *emission_systems=None*)

Inherit: :py:class:object

Distribution system class

property distribution_consumption_fix_flow

Get distribution_consumption if the pump or fan work at fix mass or volume flow in ratio over peak power (W/W)

Returns float

property distribution_consumption_variable_flow

Get distribution_consumption if the pump or fan work at variable mass or volume flow in ratio

over energy produced (J/J)

Returns float

property emission_systems → Optional[[*EmissionSystem*]]

Get energy emission systems connected to this distribution system

Returns [EmissionSystem]

property energy_storage_systems → Optional[[EnergyStorageSystem]]

Get energy storage systems connected to this distribution system

Returns [EnergyStorageSystem]

property generation_systems → Optional[[*GenerationSystem*]]

Get generation systems connected to the distribution system

Returns [GenerationSystem]

property heat_losses

Get heat_losses in ratio over energy produced in J/J

Returns float

property id

Get system id

Returns float

property model_name

Get model name

Returns string

property supply_temperature

Get supply_temperature in degree Celsius

Returns float

to_dictionary()

Class content to dictionary

property type

Get type from [air, water, refrigerant]

Returns string

4.6.5 Energy Systems Emission System Data Model

class EmissionSystem(*system_id*, *model_name=None*, *system_type=None*, *parasitic_energy_consumption=0*)

Inherit: :py:class:object

Emission system class

property id

Get system id

Returns float

property model_name

Get model name

Returns string

property parasitic_energy_consumption

Get parasitic_energy_consumption in ratio (J/J)

Returns float

to_dictionary()

Class content to dictionary

property type

Get type

Returns string

4.6.6 Energy Systems Generation System Data Model

class GenerationSystem(*system_id*, *name*, *model_name=None*, *manufacturer=None*, *fuel_type=None*, *distribution_systems=None*, *energy_storage_systems=None*)

Inherit: ABC

Heat Generation system class

property distribution_systems → Optional[[*DistributionSystem*]]

Get distributions systems connected to this generation system

Returns [DistributionSystem]

property energy_storage_systems → Optional[[EnergyStorageSystem]]

Get energy storage systems connected to this generation system

Returns [EnergyStorageSystem]

property fuel_type

Get fuel_type from [renewable, gas, diesel, electricity, wood, coal, biogas]

Returns string

property id

Get system id

Returns float

property manufacturer

Get name

Returns string

property model_name

Get system id

Returns float

property name

Get system name

Returns string

property system_type

Get type

Returns string

to_dictionary()

Class content to dictionary

4.6.7 Energy Systems Energy Systems Data Model

class System(*system_id*, *demand_types*, *name=None*, *generation_systems=None*, *distribution_systems=None*, *configuration_schema=None*)

Inherit: :py:class:object

System class

property configuration_schema → Path

Get system configuration schema

Returns Path

property demand_types

Get demand able to cover from [‘heating’, ‘cooling’, ‘domestic_hot_water’, ‘electricity’]

Returns [string]

property distribution_systems → Optional[[*DistributionSystem*]]

Get distribution systems

Returns [DistributionSystem]

property generation_systems → Optional[[*GenerationSystem*]]

Get generation systems

Returns [GenerationSystem]

property id

Get equipment id

Returns string

property name

Get the system name

Returns string

to_dictionary()

Class content to dictionary

4.7 Usage

4.7.1 Usage Catalog Factory

class UsageCatalogFactory(*handler*, *base_path=None*)

Inherit: :py:class:object

Usage catalog factory class

property _comnet

Retrieve Comnet catalog

property _eilat

Retrieve Eilat catalog

property _nrcan

Retrieve NRCAN catalog

property _palma

Retrieve Palma catalog

property catalog → *Catalog*

Enrich the city given to the class using the class given handler

Returns Catalog

4.7.2 Usage Content Data Model

class Content(*usages*)

Inherit: :py:class:object

Content class

to_dictionary()

Class content to dictionary

property usages → [*Usage*]

Get catalog usages

4.7.3 Usage Appliances Data Model

class Appliances(*density*, *convective_fraction*, *radiative_fraction*, *latent_fraction*, *schedules*)

Inherit: :py:class:object

Appliances class

property convective_fraction → Optional[float]

Get convective fraction

Returns None or float

property density → Optional[float]

Get appliances density in W/m2

Returns None or float

property latent_fraction → Optional[float]

Get latent fraction

Returns None or float

property radiative_fraction → Optional[float]

Get radiant fraction

Returns None or float

property schedules → Optional[[*Schedule*]]

Get schedules

dataType = fraction

Returns None or [Schedule]

to_dictionary()

Class content to dictionary

4.7.4 Usage Content Data Model

class Content(*usages*)

Inherit: :py:class:object

Content class

to_dictionary()

Class content to dictionary

property usages → [*Usage*]

Get catalog usages

4.7.5 Usage Domestic Hot Water Data Model

class DomesticHotWater(*density*, *peak_flow*, *service_temperature*, *schedules*)

Inherit: :py:class:object

DomesticHotWater class

property density → Optional[float]

Get domestic hot water load density in Watts per m2

Returns None or float

property peak_flow → Optional[float]

Get domestic hot water peak_flow density in m3 per second and m2

Returns None or float

property schedules → Optional[[*Schedule*]]

Get schedules

dataType = fraction of loads

Returns None or [Schedule]

property service_temperature → Optional[float]

Get service temperature in degrees Celsius

Returns None or float

to_dictionary()

Class content to dictionary

4.7.6 Usage Internal Gain Data Model

4.7.7 Usage Lighting Data Model

class Lighting(*density*, *convective_fraction*, *radiative_fraction*, *latent_fraction*, *schedules*)

Inherit: :py:class:object

Lighting class

property convective_fraction → Optional[float]

Get convective fraction

Returns None or float

property density → Optional[float]

Get lighting density in Watts per m2

Returns None or float

property latent_fraction → Optional[float]

Get latent fraction

Returns None or float

property radiative_fraction → Optional[float]

Get radiant fraction

Returns None or float

property schedules → Optional[[*Schedule*]]

Get schedules

dataType = fraction

Returns None or [Schedule]

to_dictionary()

Class content to dictionary

4.7.8 Usage Occupancy Data Model

class Occupancy(*occupancy_density*, *sensible_convective_internal_gain*, *sensible_radiative_internal_gain*, *latent_internal_gain*, *schedules*)

Inherit: :py:class:object

Occupancy class

property latent_internal_gain → Optional[float]

Get latent internal gain in Watts per m2

Returns None or float

property occupancy_density → Optional[float]

Get density in persons per m2

Returns None or float

property schedules → Optional[[*Schedule*]]

Get occupancy schedules

dataType = fraction

Returns None or [Schedule]

property sensible_convective_internal_gain → Optional[float]

Get sensible convective internal gain in Watts per m2

Returns None or float

property sensible_radiative_internal_gain → Optional[float]

Get sensible radiant internal gain in Watts per m2

Returns None or float

to_dictionary()

Class content to dictionary

4.7.9 Usage Schedule Data Model

class Schedule(*schedule_type*, *values*, *data_type*, *time_step*, *time_range*, *day_types*)

Inherit: :py:class:object

Schedule class

property data_type → Optional[str]

Get schedule data type from:

['any_number', 'fraction', 'on_off', 'temperature', 'humidity', 'control_type']

Returns None or str

property day_types → Optional[[str]]

Get schedule day types, as many as needed from:

['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'holiday', 'winter_design_day',

'summer_design_day']

Returns None or [str]

property time_range → Optional[str]

Get schedule time range from:

['minute', 'hour', 'day', 'week', 'month', 'year']

Returns None or str

property time_step → Optional[str]

Get schedule time step from:

['second', 'minute', 'hour', 'day', 'week', 'month']

Returns None or str

`to_dictionary()`

Class content to dictionary

`property type` → Optional[str]

Get schedule type

Returns None or str

`property values`

Get schedule values

Returns [Any]

4.7.10 Usage Thermal Control Data Model

`class ThermalControl`(*mean_heating_set_point*, *heating_set_back*, *mean_cooling_set_point*, *hvac_availability_schedules*, *heating_set_point_schedules*, *cooling_set_point_schedules*)

`Inherit: :py:class:object`

ThermalControl class

`property cooling_set_point_schedules` → Optional[[*Schedule*]]

Get cooling set point schedule defined for a thermal zone in Celsius

dataType = temperature

Returns None or [Schedule]

`property heating_set_back` → Optional[float]

Get heating set back defined for a thermal zone in Celsius

Returns None or float

`property heating_set_point_schedules` → Optional[[*Schedule*]]

Get heating set point schedule defined for a thermal zone in Celsius

dataType = temperature

Returns None or [Schedule]

`property hvac_availability_schedules` → Optional[[*Schedule*]]

Get the availability of the conditioning system defined for a thermal zone

dataType = on/off

Returns None or [Schedule]

`property mean_cooling_set_point` → Optional[float]

Get cooling set point defined for a thermal zone in Celsius

Returns None or float

`property mean_heating_set_point` → Optional[float]

Get heating set point defined for a thermal zone in Celsius

Returns None or float

`to_dictionary()`

Class content to dictionary

4.7.11 Usage Usage Data Model

class Usage(*name*, *hours_day*, *days_year*, *mechanical_air_change*, *ventilation_rate*, *occupancy*, *lighting*, *appliances*, *thermal_control*, *domestic_hot_water*)

Inherit: :py:class:object

Usage class

property appliances → Optional[*Appliances*]

Get appliances information

Returns None or Appliances

property days_year → Optional[float]

Get usage zone usage days per year

Returns None or float

property domestic_hot_water → Optional[*DomesticHotWater*]

Get domestic hot water information

Returns None or DomesticHotWater

property hours_day → Optional[float]

Get usage zone usage hours per day

Returns None or float

property lighting → Optional[*Lighting*]

Get lighting information

Returns None or Lighting

property mechanical_air_change → Optional[float]

Get usage zone mechanical air change in air change per second (1/s)

Returns None or float

property name → Optional[str]

Get usage zone usage name

Returns None or str

property occupancy → Optional[*Occupancy*]

Get occupancy in the usage zone

Returns None or Occupancy

property thermal_control → Optional[*ThermalControl*]

Get thermal control information

Returns None or ThermalControl

to_dictionary()

Class content to dictionary

property ventilation_rate → Optional[float]

Get usage zone ventilation rate in m3/m2s

Returns None or float

CHAPTER

FIVE

HELPERS

CERC hub provides a set of *helpers* that will simplify certain operations; these helpers are mean to be freely used at any point and therefore could be consumed from several places.

5.1 Folder structure

```
../hub/hub/helpers/
├── data
│   ├── alkis_function_to_hub_function.py
│   ├── eilat_function_to_hub_function.py
│   ├── hft_function_to_hub_function.py
│   ├── hub_function_to_eilat_construction_function.py
│   ├── hub_function_to_montreal_custom_costs_function.py
│   ├── hub_function_to_nrcan_construction_function.py
│   ├── hub_function_to_nrel_construction_function.py
│   ├── hub_function_to_palma_construction_function.py
│   ├── hub_usage_to_comnet_usage.py
│   ├── hub_usage_to_eilat_usage.py
│   ├── hub_usage_to_hft_usage.py
│   ├── hub_usage_to_nrcan_usage.py
│   ├── hub_usage_to_palma_usage.py
│   ├── __init__.py
│   ├── montreal_custom_fuel_to_hub_fuel.py
│   ├── montreal_demand_type_to_hub_energy_demand_type.py
│   ├── montreal_function_to_hub_function.py
│   ├── montreal_generation_system_to_hub_energy_generation_system.py
│   ├── montreal_system_to_hub_energy_generation_system.py
│   ├── north_america_custom_fuel_to_hub_fuel.py
│   ├── north_america_demand_type_to_hub_energy_demand_type.py
│   ├── north_america_storage_system_to_hub_storage.py
│   ├── north_america_system_to_hub_energy_generation_system.py
│   ├── palma_function_to_hub_function.py
│   └── pluto_function_to_hub_function.py
├── parsers
│   ├── __init__.py
│   ├── list_usage_to_hub.py
│   └── string_usage_to_hub.py
├── peak_calculation
│   ├── __init__.py
│   └── loads_calculation.py
├── auth.py
├── configuration_helper.py
├── constants.py
├── dictionaries.py
├── geometry_helper.py
├── __init__.py
├── location.py
├── monthly_values.py
├── peak_loads.py
├── thermal_zones_creation.py
├── usage_parsers.py
└── utils.py

4 directories, 42 files
```

5.2 Configuration Helper

class ConfigurationHelper

Inherit: :py:class:object

Configuration class

property cold_water_temperature → float

Get configured cold water temperature in Celsius

Returns 10

property comnet_lighting_convective → float

Get configured convective ratio of internal gains do to lighting used for Comnet (ASHRAE) standard

Returns 0.5

property comnet_lighting_latent → float

Get configured latent ratio of internal gains do to lighting used for Comnet (ASHRAE) standard

Returns 0

property comnet_lighting_radiant → float

Get configured radiant ratio of internal gains do to lighting used for Comnet (ASHRAE) standard

Returns 0.5

property comnet_occupancy_sensible_convective → float

Get configured convective ratio of the sensible part of internal gains do to occupancy

used for Comnet (ASHRAE) standard

Returns 0.9

property comnet_occupancy_sensible_radiant → float

Get configured radiant ratio of the sensible part of internal gains do to occupancy

used for Comnet (ASHRAE) standard

Returns 0.1

property comnet_plugs_convective → float

Get configured convective ratio of internal gains do to electrical appliances used for Comnet (ASHRAE) standard

Returns 0.75

property comnet_plugs_latent → float

Get configured latent ratio of internal gains do to electrical appliances used for Comnet (ASHRAE) standard

Returns 0

property comnet_plugs_radiant → float

Get configured radiant ratio of internal gains do to electrical appliances used for Comnet (ASHRAE) standard

Returns 0.25

property convective_heat_transfer_coefficient_exterior → float

Get configured convective heat transfer coefficient for surfaces outside the building

Returns 20 W/m2K

property convective_heat_transfer_coefficient_interior → float

Get configured convective heat transfer coefficient for surfaces inside the building

Returns 3.5 W/m2K

property max_coordinate → float

Get configured maximal coordinate value

Returns 1.7976931348623157e+308

property min_coordinate → float

Get configured minimal coordinate value

Returns -1.7976931348623157e+308

property short_wave_reflectance → float

Get configured short wave reflectance for surfaces that don't have construction assigned

Returns 0.3

property soil_conductivity → float

Get configured soil conductivity for surfaces touching the ground

Returns 3 W/mK

property soil_thickness → float

Get configured soil thickness for surfaces touching the ground

Returns 0.5 m

5.3 Constants

KELVIN = 273.15
HOUR_TO_MINUTES = 60
MINUTES_TO_SECONDS = 60
HOUR_TO_SECONDS = 3600
METERS_TO_FEET = 3.28084
BTU_H_TO_WATTS = 0.29307107
KILO_WATTS_HOUR_TO_JULES = 3600000
WATTS_HOUR_TO_JULES = 3600
GALLONS_TO_QUBIC_METERS = 0.0037854117954011185
INFILTRATION_75PA_TO_4PA = (4 / 75) ** 0.65
SECOND = 'second'
MINUTE = 'minute'
HOUR = 'hour'
DAY = 'day'
WEEK = 'week'
MONTH = 'month'
YEAR = 'year'
MONDAY = 'monday'
TUESDAY = 'tuesday'
WEDNESDAY = 'wednesday'
THURSDAY = 'thursday'
FRIDAY = 'friday'
SATURDAY = 'saturday'
SUNDAY = 'sunday'
HOLIDAY = 'holiday'
WINTER_DESIGN_DAY = 'winter_design_day'
SUMMER_DESIGN_DAY = 'summer_design_day'
WEEK_DAYS = 'Weekdays'
WEEK_ENDS = 'Weekends'
ALL_DAYS = 'Alldays'
JANUARY = 'January'
FEBRUARY = 'February'
MARCH = 'March'
APRIL = 'April'
MAY = 'May'
JUNE = 'June'
JULY = 'July'
AUGUST = 'August'
SEPTEMBER = 'September'
OCTOBER = 'October'
NOVEMBER = 'November'
DECEMBER = 'December'
MONTHS = [JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER]
WEEK_DAYS_A_MONTH = {JANUARY: {MONDAY: 5,

```
        TUESDAY: 5,
        WEDNESDAY: 5,
        THURSDAY: 4,
        FRIDAY: 4,
        SATURDAY: 4,
        SUNDAY: 4,
        HOLIDAY: 0},
    FEBRUARY: {MONDAY: 4,
        TUESDAY: 4,
        WEDNESDAY: 4,
        THURSDAY: 4,
        FRIDAY: 4,
        SATURDAY: 4,
        SUNDAY: 4,
        HOLIDAY: 0},
    MARCH: {MONDAY: 4,
        TUESDAY: 4,
        WEDNESDAY: 4,
        THURSDAY: 5,
        FRIDAY: 5,
        SATURDAY: 5,
        SUNDAY: 4,
        HOLIDAY: 0},
    APRIL: {MONDAY: 5,
        TUESDAY: 4,
        WEDNESDAY: 4,
        THURSDAY: 4,
        FRIDAY: 4,
        SATURDAY: 4,
        SUNDAY: 5,
        HOLIDAY: 0},
    MAY: {MONDAY: 4,
        TUESDAY: 5,
        WEDNESDAY: 5,
        THURSDAY: 5,
        FRIDAY: 4,
        SATURDAY: 4,
        SUNDAY: 4,
        HOLIDAY: 0},
    JUNE: {MONDAY: 4,
        TUESDAY: 4,
        WEDNESDAY: 4,
        THURSDAY: 4,
        FRIDAY: 5,
        SATURDAY: 5,
        SUNDAY: 4,
        HOLIDAY: 0},
```

JULY: {MONDAY: 5,
TUESDAY: 5,
WEDNESDAY: 4,
THURSDAY: 4,
FRIDAY: 4,
SATURDAY: 4,
SUNDAY: 5,
HOLIDAY: 0},
AUGUST: {MONDAY: 4,
TUESDAY: 4,
WEDNESDAY: 5,
THURSDAY: 5,
FRIDAY: 5,
SATURDAY: 4,
SUNDAY: 4,
HOLIDAY: 0},
SEPTEMBER: {MONDAY: 4,
TUESDAY: 4,
WEDNESDAY: 4,
THURSDAY: 4,
FRIDAY: 4,
SATURDAY: 5,
SUNDAY: 5,
HOLIDAY: 0},
OCTOBER: {MONDAY: 5,
TUESDAY: 5,
WEDNESDAY: 5,
THURSDAY: 4,
FRIDAY: 4,
SATURDAY: 4,
SUNDAY: 4,
HOLIDAY: 0},
NOVEMBER: {MONDAY: 4,
TUESDAY: 4,
WEDNESDAY: 4,
THURSDAY: 5,
FRIDAY: 5,
SATURDAY: 4,
SUNDAY: 4,
HOLIDAY: 0},
DECEMBER: {MONDAY: 5,
TUESDAY: 4,
WEDNESDAY: 4,
THURSDAY: 4,
FRIDAY: 4,
SATURDAY: 5,
SUNDAY: 5,

```
            HOLIDAY: 0},
        }
    WEEK_DAYS_A_YEAR = {MONDAY: 51,
        TUESDAY: 50,
        WEDNESDAY: 50,
        THURSDAY: 50,
        FRIDAY: 50,
        SATURDAY: 52,
        SUNDAY: 52,
        HOLIDAY: 10}
    DAYS_A_MONTH = {JANUARY: 31,
        FEBRUARY: 28,
        MARCH: 31,
        APRIL: 30,
        MAY: 31,
        JUNE: 30,
        JULY: 31,
        AUGUST: 31,
        SEPTEMBER: 30,
        OCTOBER: 31,
        NOVEMBER: 30,
        DECEMBER: 31}
    HOURS_A_MONTH = {JANUARY: 744,
        FEBRUARY: 672,
        MARCH: 744,
        APRIL: 720,
        MAY: 744,
        JUNE: 720,
        JULY: 744,
        AUGUST: 744,
        SEPTEMBER: 720,
        OCTOBER: 744,
        NOVEMBER: 720,
        DECEMBER: 744}
    ANY_NUMBER = 'any_number'
    FRACTION = 'fraction'
    ON_OFF = 'on_off'
    TEMPERATURE = 'temperature'
    HUMIDITY = 'humidity'
    CONTROL_TYPE = 'control_type'
    CONTINUOUS = 'continuous'
    DISCRETE = 'discrete'
    CONSTANT = 'constant'
    INTERNAL_GAINS = 'internal_gains'
    WALL = 'Wall'
    GROUND_WALL = 'Ground wall'
    GROUND = 'Ground'
```

ATTIC_FLOOR = 'Attic floor'
ROOF = 'Roof'
INTERIOR_SLAB = 'Interior slab'
INTERIOR_WALL = 'Interior wall'
VIRTUAL_INTERNAL = 'Virtual internal'
WINDOW = 'Window'
DOOR = 'Door'
SKYLIGHT = 'Skylight'
RESIDENTIAL = 'residential'
SINGLE_FAMILY_HOUSE = 'single family house'
MULTI_FAMILY_HOUSE = 'multifamily house'
ROW_HOUSE = 'row house'
MID_RISE_APARTMENT = 'mid rise apartment'
HIGH_RISE_APARTMENT = 'high rise apartment'
OFFICE_AND_ADMINISTRATION = 'office and administration'
SMALL_OFFICE = 'small office'
MEDIUM_OFFICE = 'medium office'
LARGE_OFFICE = 'large office'
COURTHOUSE = 'courthouse'
FIRE_STATION = 'fire station'
PENITENTIARY = 'penitentiary'
POLICE_STATION = 'police station'
POST_OFFICE = 'post office'
LIBRARY = 'library'
EDUCATION = 'education'
PRIMARY_SCHOOL = 'primary school'
PRIMARY_SCHOOL_WITH_SHOWER = 'school with shower'
SECONDARY_SCHOOL = 'secondary school'
UNIVERSITY = 'university'
LABORATORY_AND_RESEARCH_CENTER = 'laboratory and research centers'
STAND_ALONE_RETAIL = 'stand alone retail'
HOSPITAL = 'hospital'
OUT_PATIENT_HEALTH_CARE = 'out-patient health care'
HEALTH_CARE = 'health care'
RETIREMENT_HOME_OR_ORPHANAGE = 'retirement home or orphanage'
COMMERCIAL = 'commercial'
STRIP_MALL = 'strip mall'
SUPERMARKET = 'supermarket'
RETAIL_SHOP_WITHOUT_REFRIGERATED_FOOD = 'retail shop without refrigerated food'
RETAIL_SHOP_WITH_REFRIGERATED_FOOD = 'retail shop with refrigerated food'
RESTAURANT = 'restaurant'
QUICK_SERVICE_RESTAURANT = 'quick service restaurant'
FULL_SERVICE_RESTAURANT = 'full service restaurant'
HOTEL = 'hotel'
HOTEL_MEDIUM_CLASS = 'hotel medium class'
SMALL_HOTEL = 'small hotel'
LARGE_HOTEL = 'large hotel'

DORMITORY = 'dormitory'
EVENT_LOCATION = 'event location'
CONVENTION_CENTER = 'convention center'
HALL = 'hall'
GREEN_HOUSE = 'green house'
INDUSTRY = 'industry'
WORKSHOP = 'workshop'
WAREHOUSE = 'warehouse'
WAREHOUSE_REFRIGERATED = 'warehouse refrigerated'
SPORTS_LOCATION = 'sports location'
SPORTS_ARENA = 'sports arena'
GYMNASIUM = 'gymnasium'
MOTION_PICTURE_THEATRE = 'motion picture theatre'
MUSEUM = 'museum'
PERFORMING_ARTS_THEATRE = 'performing arts theatre'
TRANSPORTATION = 'transportation'
AUTOMOTIVE_FACILITY = 'automotive facility'
PARKING_GARAGE = 'parking garage'
RELIGIOUS = 'religious'
NON_HEATED = 'non-heated'
DATACENTER = 'datacenter'
FARM = 'farm'
LIGHTING = 'Lights'
OCCUPANCY = 'Occupancy'
APPLIANCES = 'Appliances'
HVAC_AVAILABILITY = 'HVAC Avail'
INFILTRATION = 'Infiltration'
VENTILATION = 'Ventilation'
COOLING_SET_POINT = 'ClgSetPt'
HEATING_SET_POINT = 'HtgSetPt'
EQUIPMENT = 'Equipment'
ACTIVITY = 'Activity'
PEOPLE_ACTIVITY_LEVEL = 'People Activity Level'
DOMESTIC_HOT_WATER = 'Domestic Hot Water'
HEATING = 'Heating'
COOLING = 'Cooling'
ELECTRICITY = 'Electricity'
RENEWABLE = 'Renewable'
WOOD = 'Wood'
GAS = 'Gas'
DIESEL = 'Diesel'
COAL = 'Coal'
BIOMASS = 'Biomass'
BUTANE = 'Butane'
AIR = 'Air'
WATER = 'Water'
GEOTHERMAL = 'Geothermal'

DISTRICT_HEATING_NETWORK = 'District Heating'
GRID = 'Grid'
ONSITE_ELECTRICITY = 'Onsite Electricity'
PHOTOVOLTAIC = 'Photovoltaic'
BOILER = 'Boiler'
FURNACE = 'Furnace'
HEAT_PUMP = 'Heat Pump'
BASEBOARD = 'Baseboard'
ELECTRICITY_GENERATOR = 'Electricity generator'
CHILLER = 'Chiller'
SPLIT = 'Split'
JOULE = 'Joule'
BUTANE_HEATER = 'Butane Heater'
SENSIBLE = 'sensible'
LATENT = 'Latent'
LITHIUMION = 'Lithium Ion'
NICD = 'NiCd'
LEADACID = 'Lead Acid'
EPSILON = 0.0000001
ONLY_HEATING = 'Heating'
ONLY_COOLING = 'Colling'
ONLY_VENTILATION = 'Ventilation'
HEATING_AND_VENTILATION = 'Heating and ventilation'
COOLING_AND_VENTILATION = 'Cooling and ventilation'
HEATING_AND_COLLING = 'Heating and cooling'
FULL_HVAC = 'Heating and cooling and ventilation'
MAX_FLOAT = float('inf')
MIN_FLOAT = float('-inf')
SRA = 'sra'
INSEL_MEB = 'insel meb'
CURRENCY_PER_SQM = 'currency/m2'
CURRENCY_PER_CBM = 'currency/m3'
CURRENCY_PER_KW = 'currency/kW'
CURRENCY_PER_KWH = 'currency/kWh'
CURRENCY_PER_MONTH = 'currency/month'
CURRENCY_PER_LITRE = 'currency/l'
CURRENCY_PER_KG = 'currency/kg'
CURRENCY_PER_CBM_PER_HOUR = 'currency/(m3/h)'
PERCENTAGE = '%'
SUPERSTRUCTURE = 'B_shell'
ENVELOPE = 'D_services'
ALLOWANCES_OVERHEAD_PROFIT = 'Z_allowances_overhead_profit'

5.4 Geometry Helper

class GeometryHelper(*delta=0*, *area_delta=0*)

Inherit: :py:class:object

Geometry helper class

static city_mapping(*city*, *building_names=None*, *plot=False*) → Dict

Parameter city city to be mapped

Parameter building_names list of building names to be mapped or None

Parameter plot True if minimap image should be displayed

Returns shared_information dictionary

static coordinate_to_map_point(*coordinate*, *city*)

Transform a real world coordinate to a minimap one

Parameter coordinate real world coordinate

Parameter city current city

Returns None

static distance_between_points(*vertex1*, *vertex2*)

distance between points in an n-D Euclidean space

Parameter vertex1 point or vertex

Parameter vertex2 point or vertex

Returns float

static divide_mesh_by_plane(*trimesh*, *normal_plane*, *point_plane*)

Divide a mesh by a plane

Parameter trimesh Trimesh

Parameter normal_plane [x, y, z]

Parameter point_plane [x, y, z]

Returns [Trimesh]

static factor()

Set minimap resolution

Returns None

static get_location(*latitude*, *longitude*) → *Location*

Get Location from latitude and longitude

Parameter latitude Latitude

Parameter longitude Longitude

Returns Location

static segment_list_to_trimesh(*lines*) → Trimesh

Parameter lines lines

Returns Transform a list of segments into a Trimesh

5.5 Location

class Location(*country*, *city*, *region_code*, *climate_reference_city_latitude*, *climate_reference_city_longitude*)
Inherit: :py:class:object

property city
Get city name

property climate_reference_city_latitude
Get climate-reference-city latitude

property climate_reference_city_longitude
Get climate-reference-city longitude

property country
Get country code

property region_code
Get region

5.6 Dictionaries

class Dictionaries
Inherit: :py:class:object
Dictionaries class

property alkis_function_to_hub_function → dict
Get Alkis function to hub function, transformation dictionary

property eilat_function_to_hub_function → dict
Get Eilat's function to hub function, transformation dictionary

property hft_function_to_hub_function → dict
Get Hft function to hub function, transformation dictionary

Returns dict

property hub_function_to_eilat_construction_function → dict
Get hub function to NRCAN construction function, transformation dictionary

Returns dict

property hub_function_to_montreal_custom_costs_function → dict
Get hub function to Montreal custom costs function, transformation dictionary

Returns dict

property hub_function_to_nrcan_construction_function → dict
Get hub function to NRCAN construction function, transformation dictionary

Returns dict

property hub_function_to_nrel_construction_function → dict
Get hub function to NREL construction function, transformation dictionary

Returns dict

property hub_function_to_palma_construction_function → dict

Get hub function to Palma construction function, transformation dictionary

Returns dict

property hub_usage_to_comnet_usage → dict

Hub usage to Comnet usage, transformation dictionary

Returns dict

property hub_usage_to_eilat_usage → dict

Hub usage to Eilat usage, transformation dictionary

Returns dict

property hub_usage_to_hft_usage → dict

Hub usage to HfT usage, transformation dictionary

Returns dict

property hub_usage_to_nrcan_usage → dict

Get hub usage to NRCAN usage, transformation dictionary

Returns dict

property hub_usage_to_palma_usage → dict

Hub usage to Palma usage, transformation dictionary

Returns dict

property montreal_custom_fuel_to_hub_fuel → dict

Get hub fuel from montreal_custom catalog fuel

property montreal_demand_type_to_hub_energy_demand_type

Get montreal custom system demand type to hub energy demand type, transformation dictionary

property montreal_function_to_hub_function → dict

Get Montreal function to hub function, transformation dictionary

property montreal_generation_system_to_hub_energy_generation_system

Get montreal custom generation system names to hub energy system names, transformation dictionary

property montreal_system_to_hub_energy_generation_system

Get montreal custom system names to hub energy system names, transformation dictionary

property north_america_custom_fuel_to_hub_fuel → dict

Get hub fuel from north_america catalog fuel

property north_america_demand_type_to_hub_energy_demand_type

Get north america system demand type to hub energy demand type, transformation dictionary

property north_america_storage_system_to_hub_storage

Get montreal custom system names to hub storage system

property north_america_system_to_hub_energy_generation_system

Get north america system names to hub energy system names, transformation dictionary

property palma_function_to_hub_function → dict

Get Palma function to hub function, transformation dictionary

Returns dict

property pluto_function_to_hub_function → dict

Get Pluto function to hub function, transformation dictionary

Returns dict

CHAPTER

SIX

ADDITIONAL FILES

6.1 Readme

README.md

6.2 License

LICENSE.md

6.3 Code of conduct

CODE_OF_CONDUCT.md

6.4 How to contribute

CONTRIBUTING.md

6.5 Coding style

PYGUIDE.md

INDEX

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Y