

---

# **CERC hub reference manual**

*Release 0.3.0.3*

## **CERC Next-Generation Cities**

**Dec 04, 2024**

# 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 . . . . .	10
2.3.3	City Objects Cluster . . . . .	12
2.3.4	Building . . . . .	12
2.3.5	Parts Consisting Building . . . . .	16
2.3.6	Buildings Cluster . . . . .	16
2.3.7	Network . . . . .	17
2.3.8	Level of detail . . . . .	17
2.3.9	Edge . . . . .	18
2.3.10	Node . . . . .	18
2.3.11	Plane . . . . .	18
2.3.12	Point . . . . .	19
2.3.13	Polygon . . . . .	19
2.3.14	Polyhedron . . . . .	21
2.3.15	Record . . . . .	22
2.3.16	Schedule . . . . .	22
2.3.17	Time Series . . . . .	23
2.3.18	Appliances . . . . .	24
2.3.19	Household . . . . .	24
2.3.20	Internal Gain . . . . .	25
2.3.21	Internal Zone . . . . .	25
2.3.22	Layer . . . . .	26
2.3.23	Lighting . . . . .	27
2.3.24	Occupancy . . . . .	28
2.3.25	Storey . . . . .	29

2.3.26	Surface	29
2.3.27	Thermal Boundary	32
2.3.28	Thermal Control	33
2.3.29	Thermal Opening	34
2.3.30	Thermal Zone	35
2.3.31	Usage	37
2.3.32	Plant	38
2.3.33	Soil	39
2.3.34	Vegetation	40
2.3.35	Sensor	41
2.3.36	Sensor Measure	41
2.3.37	Sensor Type	42
2.3.38	Station	42
<b>3</b>	<b>Factories</b>	<b>43</b>
3.1	Folder structure	44
3.1.1	Imports	44
3.1.2	Exports	44
3.2	Imports Classes	45
3.2.1	Construction Factory	45
3.2.2	Geometry Factory	45
3.2.3	Usage Factory	46
3.2.4	Weather Factory	46
3.3	Exports	47
3.3.1	Export Factory	47
<b>4</b>	<b>Catalogs</b>	<b>48</b>
4.1	Folder structure	49
4.1.1	Catalogs	49
4.2	Catalog Base Class	50
4.2.1	Catalog	50
4.3	Greenery	51
4.3.1	Greenery Catalog Factory	51
4.3.2	Greenery Content Data Model	51
4.3.3	Greenery Plant Data Model	51
4.3.4	Greenery Plant Percentage Data Model	52
4.3.5	Greenery Plant Soil Data Model	53
4.3.6	Greenery Vegetation Data Model	54
4.4	Construction	56
4.4.1	Construction Catalog Factory	56
4.4.2	Construction Content Data Model	56
4.4.3	Construction Archetype Data Model	57
4.4.4	Construction Construction Data Model	58
4.4.5	Construction Layer Data Model	59
4.4.6	Construction Material Data Model	59
4.4.7	Construction Window Data Model	60
4.5	Costs	62
4.5.1	Costs Catalog Factory	62
4.5.2	Costs Content Data Model	62
4.5.3	Costs Archetype Data Model	62
4.5.4	Costs Capital Cost Data Model	63
4.5.5	Costs Chapter Data Model	64
4.5.6	Costs Fuel Data Model	64
4.5.7	Costs Income Data Model	65

4.5.8	Costs Item Description Data Model . . . . .	65
4.5.9	Costs Operational Cost Data Model . . . . .	66
4.6	Energy Systems . . . . .	67
4.6.1	Energy Systems Catalog Factory . . . . .	67
4.6.2	Energy Systems Content Data Model . . . . .	67
4.6.3	Energy Systems Archetype Data Model . . . . .	67
4.6.4	Energy Systems Distribution System Data Model . . . . .	68
4.6.5	Energy Systems Emission System Data Model . . . . .	69
4.6.6	Energy Systems Generation System Data Model . . . . .	69
4.6.7	Energy Systems Energy Systems Data Model . . . . .	70
4.7	Usage . . . . .	72
4.7.1	Usage Catalog Factory . . . . .	72
4.7.2	Usage Content Data Model . . . . .	72
4.7.3	Usage Appliances Data Model . . . . .	72
4.7.4	Usage Content Data Model . . . . .	73
4.7.5	Usage Domestic Hot Water Data Model . . . . .	73
4.7.6	Usage Internal Gain Data Model . . . . .	74
4.7.7	Usage Lighting Data Model . . . . .	74
4.7.8	Usage Occupancy Data Model . . . . .	74
4.7.9	Usage Schedule Data Model . . . . .	75
4.7.10	Usage Thermal Control Data Model . . . . .	76
4.7.11	Usage Usage Data Model . . . . .	77
<b>5</b>	<b>Helpers</b>	<b>78</b>
5.1	Folder structure . . . . .	78
5.2	Configuration Helper . . . . .	79
5.3	Constants . . . . .	81
5.4	Geometry Helper . . . . .	88
5.5	Location . . . . .	89
5.6	Dictionaries . . . . .	89
<b>6</b>	<b>Additional Files</b>	<b>92</b>
6.1	Readme . . . . .	92
6.2	License . . . . .	92
6.3	Code of conduct . . . . .	92
6.4	How to contribute . . . . .	92
6.5	Coding style . . . . .	92
<b>Index</b>		<b>93</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.

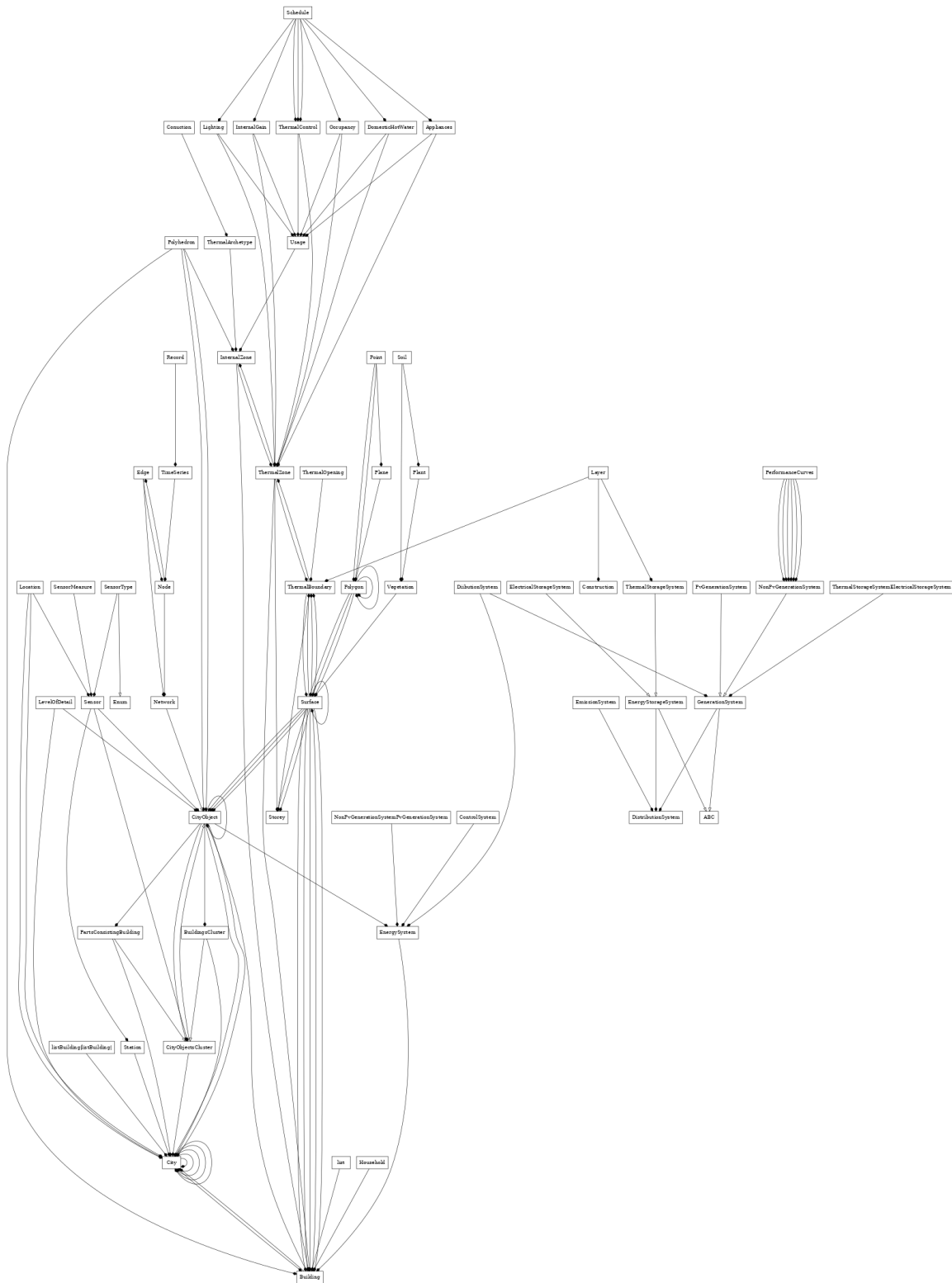
---

**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
```

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/m<sup>3</sup>**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/m<sup>2</sup>

**Returns** dict

**property global\_irradiance\_tilted** → dict

Get global irradiance on a tilted surface in W/m<sup>2</sup>

**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 m<sup>2</sup>

**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 of 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/m<sup>2</sup>K)

**Returns** None or float

**property hi** → Optional[float]

Get internal convective heat transfer coefficient (W/m<sup>2</sup>K)

**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/m<sup>2</sup>K

internal and external convective coefficient in W/m<sup>2</sup>K 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/m<sup>2</sup>K)  
**Returns** None or float

**property hi** → Optional[float]  
 Get internal convective heat transfer coefficient (W/m<sup>2</sup>K)  
**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/m<sup>2</sup>K  
**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/m<sup>3</sup>**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 absorptance

**Returns** float**property thermal\_absorptance**

Get soil thermal absorptance

**Returns** float**property visible\_absorptance**

Get soil visible absorptance

**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]

## 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
│   ├── inset_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
│   ├── id_files
│   │   ├── base.idf
│   │   ├── Energy.idf
│   │   ├── Minimal.idf
│   │   └── outputs.idf
│   ├── id_helper
│   │   ├── id_appliance.py
│   │   ├── id_base.py
│   │   ├── id_construction.py
│   │   ├── id_dhw.py
│   │   ├── id_file_schedule.py
│   │   ├── id_heating_system.py
│   │   ├── id_infiltration.py
│   │   ├── id_lighting.py
│   │   ├── id_material.py
│   │   ├── id_occupancy.py
│   │   ├── id_schedule.py
│   │   ├── id_shading.py
│   │   ├── id_surfaces.py
│   │   ├── id_thermostat.py
│   │   ├── id_ventilation.py
│   │   ├── id_window.py
│   │   ├── id_windows_constructions.py
│   │   ├── id_windows_materials.py
│   │   └── id_zone.py
│   └── __init__.py
├── inset
│   ├── __init__.py
│   └── inset_monthly_energy_balance.py
├── cerc_id.py
├── energy_idb.py
├── idf.py
├── __init__.py
├── energy_systems
│   ├── air_source_tp_export.py
│   ├── heat_pump_export.py
│   ├── water_to_water_tp_export.py
│   ├── energy_building_exports_factory.py
│   ├── energy_systems_exports_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

## 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
│   ├── eliat_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_catalogue.py
├── greenery
│   ├── ecore_greenery
│   │   ├── greenerycatalog.ecore
│   │   ├── greenerycatalog_no_quantities.ecore
│   │   └── greenerycatalog.py
│   ├── greenery_catalog.py
│   └── __init__.py
├── usage
│   ├── comnet_catalog.py
│   ├── eliat_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/m<sup>3</sup>

**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 absorptance

**Returns** float

**property** thermal\_absorptance

Get soil thermal absorptance

**Returns** float

**to\_dictionary**()

Class content to dictionary

**property** visible\_absorptance

Get soil visible absorptance

**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/m<sup>3</sup>

**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 absorptance

**Returns** float

**property soil\_thermal\_absorptance**

Get soil thermal absorptance

**Returns** float

**property soil\_thickness**

Get soil thickness in m

**Returns** float

**property soil\_visible\_absorptance**

Get soil visible absorptance

**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/m<sup>3</sup>K**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/m<sup>3</sup>

**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/m<sup>2</sup>K

**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/m<sup>2</sup>

**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

## 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
│   ├── ahrs_function_to_hub_function.py
│   ├── elat_function_to_hub_function.py
│   ├── hr_function_to_hub_function.py
│   ├── hub_function_to_elat_construction_function.py
│   ├── hub_function_to_montreal_custom_costs_function.py
│   ├── hub_function_to_mreah_construction_function.py
│   ├── hub_function_to_mreah_construction_function.py
│   ├── hub_function_to_mreah_construction_function.py
│   ├── hub_function_to_palmia_construction_function.py
│   ├── hub_usage_to_commit_usage.py
│   ├── hub_usage_to_elat_usage.py
│   ├── hub_usage_to_hr_usage.py
│   ├── hub_usage_to_mreah_usage.py
│   ├── hub_usage_to_palmia_usage.py
│   ├── __init__.py
│   ├── montreal_custom_tsl_to_hub_tsl.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_tsl_to_hub_tsl.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
│   ├── palmia_function_to_hub_function.py
│   └── palmia_function_to_hub_function.py
├── parameters
│   ├── __init__.py
│   ├── hr_usage_to_hub.py
│   ├── string_usage_to_hub.py
├── peak_calculation
│   ├── __init__.py
│   ├── loads_calculation.py
├── utility
│   ├── configuration_helper.py
│   ├── constants.py
│   ├── dictionaries.py
│   ├── geometry_helper.py
│   ├── __init__.py
│   ├── location.py
│   ├── monetary_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/m<sup>2</sup>K

**property convective\_heat\_transfer\_coefficient\_interior** → float

Get configured convective heat transfer coefficient for surfaces inside the building

**Returns** 3.5 W/m<sup>2</sup>K

**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

## 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

\_cesiumjs\_tileset (*hub.exports.exports\_factory* :property: ExportsFactory property), 47  
 \_citygml (*hub.imports.geometry\_factory* :property: GeometryFactory property), 45  
 \_comnet (*hub.catalog\_factories.usage\_catalog\_factory* :property: UsageCatalogFactory property), 72  
 \_comnet() (*hub.imports.usage\_factory*:UsageFactory method), 46  
 \_eilat (*hub.catalog\_factories.construction\_catalog\_factory* :property: ConstructionCatalogFactory property), 56  
 \_eilat (*hub.catalog\_factories.usage\_catalog\_factory* :property: UsageCatalogFactory property), 72  
 \_eilat() (*hub.imports.construction\_factory*:ConstructionFactory method), 45  
 \_eilat() (*hub.imports.usage\_factory*:UsageFactory method), 46  
 \_epw() (*hub.imports.weather\_factory*:WeatherFactory method), 46  
 \_geojson (*hub.imports.geometry\_factory* :property: GeometryFactory property), 45  
 \_montreal\_custom (*hub.catalog\_factories.costs\_catalog\_factory* :property: CostsCatalogFactory property), 62  
 \_montreal\_custom (*hub.catalog\_factories.energy\_systems\_catalog\_factory* :property: EnergySystemsCatalogFactory property), 67  
 \_montreal\_future (*hub.catalog\_factories.energy\_systems\_catalog\_factory* :property: EnergySystemsCatalogFactory property), 67  
 \_nrcan (*hub.catalog\_factories.construction\_catalog\_factory* :property: ConstructionCatalogFactory property), 56  
 \_nrcan (*hub.catalog\_factories.usage\_catalog\_factory* :property: UsageCatalogFactory property), 72  
 \_nrcan() (*hub.imports.construction\_factory*:ConstructionFactory method), 45  
 \_nrcan() (*hub.imports.usage\_factory*:UsageFactory method), 46  
 \_nrel (*hub.catalog\_factories.construction\_catalog\_factory* :property: ConstructionCatalogFactory property), 56  
 \_nrel (*hub.catalog\_factories.greenery\_catalog\_factory* :property: GreeneryCatalogFactory property), 51  
 \_nrel() (*hub.imports.construction\_factory*:ConstructionFactory method), 45  
 \_obj (*hub.exports.exports\_factory* :property: ExportsFactory property), 47  
 \_obj (*hub.imports.geometry\_factory* :property: GeometryFactory property), 45  
 \_palma (*hub.catalog\_factories.construction\_catalog\_factory* :property: ConstructionCatalogFactory property), 56  
 \_palma (*hub.catalog\_factories.energy\_systems\_catalog\_factory* :property: EnergySystemsCatalogFactory property), 67  
 \_palma (*hub.catalog\_factories.usage\_catalog\_factory* :property: UsageCatalogFactory property), 72  
 \_palma() (*hub.imports.construction\_factory*:ConstructionFactory method), 45  
 \_palma() (*hub.imports.usage\_factory*:UsageFactory method), 46  
 \_sra (*hub.exports.exports\_factory* :property: ExportsFactory property), 47  
 \_stl (*hub.exports.exports\_factory* :property: ExportsFactory property), 47

## A

add\_alias() (*hub.city\_model\_structure.building*:Building method), 12  
 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), 12  
 add\_city\_objects\_cluster() (*hub.city\_model\_structure.city*:City method), 7  
 additional\_thermal\_bridge\_u\_value (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 35  
 air\_gap (*hub.catalog\_factories.data\_models.greenery.vegetation* :property: Vegetation property), 54  
 air\_gap (*hub.city\_model\_structure.greenery.vegetation* :property: Vegetation property), 40  
 aliases (*hub.city\_model\_structure.building* :property: Building property), 12  
 alks\_function\_to\_hub\_function (*hub.helpers.dictionaries* :property: Dictionaries property), 89  
 Appliances (built-in class), 24, 72  
 appliances (*hub.catalog\_factories.data\_models.usages.usage* :property: Usage property), 77  
 appliances (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 35  
 appliances (*hub.city\_model\_structure.building\_demand.usage* :property: Usage property), 37  
 appliances\_electrical\_demand (*hub.city\_model\_structure.building* :property: Building property), 12  
 appliances\_peak\_load (*hub.city\_model\_structure.building* :property: Building property), 12  
 Archetype (built-in class), 57, 62, 67  
 archetypes (*hub.catalog\_factories.data\_models.construction.content* :property: Content property), 56  
 archetypes (*hub.catalog\_factories.data\_models.cost.content* :property: Content property), 62  
 archetypes (*hub.catalog\_factories.data\_models.energy\_systems.content* :property: Content property), 67  
 area (*hub.city\_model\_structure.attributes.polygon* :property: Polygon property), 19  
 area (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalZone property), 25  
 area (*hub.city\_model\_structure.building\_demand.thermal\_opening* :property: ThermalOpening property), 34  
 associated\_thermal\_boundaries (*hub.city\_model\_structure.building\_demand.surface* :property: Surface property), 29  
 attic\_heated (*hub.city\_model\_structure.building* :property: Building property), 12  
 average\_internal\_gain (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalGain property), 25  
 average\_storey\_height (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype property), 57  
 average\_storey\_height (*hub.city\_model\_structure.building* :property: Building property), 13  
 azimuth (*hub.city\_model\_structure.building\_demand.surface* :property: Surface property), 29

## B

basement\_heated (*hub.city\_model\_structure.building* :property: Building property), 13  
 beam (*hub.city\_model\_structure.city\_object* :property: CityObject property), 10  
 Building (built-in class), 12  
 building\_alias() (*hub.city\_model\_structure.city*:City method), 7  
 buildings (*hub.city\_model\_structure.city* :property: City property), 7  
 buildings\_clusters (*hub.city\_model\_structure.city* :property: City property), 7  
 BuildingsCluster (built-in class), 16

## C

capital\_cost (*hub.catalog\_factories.data\_models.cost.archetype* :property: Archetype property), 62  
 CapitalCost (built-in class), 63  
 Catalog (built-in class), 50  
 catalog (*hub.catalog\_factories.construction\_catalog\_factory* :property: ConstructionCatalogFactory property), 56  
 catalog (*hub.catalog\_factories.costs\_catalog\_factory* :property: CostsCatalogFactory property), 62  
 catalog (*hub.catalog\_factories.energy\_systems\_catalog\_factory* :property: EnergySystemsCatalogFactory property), 67  
 catalog (*hub.catalog\_factories.greenery\_catalog\_factory* :property: GreeneryCatalogFactory property), 51  
 catalog (*hub.catalog\_factories.usage\_catalog\_factory* :property: UsageCatalogFactory property), 72  
 category (*hub.catalog\_factories.data\_models.greenery.plant* :property: Plant property), 51  
 category (*hub.catalog\_factories.data\_models.greenery.vegetation* :property: Vegetation property), 54  
 centroid (*hub.city\_model\_structure.attributes.polyhedron* :property: Polyhedron property), 21  
 centroid (*hub.city\_model\_structure.city\_object* :property: CityObject property), 10  
 Chapter (built-in class), 64  
 chapter() (*hub.catalog\_factories.data\_models.cost.capital\_cost*:CapitalCost method), 63  
 chapter\_type (*hub.catalog\_factories.data\_models.cost.chapter* :property: Chapter property), 64  
 City (built-in class), 7  
 city (*hub.city\_model\_structure.building* :property: Building property), 13  
 city (*hub.helpers.location* :property: Location property), 89  
 city (*hub.imports.geometry\_factory* :property: GeometryFactory property), 45  
 city\_mapping() (*hub.helpers.geometry\_helper*:GeometryHelper static method), 88  
 city\_object() (*hub.city\_model\_structure.city*:City method), 7  
 city\_objects (*hub.city\_model\_structure.buildings\_cluster* :property: BuildingsCluster property), 16  
 city\_objects (*hub.city\_model\_structure.city* :property: City property), 7  
 city\_objects (*hub.city\_model\_structure.city\_objects\_cluster* :property: CityObjectsCluster property), 12  
 city\_objects (*hub.city\_model\_structure.parts\_consisting\_building* :property: PartsConsistingBuilding property), 16  
 city\_objects\_clusters (*hub.city\_model\_structure.city* :property: City property), 7  
 CityObject (built-in class), 10  
 CityObjectsCluster (built-in class), 12  
 city (*hub.city\_model\_structure.city* :property: City property), 7  
 climate\_file (*hub.city\_model\_structure.city* :property: City property), 8  
 climate\_reference\_city (*hub.helpers.location* :property: Location property), 89  
 climate\_reference\_city\_latitude (*hub.helpers.location* :property: Location property), 89  
 climate\_reference\_city\_longitude (*hub.helpers.location* :property: Location property), 89  
 climate\_zone (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype property), 57  
 cluster\_id (*hub.catalog\_factories.data\_models.energy\_systems.archetype* :property: Archetype property), 67  
 co2 (*hub.catalog\_factories.data\_models.cost.operational\_cost* :property: OperationalCost property), 66  
 co2\_sequestration (*hub.catalog\_factories.data\_models.greenery.plant* :property: Plant property), 51  
 co2\_sequestration (*hub.city\_model\_structure.greenery.plant* :property: Plant property), 38  
 cold\_water\_temperature (*hub.city\_model\_structure.building* :property: Building property), 13  
 cold\_water\_temperature (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comet\_lighting\_convective (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comet\_lighting\_latent (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comet\_lighting\_radiant (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comet\_occupancy\_sensible\_convective (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comnet\_occupancy\_sensible\_radiant (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comnet\_plugs\_convective (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comnet\_plugs\_latent (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 comnet\_plugs\_radiant (*hub.helpers.configuration\_helper* :property: ConfigurationHelper property), 79  
 conductivity (*hub.catalog\_factories.data\_models.construction.material* :property: Material property), 59  
 conductivity (*hub.city\_model\_structure.building\_demand.layer* :property: Layer property), 26  
 conductivity (*hub.city\_model\_structure.building\_demand.thermal\_opening* :property: ThermalOpening property), 34  
 configuration\_schema (*hub.catalog\_factories.data\_models.energy\_systems.system* :property: System property), 70  
 ConfigurationHelper (built-in class), 79  
 Construction (built-in class), 58  
 construction (*hub.city\_model\_structure.level\_of\_detail* :property: LevelOfDetail property), 17  
 construction\_name (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 32  
 construction\_name (*hub.city\_model\_structure.building\_demand.thermal\_opening* :property: ThermalOpening property), 34  
 construction\_name (*hub.city\_model\_structure.building\_demand.thermal\_opening* :property: ThermalOpening property), 34  
 construction\_period (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype property), 57  
 construction\_subsidy (*hub.catalog\_factories.data\_models.cost.income* :property: Income property), 65  
 ConstructionCatalogFactory (built-in class), 56  
 ConstructionFactory (built-in class), 45  
 constructions (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype property), 57  
 constructions (*hub.catalog\_factories.data\_models.construction.content* :property: Content property), 56  
 Content (built-in class), 51, 56, 62, 67, 72, 73  
 convective\_fraction (*hub.catalog\_factories.data\_models.usages.appliances* :property: Appliances property), 72  
 convective\_fraction (*hub.catalog\_factories.data\_models.usages.lighting* :property: Lighting property), 74  
 convective\_fraction (*hub.city\_model\_structure.building\_demand.appliances* :property: Appliances property), 24  
 convective\_fraction (*hub.city\_model\_structure.building\_demand.internal\_gain* :property: InternalGain property), 25  
 convective\_fraction (*hub.city\_model\_structure.building\_demand.lighting* :property: Lighting property), 27

convective\_heat\_transfer\_coefficient\_exterior (hub.helpers.configuration\_helper :property: ConfigurationHelper property), 79  
 convective\_heat\_transfer\_coefficient\_interior (hub.helpers.configuration\_helper :property: ConfigurationHelper property), 79  
 cooling\_consumption (hub.city\_model\_structure.building :property: Building property), 13  
 cooling\_demand (hub.city\_model\_structure.building :property: Building property), 13  
 cooling\_peak\_load (hub.city\_model\_structure.building :property: Building property), 13  
 cooling\_set\_point\_schedules (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 cooling\_set\_point\_schedules (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 33  
 coordinate\_to\_map\_point() (hub.helpers.geometry\_helper.GeometryHelper static method), 88  
 coordinates (hub.city\_model\_structure.attributes.point :property: Point property), 19  
 coordinates (hub.city\_model\_structure.attributes.polygon :property: Polygon property), 19  
 copy (hub.city\_model\_structure.city :property: City property), 8  
 CostsCatalogFactory (built-in class), 62  
 country (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 62  
 country (hub.helpers.location :property: Location property), 89  
 country\_code (hub.city\_model\_structure.city :property: City property), 8  
 currency (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 62

## D

data\_type (hub.catalog\_factories.data\_models.usages.schedule :property: Schedule property), 75  
 data\_type (hub.city\_model\_structure.attributes.schedule :property: Schedule property), 22  
 day\_types (hub.catalog\_factories.data\_models.usages.schedule :property: Schedule property), 75  
 day\_types (hub.city\_model\_structure.attributes.schedule :property: Schedule property), 22  
 days\_year (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77  
 days\_year (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 days\_year (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 demand\_types (hub.catalog\_factories.data\_models.energy\_systems.system :property: System property), 70  
 density (hub.catalog\_factories.data\_models.construction.material :property: Material property), 59  
 density (hub.catalog\_factories.data\_models.usages.appliances :property: Appliances property), 72  
 density (hub.catalog\_factories.data\_models.usages.domestic\_hot\_water :property: DomesticHotWater property), 73  
 density (hub.catalog\_factories.data\_models.usages.lighting :property: Lighting property), 74  
 density (hub.city\_model\_structure.building\_demand.appliances :property: Appliances property), 24  
 density (hub.city\_model\_structure.building\_demand.layer :property: Layer property), 26  
 density (hub.city\_model\_structure.building\_demand.lighting :property: Lighting property), 27  
 design\_allowance (hub.catalog\_factories.data\_models.cost.capital\_cost :property: CapitalCost property), 63  
 detailed\_polyhedron (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 Dictionaries (built-in class), 89  
 diffuse (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 direct\_normal (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 distance\_between\_points() (hub.helpers.geometry\_helper.GeometryHelper static method), 88  
 distance\_to\_point() (hub.city\_model\_structure.attributes.plane :property: Plane method), 18  
 distance\_to\_point() (hub.city\_model\_structure.attributes.point :property: Point method), 19  
 distribution\_consumption\_fix\_flow (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 distribution\_consumption\_variable\_flow (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 distribution equipments (hub.catalog\_factories.data\_models.energy\_systems.content :property: Content property), 67  
 distribution\_systems (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 69  
 distribution\_systems (hub.catalog\_factories.data\_models.energy\_systems.system :property: System property), 70  
 distribution\_systems\_electrical\_consumption (hub.city\_model\_structure.building :property: Building property), 13  
 DistributionSystem (built-in class), 68  
 divide() (hub.city\_model\_structure.attributes.polygon.Polygon method), 19  
 divide() (hub.city\_model\_structure.building\_demand.surface :property: Surface method), 30  
 divide\_mesh\_by\_plane() (hub.helpers.geometry\_helper.GeometryHelper static method), 88  
 domestic\_hot\_water (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77  
 domestic\_hot\_water (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 domestic\_hot\_water (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 domestic\_hot\_water\_consumption (hub.city\_model\_structure.building :property: Building property), 13  
 domestic\_hot\_water\_heat\_demand (hub.city\_model\_structure.building :property: Building property), 13  
 domestic\_hot\_water\_peak\_load (hub.city\_model\_structure.building :property: Building property), 13  
 DomesticHotWater (built-in class), 73  
 dry\_conductivity (hub.catalog\_factories.data\_models.greenery.soil :property: Soil property), 53  
 dry\_conductivity (hub.city\_model\_structure.greenery.soil :property: Soil property), 39  
 dry\_density (hub.catalog\_factories.data\_models.greenery.soil :property: Soil property), 53  
 dry\_density (hub.city\_model\_structure.greenery.soil :property: Soil property), 39  
 dry\_soil\_conductivity (hub.catalog\_factories.data\_models.greenery.vegetation :property: Vegetation property), 54  
 dry\_soil\_density (hub.catalog\_factories.data\_models.greenery.vegetation :property: Vegetation property), 54  
 dry\_soil\_specific\_heat (hub.catalog\_factories.data\_models.greenery.vegetation :property: Vegetation property), 54  
 dry\_specific\_heat (hub.catalog\_factories.data\_models.greenery.soil :property: Soil property), 53  
 dry\_specific\_heat (hub.city\_model\_structure.greenery.soil :property: Soil property), 39

## E

eave\_height (hub.city\_model\_structure.building :property: Building property), 13  
 Edge (built-in class), 18  
 edges (hub.city\_model\_structure.attributes.node :property: Node property), 18  
 edges (hub.city\_model\_structure.attributes.polygon :property: Polygon property), 20  
 edges (hub.city\_model\_structure.network :property: Network property), 17  
 effective\_thermal\_capacity (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 eilat\_function\_to\_hub\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 electricity\_export (hub.catalog\_factories.data\_models.cost.income :property: Income property), 65  
 emission\_systems (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 EmissionSystem (built-in class), 69  
 end\_of\_life\_cost (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 62  
 energy\_consumption\_breakdown (hub.city\_model\_structure.building :property: Building property), 14  
 energy\_storage\_systems (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 energy\_storage\_systems (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 energy\_systems (hub.city\_model\_structure.building :property: Building property), 14

energy\_systems (hub.city\_model\_structure.level\_of\_detail :property: LevelOfDetail property), 17  
 energy\_systems\_archetype\_cluster\_id (hub.city\_model\_structure.building :property: Building property), 14  
 energy\_systems\_archetype\_name (hub.city\_model\_structure.building :property: Building property), 14  
 EnergySystemsCatalogFactory (built-in class), 67  
 enrich() (hub.imports.construction\_factory.ConstructionFactory method), 45  
 enrich() (hub.imports.usage\_factory.UsageFactory method), 46  
 enrich() (hub.imports.weather\_factory.WeatherFactory method), 46  
 entries() (hub.catalog\_factories.catalog.Catalog method), 50  
 equation (hub.city\_model\_structure.attributes.plane :property: Plane property), 18  
 export() (hub.exports.exports\_factory.ExportsFactory method), 47  
 ExportsFactory (built-in class), 47  
 external\_temperature (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 extra\_losses\_due\_to\_thermal\_bridges (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57

## F

faces (hub.city\_model\_structure.attributes.polygon :property: Polygon property), 20  
 faces (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 factor() (hub.helpers.geometry\_helper.GeometryHelper static method), 88  
 fixed\_monthly (hub.catalog\_factories.data\_models.cost.fuel :property: Fuel property), 64  
 fixed\_power (hub.catalog\_factories.data\_models.cost.fuel :property: Fuel property), 64  
 flag (hub.city\_model\_structure.attributes.record :property: Record property), 22  
 floor\_area (hub.city\_model\_structure.building :property: Building property), 14  
 floor\_area (hub.city\_model\_structure.building\_demand.store :property: Store property), 29  
 footprint\_area (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 frame\_ratio (hub.catalog\_factories.data\_models.construction.window :property: Window property), 60  
 frame\_ratio (hub.city\_model\_structure.building\_demand.thermal\_opening :property: ThermalOpening property), 34  
 Fuel (built-in class), 64  
 fuel\_type (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 fuels (hub.catalog\_factories.data\_models.cost.operational\_cost :property: OperationalCost property), 66  
 function (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 function (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 62  
 function (hub.city\_model\_structure.building :property: Building property), 14

## G

g\_value (hub.catalog\_factories.data\_models.construction.window :property: Window property), 60  
 g\_value (hub.city\_model\_structure.building\_demand.thermal\_opening :property: ThermalOpening property), 34  
 general\_chapters (hub.catalog\_factories.data\_models.cost.capital\_cost :property: CapitalCost property), 63  
 generation equipments (hub.catalog\_factories.data\_models.energy\_systems.content :property: Content property), 67  
 generation\_systems (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 generation\_systems (hub.catalog\_factories.data\_models.energy\_systems.system :property: System property), 71  
 GenerationSystem (built-in class), 69  
 generic\_energy\_systems (hub.city\_model\_structure.city :property: City property), 8  
 geometry (hub.city\_model\_structure.building\_demand.internal\_zone :property: InternalZone property), 26  
 geometry (hub.city\_model\_structure.level\_of\_detail :property: LevelOfDetail property), 17  
 GeometryFactory (built-in class), 45  
 GeometryHelper (built-in class), 88  
 get\_entry() (hub.catalog\_factories.catalog.Catalog method), 50  
 get\_location() (hub.helpers.geometry\_helper.GeometryHelper static method), 88  
 global\_horizontal (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 global\_irradiance (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 global\_irradiance\_tilted (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 GreeneryCatalogFactory (built-in class), 51  
 ground\_temperature (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 grounds (hub.city\_model\_structure.building :property: Building property), 14  
 grows\_on (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 51  
 grows\_on (hub.city\_model\_structure.greenery.plant :property: Plant property), 38

## H

he (hub.city\_model\_structure.building\_demand.thermal\_boundary :property: ThermalBoundary property), 32  
 he (hub.city\_model\_structure.building\_demand.thermal\_opening :property: ThermalOpening property), 34  
 heat\_losses (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 heating\_consumption (hub.city\_model\_structure.building :property: Building property), 14  
 heating\_demand (hub.city\_model\_structure.building :property: Building property), 14  
 heating\_peak\_load (hub.city\_model\_structure.building :property: Building property), 14  
 heating\_set\_back (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 heating\_set\_back (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 33  
 heating\_set\_point\_schedules (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 heating\_set\_point\_schedules (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 33  
 height (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52  
 height (hub.city\_model\_structure.greenery.plant :property: Plant property), 38  
 hft\_function\_to\_hub\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 hi (hub.city\_model\_structure.building\_demand.thermal\_boundary :property: ThermalBoundary property), 32  
 hi (hub.city\_model\_structure.building\_demand.thermal\_opening :property: ThermalOpening property), 34  
 holes\_polygons (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 hours\_day (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77  
 hours\_day (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 hours\_day (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 Household (built-in class), 24  
 households (hub.city\_model\_structure.building :property: Building property), 14  
 hub\_function\_to\_eilat\_construction\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 hub\_function\_to\_montreal\_custom\_costs\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 hub\_function\_to\_nrcan\_construction\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 hub\_function\_to\_nrel\_construction\_function (hub.helpers.dictionaries :property: Dictionaries property), 89  
 hub\_function\_to\_palma\_construction\_function (hub.helpers.dictionaries :property: Dictionaries property), 89

hub\_usage\_to\_commet\_usage (hub.helpers.dictionaries :property: Dictionaries property), 90  
 hub\_usage\_to\_eilat\_usage (hub.helpers.dictionaries :property: Dictionaries property), 90  
 hub\_usage\_to\_hft\_usage (hub.helpers.dictionaries :property: Dictionaries property), 90  
 hub\_usage\_to\_rncan\_usage (hub.helpers.dictionaries :property: Dictionaries property), 90  
 hub\_usage\_to\_palma\_usage (hub.helpers.dictionaries :property: Dictionaries property), 90  
 hvac\_availability\_schedules (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 hvac\_availability\_schedules (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 33  
 hvac\_subsidy (hub.catalog\_factories.data\_models.cost.income :property: Income property), 65

**I**

id (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 id (hub.catalog\_factories.data\_models.construction.construction :property: Construction property), 58  
 id (hub.catalog\_factories.data\_models.construction.layer :property: Layer property), 59  
 id (hub.catalog\_factories.data\_models.construction.material :property: Material property), 59  
 id (hub.catalog\_factories.data\_models.construction.window :property: Window property), 60  
 id (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 id (hub.catalog\_factories.data\_models.energy\_systems.emission\_system :property: EmissionSystem property), 69  
 id (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 id (hub.catalog\_factories.data\_models.energy\_systems.system :property: System property), 71  
 id (hub.city\_model\_structure.attributes.edge :property: Edge property), 18  
 id (hub.city\_model\_structure.attributes.node :property: Node property), 18  
 id (hub.city\_model\_structure.attributes.schedule :property: Schedule property), 23  
 id (hub.city\_model\_structure.building\_demand.internal\_zone :property: InternalZone property), 26  
 id (hub.city\_model\_structure.building\_demand.layer :property: Layer property), 26  
 id (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 id (hub.city\_model\_structure.building\_demand.thermal\_boundary :property: ThermalBoundary property), 32  
 id (hub.city\_model\_structure.building\_demand.thermal\_opening :property: ThermalOpening property), 34  
 id (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 id (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 id (hub.city\_model\_structure.ios.station :property: Station property), 42  
 id (hub.city\_model\_structure.network :property: Network property), 17  
 inclination (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 Income (built-in class), 65  
 income (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 63  
 indirect\_heated\_ratio (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 indirectly\_heated\_area\_ratio (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 infiltration\_rate\_area\_for\_ventilation\_system\_off (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 infiltration\_rate\_area\_for\_ventilation\_system\_on (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 infiltration\_rate\_area\_system\_off (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 35  
 infiltration\_rate\_area\_system\_on (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 infiltration\_rate\_for\_ventilation\_system\_off (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 57  
 infiltration\_rate\_for\_ventilation\_system\_on (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 58  
 infiltration\_rate\_system\_off (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 infiltration\_rate\_system\_on (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 initial\_investment (hub.catalog\_factories.data\_models.cost.item\_description :property: ItemDescription property), 65  
 initial\_volumetric\_moisture\_content (hub.catalog\_factories.data\_models.greenery.soil :property: Soil property), 53  
 initial\_volumetric\_moisture\_content (hub.city\_model\_structure.greenery.soil :property: Soil property), 39  
 installed\_solar\_collector\_area (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 internal\_gains (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 internal\_gains (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 internal\_surface (hub.city\_model\_structure.building\_demand.thermal\_boundary :property: ThermalBoundary property), 32  
 internal\_walls (hub.city\_model\_structure.building :property: Building property), 14  
 internal\_zones (hub.city\_model\_structure.building :property: Building property), 15  
 InternalGain (built-in class), 25  
 InternalZone (built-in class), 25  
 inverse (hub.city\_model\_structure.attributes.polygon :property: Polygon property), 20  
 inverse (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 is\_conditioned (hub.city\_model\_structure.building :property: Building property), 15  
 item() (hub.catalog\_factories.data\_models.cost.chapter.Chapter method), 64  
 ItemDescription (built-in class), 65  
 items (hub.catalog\_factories.data\_models.cost.chapter :property: Chapter property), 64

**L**

latent\_fraction (hub.catalog\_factories.data\_models.usages.appliances :property: Appliances property), 72  
 latent\_fraction (hub.catalog\_factories.data\_models.usages.lighting :property: Lighting property), 74  
 latent\_fraction (hub.city\_model\_structure.building\_demand.appliances :property: Appliances property), 24  
 latent\_fraction (hub.city\_model\_structure.building\_demand.internal\_gain :property: InternalGain property), 25  
 latent\_fraction (hub.city\_model\_structure.building\_demand.lighting :property: Lighting property), 27  
 latent\_internal\_gain (hub.catalog\_factories.data\_models.usages.occupancy :property: Occupancy property), 74  
 latent\_internal\_gain (hub.city\_model\_structure.building\_demand.occupancy :property: Occupancy property), 28  
 latitude (hub.city\_model\_structure.city :property: City property), 8  
 latitude (hub.city\_model\_structure.ios.sensor\_measure :property: SensorMeasure property), 41  
 Layer (built-in class), 26, 59  
 layers (hub.catalog\_factories.data\_models.construction.construction :property: Construction property), 58  
 layers (hub.city\_model\_structure.building\_demand.thermal\_boundary :property: ThermalBoundary property), 32  
 leaf\_area\_index (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52

leaf\_area\_index (hub.city\_model\_structure.greenery.plant :property: Plant property), 38  
 leaf\_emissivity (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52  
 leaf\_emissivity (hub.city\_model\_structure.greenery.plant :property: Plant property), 38  
 leaf\_reflectivity (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52  
 leaf\_reflectivity (hub.city\_model\_structure.greenery.plant :property: Plant property), 39  
 level\_of\_detail (hub.city\_model\_structure.city :property: City property), 8  
 level\_of\_detail (hub.city\_model\_structure.city\_object :property: CityObject property), 10  
 LevelOfDetail (built-in class), 17  
 lifetime (hub.catalog\_factories.data\_models.cost.item\_description :property: ItemDescription property), 65  
 Lighting (built-in class), 27, 74  
 lighting (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77  
 lighting (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 lighting (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 37  
 lighting\_electrical\_demand (hub.city\_model\_structure.building :property: Building property), 15  
 lighting\_peak\_load (hub.city\_model\_structure.building :property: Building property), 15  
 load() (hub.city\_model\_structure.city.City static method), 8  
 load\_compressed() (hub.city\_model\_structure.city.City static method), 8  
 Location (built-in class), 89  
 location (hub.city\_model\_structure.city :property: City property), 8  
 location (hub.city\_model\_structure.ios.sensor :property: Sensor property), 41  
 lod (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 63  
 long\_wave\_emittance (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 longitude (hub.city\_model\_structure.city :property: City property), 8  
 longitude (hub.city\_model\_structure.ios.sensor\_measure :property: SensorMeasure property), 41  
 lower\_corner (hub.city\_model\_structure.building :property: Building property), 15  
 lower\_corner (hub.city\_model\_structure.building\_demand.surface :property: Surface property), 30  
 lower\_corner (hub.city\_model\_structure.city :property: City property), 8  
 lower\_corner (hub.city\_model\_structure.city\_object :property: CityObject property), 10

**M**

maintenance\_cooling (hub.catalog\_factories.data\_models.cost.operational\_cost :property: OperationalCost property), 66  
 maintenance\_heating (hub.catalog\_factories.data\_models.cost.operational\_cost :property: OperationalCost property), 66  
 maintenance\_pv (hub.catalog\_factories.data\_models.cost.operational\_cost :property: OperationalCost property), 66  
 management (hub.catalog\_factories.data\_models.greenery.vegetation :property: Vegetation property), 54  
 management (hub.city\_model\_structure.greenery.vegetation :property: Vegetation property), 40  
 manufacturer (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 Material (built-in class), 59  
 material (hub.catalog\_factories.data\_models.construction.layer :property: Layer property), 59  
 material\_name (hub.city\_model\_structure.building\_demand.layer :property: Layer property), 27  
 materials (hub.catalog\_factories.data\_models.construction.content :property: Content property), 56  
 max\_coordinate (hub.helpers.configuration\_helper :property: ConfigurationHelper property), 80  
 max\_height (hub.city\_model\_structure.city\_object :property: CityObject property), 11  
 max\_x (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 max\_y (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 max\_z (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 mean\_cooling\_set\_point (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 mean\_cooling\_set\_point (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 33  
 mean\_heating\_set\_point (hub.catalog\_factories.data\_models.usages.thermal\_control :property: ThermalControl property), 76  
 mean\_heating\_set\_point (hub.city\_model\_structure.building\_demand.thermal\_control :property: ThermalControl property), 34  
 mean\_height (hub.city\_model\_structure.building\_demand.internal\_zone :property: InternalZone property), 26  
 measures (hub.city\_model\_structure.ios.sensor :property: Sensor property), 41  
 mechanical\_air\_change (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77  
 mechanical\_air\_change (hub.city\_model\_structure.building\_demand.thermal\_zone :property: ThermalZone property), 36  
 mechanical\_air\_change (hub.city\_model\_structure.building\_demand.usage :property: Usage property), 38  
 merge() (hub.city\_model\_structure.city.City method), 8  
 min\_coordinate (hub.helpers.configuration\_helper :property: ConfigurationHelper property), 80  
 min\_x (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 min\_y (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 min\_z (hub.city\_model\_structure.attributes.polyhedron :property: Polyhedron property), 21  
 minimal\_stomatal\_resistance (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52  
 minimal\_stomatal\_resistance (hub.city\_model\_structure.greenery.plant :property: Plant property), 39  
 mobile (hub.city\_model\_structure.ios.station :property: Station property), 42  
 model\_name (hub.catalog\_factories.data\_models.energy\_systems.distribution\_system :property: DistributionSystem property), 68  
 model\_name (hub.catalog\_factories.data\_models.energy\_systems.emission\_system :property: EmissionSystem property), 69  
 model\_name (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 montreal\_custom\_fuel\_to\_hub\_fuel (hub.helpers.dictionaries :property: Dictionaries property), 90  
 montreal\_demand\_type\_to\_hub\_energy\_demand\_type (hub.helpers.dictionaries :property: Dictionaries property), 90  
 montreal\_function\_to\_hub\_function (hub.helpers.dictionaries :property: Dictionaries property), 90  
 montreal\_generation\_system\_to\_hub\_energy\_generation\_system (hub.helpers.dictionaries :property: Dictionaries property), 90  
 montreal\_system\_to\_hub\_energy\_generation\_system (hub.helpers.dictionaries :property: Dictionaries property), 90  
 municipality (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 63

**N**

name (hub.catalog\_factories.data\_models.construction.archetype :property: Archetype property), 58  
 name (hub.catalog\_factories.data\_models.construction.construction :property: Construction property), 58  
 name (hub.catalog\_factories.data\_models.construction.layer :property: Layer property), 59  
 name (hub.catalog\_factories.data\_models.construction.material :property: Material property), 59  
 name (hub.catalog\_factories.data\_models.construction.window :property: Window property), 60  
 name (hub.catalog\_factories.data\_models.cost.archetype :property: Archetype property), 63  
 name (hub.catalog\_factories.data\_models.energy\_systems.archetype :property: Archetype property), 67  
 name (hub.catalog\_factories.data\_models.energy\_systems.generation\_system :property: GenerationSystem property), 70  
 name (hub.catalog\_factories.data\_models.energy\_systems.system :property: System property), 71  
 name (hub.catalog\_factories.data\_models.greenery.plant :property: Plant property), 52  
 name (hub.catalog\_factories.data\_models.greenery.soil :property: Soil property), 53  
 name (hub.catalog\_factories.data\_models.greenery.vegetation :property: Vegetation property), 54  
 name (hub.catalog\_factories.data\_models.usages.usage :property: Usage property), 77



systems (*hub.catalog\_factories.data\_models.energy\_systems.archetype* :property: Archetype property), 68  
 systems (*hub.catalog\_factories.data\_models.energy\_systems.content* :property: Content property), 67

## T

terrains (*hub.city\_model\_structure.building* :property: Building property), 15  
 thermal\_absorptance (*hub.catalog\_factories.data\_models.construction.material* :property: Material property), 60  
 thermal\_absorptance (*hub.catalog\_factories.data\_models.greenery.soil* :property: Soil property), 53  
 thermal\_absorptance (*hub.city\_model\_structure.building\_demand.layer* :property: Layer property), 27  
 thermal\_absorptance (*hub.city\_model\_structure.greenery.soil* :property: Soil property), 40  
 thermal\_archetype (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalZone property), 26  
 thermal\_boundaries (*hub.city\_model\_structure.building\_demand.storey* :property: Storey property), 29  
 thermal\_boundaries (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 36  
 thermal\_capacity (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype property), 58  
 thermal\_control (*hub.catalog\_factories.data\_models.usages.usage* :property: Usage property), 77  
 thermal\_control (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 36  
 thermal\_control (*hub.city\_model\_structure.building\_demand.usage* :property: Usage property), 38  
 thermal\_openings (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 32  
 thermal\_resistance (*hub.catalog\_factories.data\_models.construction.material* :property: Material property), 60  
 thermal\_resistance (*hub.city\_model\_structure.building\_demand.layer* :property: Layer property), 27  
 thermal\_zone (*hub.city\_model\_structure.building\_demand.storey* :property: Storey property), 29  
 thermal\_zones (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 32  
 thermal\_zones\_from\_internal\_zones (*hub.city\_model\_structure.building* :property: Building property), 16  
 thermal\_zones\_from\_internal\_zones (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalZone property), 26  
 ThermalBoundary (built-in class), 32  
 ThermalControl (built-in class), 33, 76  
 ThermalOpening (built-in class), 34  
 ThermalZone (built-in class), 35  
 thickness (*hub.catalog\_factories.data\_models.construction.layer* :property: Layer property), 59  
 thickness (*hub.city\_model\_structure.building\_demand.layer* :property: Layer property), 27  
 thickness (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 32  
 thickness (*hub.city\_model\_structure.building\_demand.thermal\_opening* :property: ThermalOpening property), 34  
 time (*hub.city\_model\_structure.attributes.record* :property: Record property), 22  
 time\_range (*hub.catalog\_factories.data\_models.usages.schedule* :property: Schedule property), 75  
 time\_range (*hub.city\_model\_structure.attributes.schedule* :property: Schedule property), 23  
 time\_series (*hub.city\_model\_structure.attributes.node* :property: Node property), 18  
 time\_series\_type (*hub.city\_model\_structure.attributes.time\_series* :property: TimeSeries property), 23  
 time\_step (*hub.catalog\_factories.data\_models.usages.schedule* :property: Schedule property), 75  
 time\_step (*hub.city\_model\_structure.attributes.schedule* :property: Schedule property), 23  
 time\_zone (*hub.city\_model\_structure.city* :property: City property), 9  
 TimeSeries (built-in class), 23  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.archetype* :property: Archetype method), 58  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.construction* :property: Construction method), 58  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.content* :property: Content method), 56  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.layer* :property: Layer method), 59  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.material* :property: Material method), 60  
 to\_dictionary() (*hub.catalog\_factories.data\_models.construction.window* :property: Window method), 61  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.archetype* :property: Archetype method), 63  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.capital\_cost* :property: CapitalCost method), 63  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.chapter* :property: Chapter method), 64  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.content* :property: Content method), 62  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.fuel* :property: Fuel method), 64  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.income* :property: Income method), 65  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.item\_description* :property: ItemDescription method), 66  
 to\_dictionary() (*hub.catalog\_factories.data\_models.cost.operational\_cost* :property: OperationalCost method), 66  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.archetype* :property: Archetype method), 68  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.content* :property: Content method), 67  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.distribution\_system* :property: DistributionSystem method), 69  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.emission\_system* :property: EmissionSystem method), 69  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.generation\_system* :property: GenerationSystem method), 70  
 to\_dictionary() (*hub.catalog\_factories.data\_models.energy\_systems.system* :property: System method), 71  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.content* :property: Content method), 51  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.plant* :property: Plant method), 52  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.plant\_percentage* :property: PlantPercentage method), 52  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.soil* :property: Soil method), 53  
 to\_dictionary() (*hub.catalog\_factories.data\_models.greenery.vegetation* :property: Vegetation method), 55  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.appliances* :property: Appliances method), 73  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.content* :property: Content method), 72, 73  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.domestic\_hot\_water* :property: DomesticHotWater method), 74  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.lighting* :property: Lighting method), 74  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.occupancy* :property: Occupancy method), 75  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.schedule* :property: Schedule method), 75  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.thermal\_control* :property: ThermalControl method), 76  
 to\_dictionary() (*hub.catalog\_factories.data\_models.usages.usage* :property: Usage method), 77  
 total\_floor\_area (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 36  
 triangle\_mesh() (*hub.city\_model\_structure.attributes.polygon* :property: Polygon static method), 20  
 triangles (*hub.city\_model\_structure.attributes.polygon* :property: Polygon property), 20  
 trimesh (*hub.city\_model\_structure.attributes.polyhedron* :property: Polyhedron property), 22  
 type (*hub.catalog\_factories.data\_models.construction.construction* :property: Construction property), 58  
 type (*hub.catalog\_factories.data\_models.construction.window* :property: Window property), 61  
 type (*hub.catalog\_factories.data\_models.cost.fuel* :property: Fuel property), 64  
 type (*hub.catalog\_factories.data\_models.cost.item\_description* :property: ItemDescription property), 66  
 type (*hub.catalog\_factories.data\_models.energy\_systems.distribution\_system* :property: DistributionSystem property), 69  
 type (*hub.catalog\_factories.data\_models.energy\_systems.emission\_system* :property: EmissionSystem property), 69  
 type (*hub.catalog\_factories.data\_models.usages.schedule* :property: Schedule property), 76  
 type (*hub.city\_model\_structure.attributes.schedule* :property: Schedule property), 23  
 type (*hub.city\_model\_structure.building\_demand.internal\_gain* :property: InternalGain property), 25  
 type (*hub.city\_model\_structure.building\_demand.surface* :property: Surface property), 31

type (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 32  
 type (*hub.city\_model\_structure.buildings\_cluster* :property: BuildingsCluster property), 16  
 type (*hub.city\_model\_structure.city\_object* :property: CityObject property), 11  
 type (*hub.city\_model\_structure.city\_objects\_cluster* :property: CityObjectsCluster property), 12  
 type (*hub.city\_model\_structure.iot.sensor* :property: Sensor property), 41  
 type (*hub.city\_model\_structure.parts\_consisting\_building* :property: PartsConsistingBuilding property), 16

## U

u\_value (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 33  
 units (*hub.city\_model\_structure.iot.sensor* :property: Sensor property), 41  
 upper\_corner (*hub.city\_model\_structure.building* :property: Building property), 16  
 upper\_corner (*hub.city\_model\_structure.building\_demand.surface* :property: Surface property), 31  
 upper\_corner (*hub.city\_model\_structure.city* :property: City property), 9  
 upper\_corner (*hub.city\_model\_structure.city\_object* :property: CityObject property), 11  
 Usage (built-in class), 37, 77  
 usage (*hub.city\_model\_structure.level\_of\_detail* :property: LevelOfDetail property), 17  
 usage\_name (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 37  
 UsageCatalogFactory (built-in class), 72  
 UsageFactory (built-in class), 46  
 usages (*hub.catalog\_factories.data\_models.usages.content* :property: Content property), 72, 73  
 usages (*hub.city\_model\_structure.building* :property: Building property), 16  
 usages (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalZone property), 26  
 usages (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 37  
 utc\_timestamp (*hub.city\_model\_structure.iot.sensor\_measure* :property: SensorMeasure property), 41

## V

value (*hub.city\_model\_structure.attributes.record* :property: Record property), 22  
 value (*hub.city\_model\_structure.iot.sensor\_measure* :property: SensorMeasure property), 41  
 values (*hub.catalog\_factories.data\_models.usages.schedule* :property: Schedule property), 76  
 values (*hub.city\_model\_structure.attributes.schedule* :property: Schedule property), 23  
 variable (*hub.catalog\_factories.data\_models.cost.fuel* :property: Fuel property), 64  
 Vegetation (built-in class), 40, 54  
 vegetation (*hub.city\_model\_structure.building\_demand.surface* :property: Surface property), 31  
 vegetations (*hub.catalog\_factories.data\_models.greenery.content* :property: Content property), 51  
 ventilation\_rate (*hub.catalog\_factories.data\_models.usages.usage* :property: Usage property), 77  
 vertices (*hub.city\_model\_structure.attributes.polygon* :property: Polygon property), 20  
 vertices (*hub.city\_model\_structure.attributes.polyhedron* :property: Polyhedron property), 22  
 view\_factors\_matrix (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 37  
 virtual\_surfaces (*hub.city\_model\_structure.building\_demand.storey* :property: Storey property), 29  
 visible\_absorptance (*hub.catalog\_factories.data\_models.construction.material* :property: Material property), 60  
 visible\_absorptance (*hub.catalog\_factories.data\_models.greenery.soil* :property: Soil property), 53  
 visible\_absorptance (*hub.city\_model\_structure.building\_demand.layer* :property: Layer property), 27  
 visible\_absorptance (*hub.city\_model\_structure.greenery.soil* :property: Soil property), 40  
 volume (*hub.city\_model\_structure.attributes.polyhedron* :property: Polyhedron property), 22  
 volume (*hub.city\_model\_structure.building\_demand.internal\_zone* :property: InternalZone property), 26  
 volume (*hub.city\_model\_structure.building\_demand.storey* :property: Storey property), 29  
 volume (*hub.city\_model\_structure.building\_demand.thermal\_zone* :property: ThermalZone property), 37  
 volume (*hub.city\_model\_structure.city\_object* :property: CityObject property), 11

## W

walls (*hub.city\_model\_structure.building* :property: Building property), 16  
 weather (*hub.city\_model\_structure.level\_of\_detail* :property: LevelOfDetail property), 17  
 WeatherFactory (built-in class), 46  
 Window (built-in class), 60  
 window (*hub.catalog\_factories.data\_models.construction.construction* :property: Construction property), 58  
 window\_ratio (*hub.catalog\_factories.data\_models.construction.construction* :property: Construction property), 58  
 window\_ratio (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 33  
 windows (*hub.catalog\_factories.data\_models.construction.content* :property: Content property), 56  
 windows\_areas (*hub.city\_model\_structure.building\_demand.thermal\_boundary* :property: ThermalBoundary property), 33

## Y

year\_of\_construction (*hub.city\_model\_structure.building* :property: Building property), 16