

---

# **CERC hub reference manual**

*Release 0.1.8.26*

## **CERC Next-Generation Cities**

**Nov 06, 2023**

# CONTENTS:

<b>1</b>	<b>CERC HUB' reference manual</b>	<b>1</b>
1.1	Authors . . . . .	1
1.2	Contributors . . . . .	1
1.3	About the HUB . . . . .	1
<b>2</b>	<b>City model structure</b>	<b>2</b>
2.1	City model structure UML . . . . .	3
2.2	Folder structure . . . . .	4
2.2.1	city_model_structure . . . . .	4
2.2.2	attributes . . . . .	4
2.2.3	building_demand . . . . .	5
2.2.4	energy_systems . . . . .	5
2.2.5	iot . . . . .	5
2.2.6	full schema . . . . .	6
2.3	Classes . . . . .	7
2.3.1	City . . . . .	7
2.3.2	CityObject . . . . .	8
2.3.3	City Objects Cluster . . . . .	8
2.3.4	Building . . . . .	9
2.3.5	Parts Consisting Building . . . . .	9
2.3.6	Buildings Cluster . . . . .	9
2.3.7	Network . . . . .	9
2.3.8	Level of detail . . . . .	9
2.3.9	Edge . . . . .	9
2.3.10	Node . . . . .	9
2.3.11	Plane . . . . .	10
2.3.12	Point . . . . .	10
2.3.13	Polygon . . . . .	10
2.3.14	Polyhedron . . . . .	10
2.3.15	Record . . . . .	11
2.3.16	Schedule . . . . .	11
2.3.17	Time Series . . . . .	11
2.3.18	Appliances . . . . .	11
2.3.19	Household . . . . .	11
2.3.20	Internal Gain . . . . .	11
2.3.21	Internal Zone . . . . .	11
2.3.22	Layer . . . . .	12
2.3.23	Lighting . . . . .	12
2.3.24	Occupancy . . . . .	12
2.3.25	Storey . . . . .	12

2.3.26	Surface	12
2.3.27	Thermal Boundary	12
2.3.28	Thermal Control	12
2.3.29	Thermal Opening	13
2.3.30	Thermal Zone	13
2.3.31	Usage	13
2.3.32	Plant	13
2.3.33	Soil	13
2.3.34	Vegetation	13
2.3.35	Sensor	13
2.3.36	Sensor Measure	13
2.3.37	Sensor Type	14
2.3.38	Station	14
<b>3</b>	<b>Factories</b>	<b>15</b>
3.1	Folder structure	16
3.1.1	Imports	16
3.1.2	Exports	16
3.2	Imports Classes	17
3.2.1	Construction Factory	17
3.2.2	Geometry Factory	17
3.2.3	Usage Factory	17
3.2.4	Weather Factory	18
3.3	Exports	19
3.3.1	Export Factory	19
<b>4</b>	<b>Catalogs</b>	<b>20</b>
4.1	Folder structure	21
4.1.1	Catalogs	21
4.2	Catalog Base Class	22
4.2.1	Catalog	22
4.3	Greenery	23
4.3.1	Greenery Catalog Factory	23
4.3.2	Greenery Content Data Model	23
4.3.3	Greenery Plant Data Model	23
4.3.4	Greenery Plant Percentage Data Model	23
4.3.5	Greenery Plant Soil Data Model	23
4.3.6	Greenery Vegetation Data Model	23
4.4	Construction	24
4.4.1	Construction Catalog Factory	24
4.4.2	Construction Content Data Model	24
4.4.3	Construction Archetype Data Model	24
4.4.4	Construction Construction Data Model	24
4.4.5	Construction Layer Data Model	24
4.4.6	Construction Material Data Model	25
4.4.7	Construction Window Data Model	25
4.5	Costs	26
4.5.1	Costs Catalog Factory	26
4.5.2	Costs Content Data Model	26
4.5.3	Costs Archetype Data Model	26
4.5.4	Costs Capital Cost Data Model	26
4.5.5	Costs Chapter Data Model	26
4.5.6	Costs Fuel Data Model	27
4.5.7	Costs Income Data Model	27

4.5.8	Costs Item Description Data Model . . . . .	27
4.5.9	Costs Operational Cost Data Model . . . . .	27
4.6	Energy Systems . . . . .	28
4.6.1	Energy Systems Catalog Factory . . . . .	28
4.6.2	Energy Systems Content Data Model . . . . .	28
4.6.3	Energy Systems Archetype Data Model . . . . .	28
4.6.4	Energy Systems Distribution System Data Model . . . . .	28
4.6.5	Energy Systems Emission System Data Model . . . . .	28
4.6.6	Energy Systems Generation System Data Model . . . . .	28
4.6.7	Energy Systems Energy Systems Data Model . . . . .	29
4.7	Usage . . . . .	30
4.7.1	Usage Catalog Factory . . . . .	30
4.7.2	Usage Content Data Model . . . . .	30
4.7.3	Usage Appliances Data Model . . . . .	30
4.7.4	Usage Content Data Model . . . . .	30
4.7.5	Usage Domestic Hot Water Data Model . . . . .	30
4.7.6	Usage Internal Gain Data Model . . . . .	30
4.7.7	Usage Lighting Data Model . . . . .	30
4.7.8	Usage Occupancy Data Model . . . . .	31
4.7.9	Usage Schedule Data Model . . . . .	31
4.7.10	Usage Thermal Control Data Model . . . . .	31
4.7.11	Usage Usage Data Model . . . . .	31
<b>5</b>	<b>Helpers</b>	<b>32</b>
5.1	Folder structure . . . . .	32
5.2	Configuration Helper . . . . .	33
5.3	Constants . . . . .	34
5.4	Geometry Helper . . . . .	40
5.5	Location . . . . .	41
5.6	Dictionaries . . . . .	41
<b>6</b>	<b>Additional Files</b>	<b>42</b>
6.1	Readme . . . . .	42
6.2	License . . . . .	42
6.3	Code of conduct . . . . .	42
6.4	How to contribute . . . . .	42
6.5	Coding style . . . . .	42
	<b>Index</b>	<b>43</b>

## 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.

## 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.

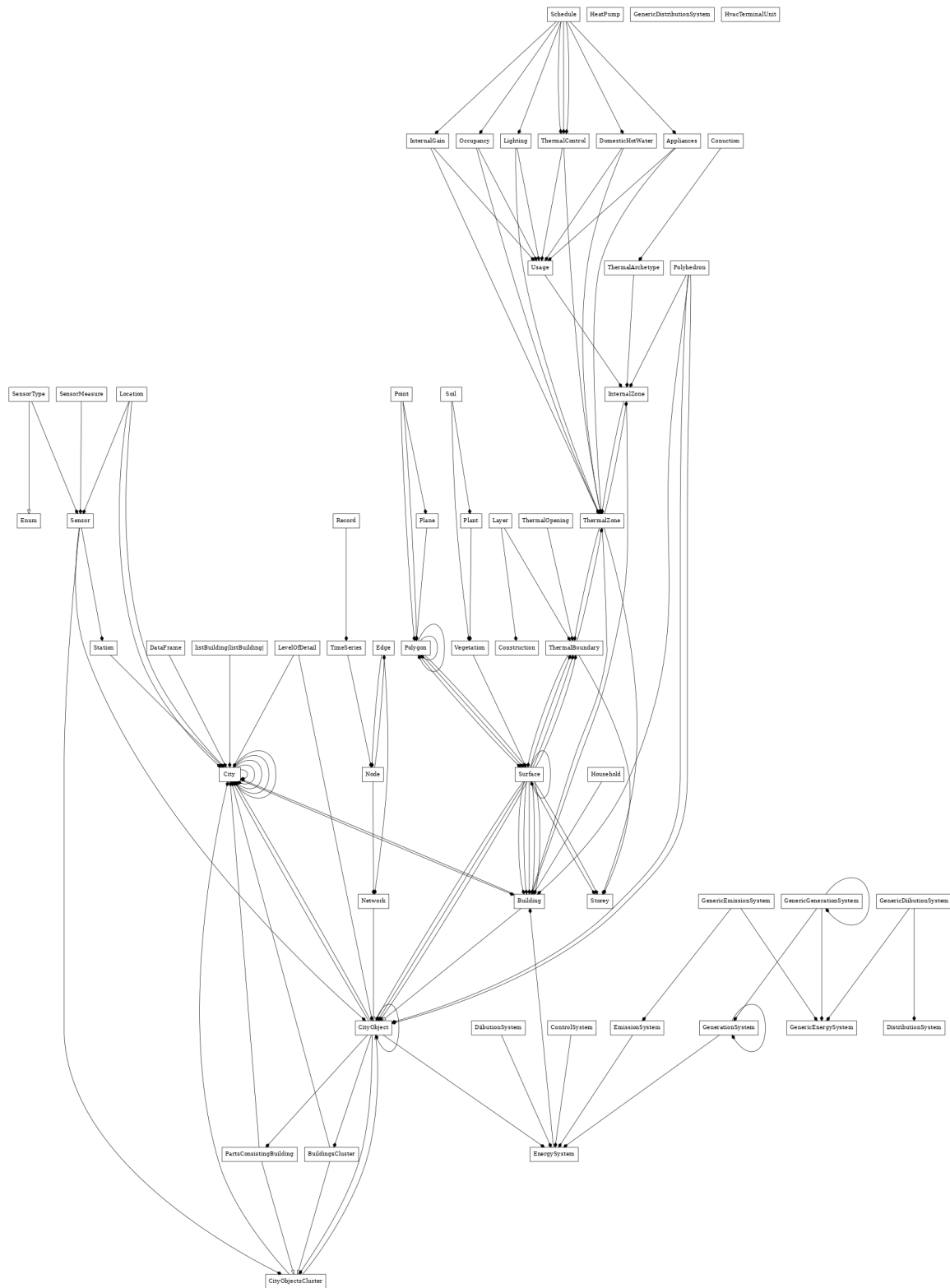
---

**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
```

1 directory, 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
```

0 directories, 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
```

0 directories, 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
├── emission_system.py
├── energy_system.py
├── generation_system.py
├── generic_distribution_system.py
├── generic_emission_system.py
├── generic_energy_system.py
├── generic_generation_system.py
├── heat_pump.py
├── hvac_terminal_unit.py
└── __init__.py
```

0 directories, 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
```

0 directories, 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
│   ├── emission_system.py
│   ├── energy_system.py
│   ├── generation_system.py
│   ├── generic_distribution_system.py
│   ├── generic_emission_system.py
│   ├── generic_energy_system.py
│   ├── generic_generation_system.py
│   ├── heat_pump.py
│   ├── hvac_terminal_unit.py
│   └── __init__.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

```

5 directories, 58 files

## 2.3 Classes

### 2.3.1 City

**class** `City` (*lower\_corner*, *upper\_corner*, *srs\_name*)

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*) → list[Building | list[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]

**city\_object** (*name*) → Optional[CityObject]

Retrieve the city CityObject with the given name

**Parameter** *name* str

**Returns** None or CityObject

**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

**merge** (*city*) → City

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

**Returns** City

**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

**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

### 2.3.2 CityObject

**class CityObject** (*name, surfaces*)

class CityObject

**surface** (*name*) → Optional[*Surface*]

Get the city object surface with a given name

**Parameter** name str

**Returns** None or Surface

**surface\_by\_id** (*identification\_number*) → Optional[*Surface*]

Get the city object surface with a given name

**Parameter** identification\_number str

**Returns** None or Surface

### 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]

### 2.3.4 Building

```
class Building (name, surfaces, year_of_construction, function, terrains=None, city=None)
Inherit: CityObject
    Building(CityObject) class

    add_alias (value)
        Add a new alias for the building
```

### 2.3.5 Parts Consisting Building

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

### 2.3.6 Buildings Cluster

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

### 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
```

### 2.3.8 Level of detail

```
class LevelOfDetail
    Level of detail for the city class
```

### 2.3.9 Edge

```
class Edge (name, nodes=None)
    Generic edge class to be used as base for the network edges
```

### 2.3.10 Node

```
class Node (name, edges=None)
    Generic node class to be used as base for the network nodes
```

### 2.3.11 Plane

**class Plane** (*origin, normal*)

Plane class

**distance\_to\_point** (*point*)

Distance between the given point and the plane

**Returns** float

### 2.3.12 Point

**class Point** (*coordinates*)

Point class

**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*)

Polygon class

**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]

**static triangle\_mesh** (*vertices, normal*) → Trimesh

Get the triangulated mesh for the polygon

**Returns** Trimesh

### 2.3.14 Polyhedron

**class Polyhedron** (*polygons*)

Polyhedron class

**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

### 2.3.15 Record

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

### 2.3.16 Schedule

**class Schedule**  
Schedule class

### 2.3.17 Time Series

**class TimeSeries** (*time\_series\_type=None, records=None*)  
TimeSeries class

### 2.3.18 Appliances

**class Appliances**  
Appliances class

### 2.3.19 Household

**class Household**  
Household class

### 2.3.20 Internal Gain

**class InternalGain**  
InternalGain class

### 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*)  
InternalZone class

### 2.3.22 Layer

**class Layer**  
Layer class

### 2.3.23 Lighting

**class Lighting**  
Lighting class

### 2.3.24 Occupancy

**class Occupancy**  
Occupancy class

### 2.3.25 Storey

**class Storey** (*name, storey\_surfaces, neighbours, volume, internal\_zone, floor\_area*)  
Storey class

### 2.3.26 Surface

**class Surface** (*solid\_polygon, perimeter\_polygon, holes\_polygons=None, name=None, sur-  
face\_type=None*)  
Surface class

**divide** (*z*)  
Divides a surface at Z plane  
**Returns** Surface, Surface, Any

### 2.3.27 Thermal Boundary

**class ThermalBoundary** (*parent\_surface, opaque\_area, windows\_areas*)  
ThermalBoundary class

### 2.3.28 Thermal Control

**class ThermalControl**  
ThermalControl class



### 2.3.29 Thermal Opening

**class ThermalOpening**  
ThermalOpening class

### 2.3.30 Thermal Zone

**class ThermalZone** (*thermal\_boundaries, parent\_internal\_zone, volume, footprint\_area, number\_of\_storeys, usage\_name=None*)  
ThermalZone class

### 2.3.31 Usage

**class Usage**  
Usage class

### 2.3.32 Plant

**class Plant** (*name, height, leaf\_area\_index, leaf\_reflectivity, leaf\_emissivity, minimal\_stomatal\_resistance, co2\_sequestration, grows\_on\_soils*)  
Plant class

### 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*)  
Soil class

### 2.3.34 Vegetation

**class Vegetation** (*name, soil, soil\_thickness, plants*)  
Vegetation class

### 2.3.35 Sensor

**class Sensor**  
Sensor abstract class

### 2.3.36 Sensor Measure

**class SensorMeasure** (*latitude, longitude, utc\_timestamp, value*)  
Sensor measure class

### 2.3.37 Sensor Type

**class** `SensorType` (*value*)

**Inherit:** `Enum`

Sensor type enumeration


### 2.3.38 Station

**class** `Station` (*station\_id=None, \_mobile=False*)

Station class

## 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
├── results
│   ├── __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

```

2 directories, 12 files

### 3.1.2 Exports

```

../hub/hub/exports/
├── building_energy
│   ├── idf_files
│   │   ├── Energy+.idd
│   │   └── Minimal.idf
│   └── insel
│       ├── __init__.py
│       └── insel_monthly_energy_balance.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

```

4 directories, 14 files

## 3.2 Imports Classes

### 3.2.1 Construction Factory

**class ConstructionFactory** (*handler, city*)

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

**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, data\_frame=None, aliases\_field=None, height\_field=None, year\_of\_construction\_field=None, function\_field=None, function\_to\_hub=None, hub\_crs=None*)

GeometryFactory class

### 3.2.3 Usage Factory

**class UsageFactory** (*handler, city*)

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

**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: hub.city\_model\_structure.city.City*)

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*)  
Exports factory class

**export** ()  
Export the city given to the class using the given export type handler

**Returns** None

## 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
├── 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
│   │   ├── emission_system.py
│   │   ├── generation_system.py
│   │   ├── __init__.py
│   │   └── 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
├── 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
│   └── 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

```

12 directories, 65 files

## 4.2 Catalog Base Class

### 4.2.1 Catalog

**class Catalog**

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)  
    GreeneryCatalogFactory class
```

### 4.3.2 Greenery Content Data Model

```
class Content (vegetations, plants, soils)  
    Content class  
  
    to_dictionary ()  
        Class content to dictionary
```

### 4.3.3 Greenery Plant Data Model

```
class Plant (category, plant)  
    Plant class  
  
    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  
  
    to_dictionary ()  
        Class content to dictionary
```

### 4.3.5 Greenery Plant Soil Data Model

```
class Soil (soil)  
    Soil class  
  
    to_dictionary ()  
        Class content to dictionary
```

### 4.3.6 Greenery Vegetation Data Model

```
class Vegetation (category, vegetation, plant_percentages)  
    Vegetation class  
  
    to_dictionary ()  
        Class content to dictionary
```

## 4.4 Construction

### 4.4.1 Construction Catalog Factory

```
class ConstructionCatalogFactory (handler, base_path=None)
    Construction catalog factory class
```

### 4.4.2 Construction Content Data Model

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

    to_dictionary ()
        Class content to dictionary
```

### 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, infiltra-
                 tion_rate_for_ventilation_system_on)
    Archetype class

    to_dictionary ()
        Class content to dictionary
```

### 4.4.4 Construction Construction Data Model

```
class Construction (construction_id, construction_type, name, layers, window_ratio=None, win-
                    dow=None)
    Construction class

    to_dictionary ()
        Class content to dictionary
```

### 4.4.5 Construction Layer Data Model

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

    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, spe-  
cific_heat=None)
```

Material class

```
to_dictionary ()
```

Class content to dictionary

#### 4.4.7 Construction Window Data Model

```
class Window (window_id, frame_ratio, g_value, overall_u_value, name, window_type=None)
```

Window class

```
to_dictionary ()
```

Class content to dictionary

## 4.5 Costs

### 4.5.1 Costs Catalog Factory

**class CostsCatalogFactory** (*file\_type, base\_path=None*)  
CostsCatalogFactory class

### 4.5.2 Costs Content Data Model

**class Content** (*archetypes*)  
Content class

**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*)  
Archetype class

**to\_dictionary** ()  
Class content to dictionary

### 4.5.4 Costs Capital Cost Data Model

**class CapitalCost** (*general\_chapters, design\_allowance, overhead\_and\_profit*)  
Capital cost class

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

**Returns** Chapter

**to\_dictionary** ()  
Class content to dictionary

### 4.5.5 Costs Chapter Data Model

**class Chapter** (*chapter\_type, items*)  
Chapter class

**item** (*name*) → *ItemDescription*  
Get specific item by name

**Returns** ItemDescription

**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)
    Fuel class

    to_dictionary ()
        Class content to dictionary
```

#### 4.5.7 Costs Income Data Model

```
class Income (construction_subsidy=None, hvac_subsidy=None, photovoltaic_subsidy=None, electric-
ity_export=None, reductions_tax=None)
    Income class

    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, re-
furbishment=None, refurbishment_unit=None, reposition=None, reposi-
tion_unit=None, lifetime=None)
    Item description class

    to_dictionary ()
        Class content to dictionary
```

#### 4.5.9 Costs Operational Cost Data Model

```
class OperationalCost (fuels, maintenance_heating, maintenance_cooling, maintenance_pv, co2)
    Operational cost class

    to_dictionary ()
        Class content to dictionary
```

## 4.6 Energy Systems

### 4.6.1 Energy Systems Catalog Factory

**class EnergySystemsCatalogFactory** (*handler, base\_path=None*)  
Energy system catalog factory class

### 4.6.2 Energy Systems Content Data Model

**class Content** (*archetypes, systems, generations, distributions, emissions*)  
Content class

**to\_dictionary** ()  
Class content to dictionary

### 4.6.3 Energy Systems Archetype Data Model

**class Archetype** (*lod, name, systems*)  
Archetype class

**to\_dictionary** ()  
Class content to dictionary

### 4.6.4 Energy Systems Distribution System Data Model

**class DistributionSystem** (*system\_id, name, system\_type, supply\_temperature, distribution\_consumption\_fix\_flow, distribution\_consumption\_variable\_flow, heat\_losses*)  
Distribution system class

**to\_dictionary** ()  
Class content to dictionary

### 4.6.5 Energy Systems Emission System Data Model

**class EmissionSystem** (*system\_id, name, system\_type, parasitic\_energy\_consumption*)  
Emission system class

**to\_dictionary** ()  
Class content to dictionary

### 4.6.6 Energy Systems Generation System Data Model

**class GenerationSystem** (*system\_id, name, system\_type, fuel\_type, source\_types, heat\_efficiency, cooling\_efficiency, electricity\_efficiency, source\_temperature, source\_mass\_flow, storage, auxiliary\_equipment*)  
Generation system class

**to\_dictionary** ()  
Class content to dictionary



### 4.6.7 Energy Systems Energy Systems Data Model

**class System**(*lod, system\_id, name, demand\_types, generation\_system, distribution\_system, emission\_system*)

System class

**to\_dictionary**()

Class content to dictionary

## 4.7 Usage

### 4.7.1 Usage Catalog Factory

**class UsageCatalogFactory** (*handler, base\_path=None*)  
Usage catalog factory class

### 4.7.2 Usage Content Data Model

**class Content** (*usages*)  
Content class

**to\_dictionary** ()  
Class content to dictionary

### 4.7.3 Usage Appliances Data Model

**class Appliances** (*density, convective\_fraction, radiative\_fraction, latent\_fraction, schedules*)  
Appliances class

**to\_dictionary** ()  
Class content to dictionary

### 4.7.4 Usage Content Data Model

**class Content** (*usages*)  
Content class

**to\_dictionary** ()  
Class content to dictionary

### 4.7.5 Usage Domestic Hot Water Data Model

**class DomesticHotWater** (*density, peak\_flow, service\_temperature, schedules*)  
DomesticHotWater class

**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*)  
Lighting class

**to\_dictionary** ()  
Class content to dictionary

### 4.7.8 Usage Occupancy Data Model

```
class Occupancy (occupancy_density,          sensible_convective_internal_gain,          sensi-
                ble_radiative_internal_gain, latent_internal_gain, schedules)
    Occupancy class

    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)
    Schedule class

    to_dictionary ()
        Class content to dictionary
```

### 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,    cool-
                        ing_set_point_schedules)
    ThermalControl class

    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)
    Usage class

    to_dictionary ()
        Class content to dictionary
```

## 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  
│   ├── akis_function_to_hub_function.py  
│   ├── ellat_function_to_hub_function.py  
│   ├── hft_function_to_hub_function.py  
│   ├── hub_function_to_ellat_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_usage_to_comnet_usage.py  
│   ├── hub_usage_to_ellat_usage.py  
│   ├── hub_usage_to_hft_usage.py  
│   ├── hub_usage_to_nrcan_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_system_to_hub_energy_generation_system.py  
│   └── pluto_function_to_hub_function.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  
└── utils.py  
  
2 directories, 30 files
```

## 5.2 Configuration Helper

```
class ConfigurationHelper
    Configuration class
```

## 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  
 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}
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'  
AIR = 'Air'  
WATER = 'Water'  
GEOTHERMAL = 'Geothermal'  
DISTRICT\_HEATING\_NETWORK = 'District Heating'  
GRID = 'Grid'  
ONSITE\_ELECTRICITY = 'Onsite Electricity'  
PHOTOVOLTAIC = 'Photovoltaic'  
BOILER = 'Boiler'  
HEAT\_PUMP = 'Heat Pump'  
BASEBOARD = 'Baseboard'  
CHILLER = 'Chiller'  
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)
    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, cli-  
mate_reference_city_longitude)
```

## 5.6 Dictionaries

```
class Dictionaries
```

Dictionaries class

## 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

## Symbols

`_comnet()` (*hub.imports.usage\_factory.UsageFactory method*), 17  
`_eilat()` (*hub.imports.construction\_factory.ConstructionFactory method*), 17  
`_eilat()` (*hub.imports.usage\_factory.UsageFactory method*), 17  
`_epw()` (*hub.imports.weather\_factory.WeatherFactory method*), 18  
`_nrcan()` (*hub.imports.construction\_factory.ConstructionFactory method*), 17  
`_nrcan()` (*hub.imports.usage\_factory.UsageFactory method*), 17  
`_nrel()` (*hub.imports.construction\_factory.ConstructionFactory method*), 17

## A

`add_alias()` (*hub.city\_model\_structure.building.Building method*), 9  
`add_building_alias()` (*hub.city\_model\_structure.city.City method*), 7  
`add_city_object()` (*hub.city\_model\_structure.city.City method*), 7  
`add_city_object()` (*hub.city\_model\_structure.city\_objects\_cluster.CityObjectsCluster method*), 8  
`add_city_objects_cluster()` (*hub.city\_model\_structure.city.City method*), 7  
**Appliances** (*built-in class*), 11, 30  
**Archetype** (*built-in class*), 24, 26, 28

## B

**Building** (*built-in class*), 9  
`building_alias()` (*hub.city\_model\_structure.city.City method*), 7  
**BuildingsCluster** (*built-in class*), 9

## C

**CapitalCost** (*built-in class*), 26  
**Catalog** (*built-in class*), 22  
**Chapter** (*built-in class*), 26  
`chapter()` (*hub.catalog\_factories.data\_models.cost.capital\_cost.CapitalCost method*), 26  
**City** (*built-in class*), 7  
`city_mapping()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 40  
`city_object()` (*hub.city\_model\_structure.city.City method*), 7  
**CityObject** (*built-in class*), 8  
**CityObjectsCluster** (*built-in class*), 8  
**ConfigurationHelper** (*built-in class*), 33  
**Construction** (*built-in class*), 24  
**ConstructionCatalogFactory** (*built-in class*), 24  
**ConstructionFactory** (*built-in class*), 17  
**Content** (*built-in class*), 23, 24, 26, 28, 30  
`coordinate_to_map_point()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 40  
**CostsCatalogFactory** (*built-in class*), 26

## D

**Dictionaries** (*built-in class*), 41  
`distance_between_points()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 40  
`distance_to_point()` (*hub.city\_model\_structure.attributes.plane.Plane method*), 10  
`distance_to_point()` (*hub.city\_model\_structure.attributes.point.Point method*), 10  
**DistributionSystem** (*built-in class*), 28  
`divide()` (*hub.city\_model\_structure.attributes.polygon.Polygon method*), 10  
`divide()` (*hub.city\_model\_structure.building\_demand.surface.Surface method*), 12  
`divide_mesh_by_plane()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 40  
**DomesticHotWater** (*built-in class*), 30

## E

**Edge** (*built-in class*), 9  
**EmissionSystem** (*built-in class*), 28  
**EnergySystemsCatalogFactory** (*built-in class*), 28  
`enrich()` (*hub.imports.construction\_factory.ConstructionFactory method*), 17  
`enrich()` (*hub.imports.usage\_factory.UsageFactory method*), 17  
`enrich()` (*hub.imports.weather\_factory.WeatherFactory method*), 18  
`entries()` (*hub.catalog\_factories.catalog.Catalog method*), 22  
`export()` (*hub.exports.exports\_factory.ExportsFactory method*), 19  
**ExportsFactory** (*built-in class*), 19

## F

`factor()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 41  
**Fuel** (*built-in class*), 27

## G

**GenerationSystem** (*built-in class*), 28

**GeometryFactory** (*built-in class*), 17  
**GeometryHelper** (*built-in class*), 40  
`get_entry()` (*hub.catalog\_factories.catalog.Catalog method*), 22  
`get_location()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 41  
**GreeneryCatalogFactory** (*built-in class*), 23

## H

**Household** (*built-in class*), 11

## I

**Income** (*built-in class*), 27  
**InternalGain** (*built-in class*), 11  
**InternalZone** (*built-in class*), 11  
`item()` (*hub.catalog\_factories.data\_models.cost.chapter.Chapter method*), 26  
**ItemDescription** (*built-in class*), 27

## L

**Layer** (*built-in class*), 12, 24  
**LevelOfDetail** (*built-in class*), 9  
**Lighting** (*built-in class*), 12, 30  
`load()` (*hub.city\_model\_structure.city.City static method*), 7  
`load_compressed()` (*hub.city\_model\_structure.city.City static method*), 7  
**Location** (*built-in class*), 41

## M

**Material** (*built-in class*), 25  
`merge()` (*hub.city\_model\_structure.city.City method*), 7

## N

`names()` (*hub.catalog\_factories.catalog.Catalog method*), 22  
**Network** (*built-in class*), 9  
**Node** (*built-in class*), 9

## O

`obj_export()` (*hub.city\_model\_structure.attributes.polyhedron.Polyhedron method*), 10  
**Occupancy** (*built-in class*), 12, 31  
**OperationalCost** (*built-in class*), 27

## P

**PartsConsistingBuilding** (*built-in class*), 9  
**Plane** (*built-in class*), 10  
**Plant** (*built-in class*), 13, 23  
**PlantPercentage** (*built-in class*), 23  
**Point** (*built-in class*), 10  
**Polygon** (*built-in class*), 10  
**Polyhedron** (*built-in class*), 10

## R

**Record** (*built-in class*), 11  
`region()` (*hub.city\_model\_structure.city.City method*), 7  
`remove_city_object()` (*hub.city\_model\_structure.city.City method*), 8

## S

`save()` (*hub.city\_model\_structure.city.City method*), 8  
`save_compressed()` (*hub.city\_model\_structure.city.City method*), 8  
**Schedule** (*built-in class*), 11, 31  
`segment_list_to_trimesh()` (*hub.helpers.geometry\_helper.GeometryHelper static method*), 41  
**Sensor** (*built-in class*), 13  
**SensorMeasure** (*built-in class*), 13  
**SensorType** (*built-in class*), 14  
`show()` (*hub.city\_model\_structure.attributes.polyhedron.Polyhedron method*), 10  
**Soil** (*built-in class*), 13, 23  
**Station** (*built-in class*), 14  
`stl_export()` (*hub.city\_model\_structure.attributes.polyhedron.Polyhedron method*), 10  
**Storey** (*built-in class*), 12

Surface (*built-in class*), 12  
 surface() (*hub.city\_model\_structure.city\_object.CityObject method*), 8  
 surface\_by\_id() (*hub.city\_model\_structure.city\_object.CityObject method*), 8  
 System (*built-in class*), 29

## T

ThermalBoundary (*built-in class*), 12  
 ThermalControl (*built-in class*), 12, 31  
 ThermalOpening (*built-in class*), 13  
 ThermalZone (*built-in class*), 13  
 TimeSeries (*built-in class*), 11  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.archetype.Archetype method*), 24  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.construction.Construction method*), 24  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.content.Content method*), 24  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.layer.Layer method*), 24  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.material.Material method*), 25  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.window.Window method*), 25  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.archetype.Archetype method*), 26  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.capital\_cost.CapitalCost method*), 26  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.chapter.Chapter method*), 26  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.content.Content method*), 26  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.fuel.Fuel method*), 27  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.income.Income method*), 27  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.item\_description.ItemDescription method*), 27  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.operational\_cost.OperationalCost method*), 27  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.archetype.Archetype method*), 28  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.content.Content method*), 28  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.distribution\_system.DistributionSystem method*), 28  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.emission\_system.EmissionSystem method*), 28  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.generation\_system.GenerationSystem method*), 28  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.system.System method*), 29  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.content.Content method*), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.plant.Plant method*), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.plant\_percentage.PlantPercentage method*), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.soil.Soil method*), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.vegetation.Vegetation method*), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.appliances.Appliances method*), 30  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.content.Content method*), 30  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.domestic\_hot\_water.DomesticHotWater method*), 30  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.lighting.Lighting method*), 30  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.occupancy.Occupancy method*), 31  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.schedule.Schedule method*), 31  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.thermal\_control.ThermalControl method*), 31  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.usage.Usage method*), 31  
 triangle\_mesh() (*hub.city\_model\_structure.attributes.polygon.Polygon static method*), 10

## U

Usage (*built-in class*), 13, 31  
 UsageCatalogFactory (*built-in class*), 30  
 UsageFactory (*built-in class*), 17

## V

Vegetation (*built-in class*), 13, 23

## W

WeatherFactory (*built-in class*), 18  
 Window (*built-in class*), 25