CERC hub reference manual

Release 0.0.2

CERC Next-Generation Cities

Feb 22, 2023

CONTENTS:

CHAPTER

ONE

CERC HUB' REFERENCE MANUAL

1.1 Authors

- Guillermo Gutierrez Morote
- Pilar Monsalvete Alvarez de Uribarri
- Sanam Dabirian
- Soroush Samareh Abolhassani

1.2 Contributors

- Seyedehrabeeh Hosseinihaghighi
- Milad Aghamohamadnia
- Peter yefi
- Koa Wells

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 set's of general tools used by any of the other parts and doesn't fit in any of the previous categories.

CHAPTER

TWO

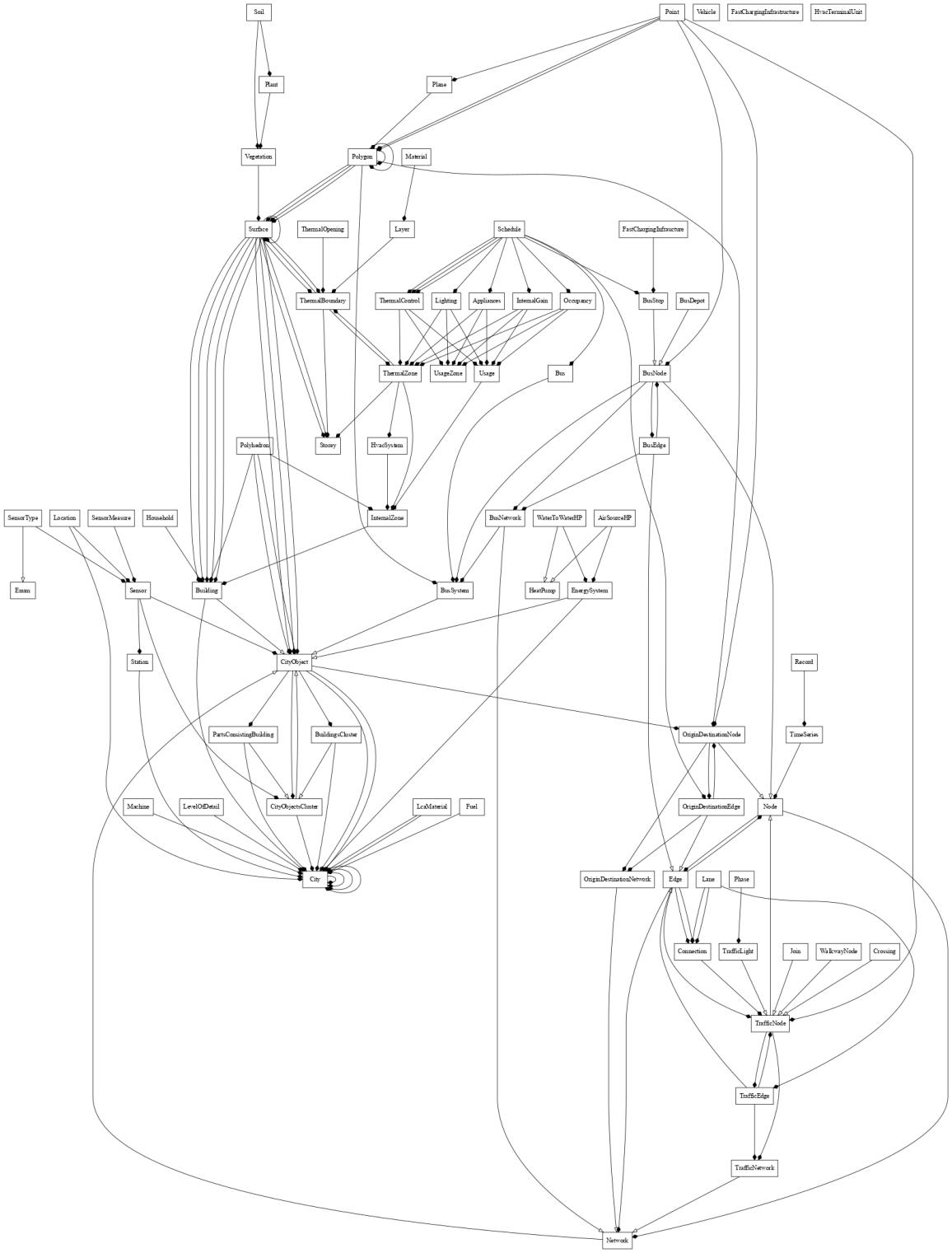
CITY MODEL STRUCTURE

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

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

[how to use the hub]

2.1 City model structure UML



2.2 Folder structure

2.2.1 city_model_structure

Main city objects.

```
../hub/hub/city_model_structure/
├── building.py
├── buildings_cluster.py
├── bus_system.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── energy_system.py
├── fuel.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── __init__.py
├── lca_material.py
├── level_of_detail.py
├── machine.py
├── network.py
├── parts_consisting_building.py
└── vehicle.py

1 directory, 19 files
```

2.2.2 attributes

Geometrical and non physical components of the city.

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

0 directories, 10 files
```

2.2.3 building_demand

Main classes to model building energy demand.

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

0 directories, 17 files
```

2.2.4 energy_systems

Main classes to model energy systems.

```
../hub/hub/city_model_structure/energy_systems/
├── air_source_hp.py
├── heat_pump.py
├── hvac_system.py
├── hvac_terminal_unit.py
├── __init__.py
└── water_to_water_hp.py

0 directories, 6 files
```

2.2.5 iot

Classes to model IoT devices.

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

0 directories, 5 files
```

2.2.6 transport

Classes to model transport.

```
../hub/hub/city_model_structure/transport/
├── bus_depot.py
├── bus_edge.py
├── bus_network.py
├── bus_node.py
├── bus.py
├── bus_stop.py
├── connection.py
├── crossing.py
├── fast_charging_infrastructure.py
├── __init__.py
├── join.py
├── lane.py
├── origin_destination_edge.py
├── origin_destination_network.py
├── origin_destination_node.py
├── phase.py
├── traffic_edge.py
├── traffic_light.py
├── traffic_network.py
├── traffic_node.py
└── walkway_node.py

0 directories, 21 files
```

2.2.7 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
│   ├── household.py
│   ├── __init__.py
│   ├── internal_gain.py
│   ├── internal_zone.py
│   ├── layer.py
│   ├── lighting.py
│   ├── material.py
│   ├── occupancy.py
│   ├── storey.py
│   ├── surface.py
│   ├── thermal_boundary.py
│   ├── thermal_control.py
│   ├── thermal_opening.py
│   ├── thermal_zone.py
│   ├── usage.py
│   └── usage_zone.py
├── energy_systems
│   ├── air_source_hp.py
│   ├── heat_pump.py
│   ├── hvac_system.py
│   ├── hvac_terminal_unit.py
│   ├── __init__.py
│   └── water_to_water_hp.py
├── greenery
│   ├── __init__.py
│   ├── plant.py
│   ├── soil.py
│   └── vegetation.py
├── iot
│   ├── __init__.py
│   ├── sensor_measure.py
│   ├── sensor.py
│   ├── sensor_type.py
│   └── station.py
├── transport
│   ├── bus_depot.py
│   ├── bus_edge.py
│   ├── bus_network.py
│   ├── bus_node.py
│   ├── bus.py
│   ├── bus_stop.py
│   ├── connection.py
│   ├── crossing.py
│   ├── fast_charging_infrastructure.py
│   ├── __init__.py
│   ├── join.py
│   ├── lane.py
│   ├── origin_destination_edge.py
│   ├── origin_destination_network.py
│   ├── origin_destination_node.py
│   ├── phase.py
│   ├── traffic_edge.py
│   ├── traffic_light.py
│   ├── traffic_network.py
│   ├── traffic_node.py
│   └── walkway_node.py
├── building.py
├── buildings_cluster.py
├── bus_system.py
├── city_object.py
├── city_objects_cluster.py
├── city.py
├── energy_system.py
├── fuel.py
├── __init__.py
├── lca_material.py
├── level_of_detail.py
├── machine.py
├── network.py
├── parts_consisting_building.py
└── vehicle.py

6 directories, 78 files
```

2.3 Classes

2.3.1 City

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

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

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*) → Optional[*CityObject*]
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 models country code

Returns str

property energy_systems → Optional[[*EnergySystem*]]
Get energy systems belonging to the city

Returns None or [EnergySystem]

property latitude → Optional[float]
Get city latitude in degrees

Returns None or float

lca_material(*lca_id*) → Optional[LcaMaterial]
Get the lca material matching the given Id

Returns LcaMaterial or None

property lca_materials → Optional[[LcaMaterial]]
Get life cycle materials for the city

Returns [LcaMaterial] or

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

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]

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

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*)
class CityObject

property beam → dict
Get beam radiation surrounding the city object in W/m2

Returns dict{DataFrame(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 W/m2

Returns dict{DataFrame(float)}

property external_temperature → {float}
Get external temperature surrounding the city object in Celsius

Returns dict{DataFrame(float)}

property global_horizontal → dict
Get global horizontal radiation surrounding the city object in W/m2

Returns dict{DataFrame(float)}

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 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*) → Optional[*Surface*]
Get the city object surface with a given name

Parameter name str

Returns None or Surface

surface_by_id(*identification_number*) → Optional[*Surface*]
Get the city object surface with a given name

Parameter identification_number str

Returns None or Surface

property surfaces → [*Surface*]
Get city object surfaces

Returns [Surface]

property type → str
Get city object type

Returns str

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*, *terrains=None*)
Inherit: CityObject
Building(CityObject) class

property alias
Get the alias name for the building

Returns str

property appliances_electrical_demand → dict
Get appliances electrical demand in Wh

Returns dict{DataFrame(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 cooling → dict
Get cooling demand in Wh

Returns dict{DataFrame(float)}

property domestic_hot_water_heat_demand → dict
Get domestic hot water heat demand in Wh

Returns dict{DataFrame(float)}

property eave_height
Get building eave height in meters

Returns float

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 heated_volume
Raises not implemented error

property heating → dict
Get heating demand in Wh

Returns dict{DataFrame(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 Wh

Returns dict{DataFrame(float)}

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 usages_percentage
Get the usages and percentages for the building

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 Bus System

class BusSystem(*name*, *surfaces*)
Inherit: CityObject
BusSystem(CityObject) class

property bus_network → *BusNetwork*
Add explanation here

Returns BusNetwork

property bus_routes → [*BusNode*]
Add explanation here

Returns [BusNode]

property buses → [*Bus*]
Add explanation here

Returns [Bus]

property restricted_polygons → [*Polygon*]
Add explanation here

Returns [Polygon]

2.3.9 Energy System

class EnergySystem(*name*, *surfaces*)
Inherit: CityObject
EnergySystem(CityObject) class

property air_source_hp → *AirSourceHP*
Heat pump energy system

Returns

property type → str
Type of city object

Returns str

property water_to_water_hp → *WaterToWaterHP*
Water to water heat pump energy system

Returns

2.3.10 Fuel

class Fuel(*fuel_id*, *name*, *carbon_emission_factor*, *unit*)

property carbon_emission_factor → float
Get fuel carbon emission factor

Returns float

property id → int
Get fuel id

Returns int

property name → str
Get fuel name

Returns str

property unit → str
Get fuel units

Returns str

2.3.11 Machine

class Machine(*machine_id*, *name*, *work_efficiency*, *work_efficiency_unit*, *energy_consumption_rate*, *energy_consumption_unit*, *carbon_emission_factor*, *carbon_emission_unit*)

Machine class

property carbon_emission_factor → float
Get carbon emission factor

Returns float

property carbon_emission_unit → str
Get carbon emission unit

Returns str

property energy_consumption_rate → float
Get energy consumption rate

Returns float

property energy_consumption_unit → str
Get energy consumption unit

Returns str

property id → int
Get machine id

Returns int

property name → str
Get machine name

Returns str

property work_efficiency → float
Get machine work efficiency

Returns float

property work_efficiency_unit → str
Get machine work efficiency unit

Returns str

2.3.12 Vehicle

class Vehicle(*vehicle_id*, *name*, *fuel_consumption_rate*, *fuel_consumption_unit*, *carbon_emission_factor*, *carbon_emission_factor_unit*)

Vehicle class

property carbon_emission_factor → float
Get vehicle carbon emission factor

Returns float

property carbon_emission_factor_unit → str
Get carbon emission units

Returns str

property fuel_consumption_rate → float
Get vehicle fuel consumption rate

Returns float

property fuel_consumption_unit → str
Get fuel consumption unit

Returns str

property id → int
Get vehicle id

Returns int

property name → str
Get vehicle name

Returns str

2.3.13 Level of detail

class LevelOfDetail
Level of detail for the city class

property construction
Get the city minimal construction 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

2.3.14 Edge

class Edge(*name*, *nodes=None*)
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]

2.3.15 Node

class Node(*name*, *edges=None*)
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

2.3.16 Plane

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

distance(*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

2.3.17 Point

class Point(*coordinates*)
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

2.3.18 Polygon

class Polygon(*coordinates*)
Polygon class

property area
Get surface area in square meters

Returns float

contains_point(*point*)
Determines if the given point is contained by the current polygon

Returns boolean

contains_polygon(*polygon*)
Determines if the given polygon is contained by the current polygon

Returns boolean

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

property vertices → ndarray
Get polygon vertices

Returns np.ndarray(int)

2.3.19 Polyhedron

class Polyhedron(*polygons*)
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

2.3.20 Record

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

2.3.21 Schedule

class Schedule
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]

2.3.22 Time Series

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

property records → [*Record*]
Add explanation here

Returns List[Record]

property time_series_type
Add explanation here

Returns add type of variable here

2.3.23 Appliances

class Appliances
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]

2.3.24 Household

class Household
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

2.3.25 Internal Gain

class InternalGain
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

2.3.26 Internal Zone

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

property area
Get internal zone area in square meters

Returns float

property geometry → *Polyhedron*
Get internal zone geometry

Returns Polyhedron

property hvac_system → Optional[*HvacSystem*]
Get HVAC system installed for this thermal zone

Returns None or HvacSystem

property id
Get internal zone id, a universally unique identifier randomly generated

Returns str

property surfaces
Get internal zone surfaces

Returns [Surface]

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

Returns [ThermalZone]

property usages → [Usage]
Get internal zone usage zones

Returns [UsageZone]

property volume
Get internal zone volume in cubic meters

Returns float

2.3.27 Layer

class Layer
Layer class

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

Returns str

property material → *Material*
Get layer material

Returns Material

property thickness → Optional[float]
Get layer thickness in meters

Returns None or float

2.3.28 Lighting

class Lighting
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]

2.3.29 Material

class Material
Material class

property conductivity → Optional[float]
Get material conductivity in W/mK

Returns None or float

property density → Optional[float]
Get material density in kg/m3

Returns None or float

property id
Get material id

Returns int

property 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 visible_absorptance → Optional[float]
Get material visible absorptance

Returns None or float

2.3.30 Occupancy

class Occupancy
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

2.3.31 Storey

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

2.3.32 Surface

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

Returns float

divide(*z*)
Divides a surface at Z plane

Returns Surface, Surface, Any

property global_irradiance → dict
Get global irradiance on surface in Wh/m2

Returns dict{DataFrame(float)}

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

Returns float

property inverse → *Surface*
Get the inverse surface (the same surface pointing backwards)

Returns Surface

property long_wave_emittance
Get the long wave emittance af the surface

The thermal absorptance can be calculated as 1-long_wave_emittance

Returns float

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

Returns [float]

property name
Get the surface name

Returns str

property 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

shared_surfaces()
Raises not implemented error

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 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

2.3.33 Thermal Boundary

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

property construction_name → Optional[str]
Get construction name

Returns None or str

property he → Optional[float]
Get external convective heat transfer coefficient (W/m2K)

Returns None or float

property hi → Optional[float]
Get internal convective heat transfer coefficient (W/m2K)

Returns None or float

property id
Get thermal zone id, a universally unique identifier randomly generated

Returns str

property internal_surface → *Surface*
Get the internal surface of the thermal boundary

Returns Surface

property layers → [*Layer*]
Get thermal boundary layers

Returns [Layers]

property opaque_area
Get the thermal boundary area in square meters

Returns float

property parent_surface → *Surface*
Get the surface that belongs to the thermal boundary

Returns Surface

property thermal_openings → Optional[[*ThermalOpening*]]
Get thermal boundary thermal openings

Returns None or [ThermalOpening]

property thermal_zones → [*ThermalZone*]
Get the thermal zones delimited by the thermal boundary

Returns [ThermalZone]

property thickness
Get the thermal boundary thickness in meters

Returns float

property type
Get thermal boundary surface type

Returns str

property u_value → Optional[float]
Get thermal boundary U-value in W/m2K

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

Returns None or float

property window_ratio → Optional[float]
Get thermal boundary window ratio

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

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

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

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

Returns float

property windows_areas → [float]
Get windows areas

Returns [float]

2.3.34 Thermal Control

class ThermalControl
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

2.3.35 Thermal Opening

class ThermalOpening
ThermalOpening class

property area → Optional[float]
Get thermal opening area in square meters

Returns None or float

property conductivity → Optional[float]
Get thermal opening conductivity in W/mK

Returns None or float

property construction_name
Get thermal opening construction name

property frame_ratio → Optional[float]
Get thermal opening frame ratio

Returns None or float

property g_value → Optional[float]
Get thermal opening transmittance at normal incidence

Returns None or float

property he → Optional[float]
Get external convective heat transfer coefficient (W/m2K)

Returns None or float

property hi → Optional[float]
Get internal convective heat transfer coefficient (W/m2K)

Returns None or float

property id
Get thermal zone id, a universally unique identifier randomly generated

Returns str

property overall_u_value → Optional[float]
Get thermal opening overall U-value in W/m2K

Returns None or float

property thickness → Optional[float]
Get thermal opening thickness in meters

Returns None or float

2.3.36 Thermal Zone

class ThermalZone(*thermal_boundaries*, *parent_internal_zone*, *volume*, *footprint_area*, *usage_name=None*)
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 effective_thermal_capacity → Optional[float]
Get thermal zone effective thermal capacity in J/m2K

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_system_off
Get thermal zone infiltration rate system off in air changes per hour (ACH)

Returns None or float

property infiltration_rate_system_on
Get thermal zone infiltration rate system on in air changes per hour (ACH)

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 hour (ACH)

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 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

Returns float

property usage_name → Optional[str]
Get thermal zone usage name

Returns None or str

property view_factors_matrix
Get thermal zone view factors matrix

Returns [[float]]

property volume
Get thermal zone volume

Returns float

2.3.37 Usage Zone

class UsageZone
UsageZone 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 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 hour (ACH)

Returns None or float

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 of this thermal zone

Returns None or ThermalControl

property usage → Optional[str]
Get usage zone usage

Returns None or str

2.3.38 Air Source

class AirSourceHP
Inherit: HeatPump
AirSourceHP class

property cooling_capacity → [float]
Get cooling capacity in kW

Returns [[float]]

property cooling_capacity_coff → [float]
Get cooling capacity coefficients

Returns [float]

property cooling_comp_power → [float]
Get cooling compressor power input in kW

Returns [[float]]

property heating_capacity → [float]
Get heating capacity kW

Returns [[float]]

property heating_capacity_coff → [float]
Get heating capacity coefficients

Returns [float]

property heating_comp_power → [float]
Get heating compressor power kW

Returns [[float]]

2.3.39 Heat Pump

class HeatPump
HeatPump class

property hp_monthly_electricity_demand
Electricity demand that results from insel simulation

"**Returns** []

Returns

property hp_monthly_fossil_consumption
Fossil fuel consumption that results from insel simulation

"**Returns** []

Returns

property model → str
Get model name

Returns str

2.3.40 HVAC System

class HvacSystem
HvacSystem class

property thermal_zones → Optional[[*ThermalZone*]]
Get list of zones that this unit serves

Returns None or [ThermalZone]

property type → Optional[str]
Get hvac system type

Returns None or str

2.3.41 HVAC Terminal Unit

class HvacTerminalUnit
HvacTerminalUnit class

property type → Optional[str]
Get type of hvac terminal unit defined for a thermal zone

Returns None or str

2.3.42 Water to Water HP

class WaterToWaterHP
Inherit: HeatPump
WaterToWaterHP class

property entering_water_temp → [float]
Get entering water temperature in degree celsius

Returns [[float]]

property flow_rate → [float]
Get flow rate in kg/s

Returns [[float]]

property leaving_water_temp → [float]
Get leaving water temperature in degree celsius

Returns [[float]]

property power_demand → [float]
Get power demand in kW

Returns [float]

property power_demand_coff → [float]
Get power demand coefficients

Returns [float]

property total_cooling_capacity → [float]
Get total cooling capacity

Returns [float]

2.3.43 Plant

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

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

2.3.44 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*)

property dry_conductivity
Get soil dry conductivity in W/mK

Returns float

property dry_density
Get soil dry density in kg/m3

Returns float

property dry_specific_heat
Get soil dry specific heat in J/kgK

Returns float

property initial_volumetric_moisture_content
Get soil initial volumetric moisture content

Returns None or float

property name
Get soil name

Returns string

property residual_volumetric_moisture_content
Get soil residual volumetric moisture content

Returns None or float

property roughness
Get soil roughness

Returns string

property saturation_volumetric_moisture_content
Get soil saturation volumetric moisture content

Returns float

property solar_absorptance
Get soil solar absortance

Returns float

property thermal_absorptance
Get soil thermal absortance

Returns float

property visible_absorptance
Get soil visible absortance

Returns float

2.3.45 Vegetation

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

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

2.3.46 Sensor

class Sensor
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

2.3.47 Sensor Measure

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

property latitude
Get measure latitude

property longitude
Get measure longitude

property utc_timestamp
Get measure timestamp in utc

property value
Get sensor measure value

2.3.48 Sensor Type

class SensorType(*value*)
Inherit: Enum
An enumeration.

2.3.49 Station

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

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]

2.3.50 Connection

class Connection
Connection class

property from_edge → *Edge*
Get "from" edge

Returns Edge

property from_lane → *Lane*
Get "from" lane

Returns Lane

property keep_clear → Optional[bool]
Get if vehicles which pass this (lane to lane) connection should keep the intersection clear

Returns None or Boolean

property pass_not_wait → Optional[bool]
Get if the vehicles which pass this (lane to lane) connection will not wait

Returns None or Boolean

property to_edge → *Edge*
Get "to" edge

Returns Edge

property to_lane → *Lane*
Get "to" lane

Returns Lane

2.3.51 Crossing

class Crossing(*name*, *coordinates*, *priority*, *width*, *shape=None*, *edges=None*)
Inherit: TrafficNode

Crossing class

property priority → Optional[bool]
Get whether the pedestrians have priority over the vehicles

Returns None or bool

property shape → Optional[[[float]]]
Get the list of positions

Returns None or [[x, y, (z)]]

property width → Optional[float]
Get crossing width in meters

Returns None or float

2.3.52 Join

class Join(*name*, *coordinates*, *nodes*)
Inherit: TrafficNode

Join class

2.3.53 Lane

class Lane

Lane class

property allow → Optional[[str]]
Get the list of allowed vehicle classes

Returns None or [str]

property change_left → Optional[[str]]
Get the list of vehicle classes that may change left from this lane

Returns None or [str]

property change_right → Optional[[str]]
Get the list of vehicle classes that may change right from this lane

Returns None or [str]

property disallow → Optional[[str]]
Get the list of not allowed vehicle classes

Returns None or [str]

property index → Optional[int]
Get lane index

The enumeration index of the lane (0 is the rightmost lane, <NUMBER_LANES>-1 is the leftmost one)

Returns None or int

property speed → Optional[float]
Get the lane speed in m/s

Returns None or float

property width → Optional[float]
Get the lane width in meters

Returns None or float

2.3.54 Phase

class Phase
Phase class

property duration → Optional[int]
Get phase duration in seconds

Returns None or int

property max_duration → Optional[int]
Get phase maximum duration in seconds

Returns None or int

property min_duration → Optional[int]
Get phase minimum duration in seconds

Returns None or int

property name → Optional[str]
Get phase name

Returns None or str

property next → Optional[[int]]
Get the next phase in the cycle after the current.

This is useful when adding extra transition phases to a traffic light plan which are not part of every cycle.

Traffic lights of type 'actuated' can make use of a list of indices for selecting among alternative

successor phases.

Returns None or [int]

property state → Optional[[str]]
Get the list of signal states

Returns None or [str]

2.3.55 Traffic Edge

class TrafficEdge(*name*, *nodes*, *priority*, *speed*, *lanes*, *length*, *allows=None*, *disallows=None*, *sidewalk_width=None*, *edge_type='TrafficEdge'*)
Inherit: Edge
TrafficEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

property allows → Optional[[str]]
Get the list of allowed vehicle classes

Returns None or [str]

property disallows → Optional[[str]]
Get the list of not allowed vehicle classes

Returns None or [str]

property edge_type
Get the edge type

Returns str

property lanes → [*Lane*]
Get the lanes on an edge

Returns List[Lane]

property length → Optional[float]
Get the lane length in meters

Returns None or float

property nodes → [*TrafficNode*]
Get delimiting nodes for the edge

Returns [TrafficNode]

property priority → Optional[int]
Get the priority between different road types.

It starts with one; higher numbers represent more important roads.

Returns None or int

property speed → Optional[float]
Get he speed limit in m/s

Returns None or float

2.3.56 Traffic Light

class TrafficLight(*name*, *coordinates*, *offset*, *phases=None*, *edges=None*, *right_on_red=False*)
Inherit: TrafficNode
Traffic light class

property offset → Optional[int]
Get program initial time offset

Returns None or int

property phases → [*Phase*]
Get traffic light logic phases

Returns [Phase]

property right_on_red → Optional[bool]
Get if is possible to turn right when the traffic light is red

Returns None or Boolean

2.3.57 Traffic Network

class TrafficNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network

TrafficNetwork(Network) class

property edges → [*TrafficEdge*]
Get network edges

Returns [TrafficEdge]

property nodes → [*TrafficNode*]
Get network nodes

Returns [TrafficNode]

property type
Get network type

Returns str

2.3.58 Traffic Node

class TrafficNode(*name*, *coordinates*, *node_type='TrafficNode'*, *edges=None*, *prohibitions=None*, *connections=None*)
Inherit: Node

TrafficNode class

property connections → [*Connection*]
Get node connections

Returns [Connection]

property coordinates → *Point*
Get node coordinates

Returns Point

property edges → [*TrafficEdge*]
get edges delimited by the node

Returns [TrafficEdge]

property node_type
Get node type

Returns str

property prohibitions → [*Edge*]
Get node prohibitions

Returns [(Edge, Edge)]

2.3.59 Walkway Node

class WalkwayNode(*name*, *coordinates*, *edges=None*, *shape=None*)
Inherit: TrafficNode
WalkwayNode class

property shape → Optional[[[float]]]
Get the list of positions

Returns None or [[x, y, (z)]]

2.3.60 Bus

class Bus
Bus class

property capacity
Add explanation here

Returns add type of variable here

property charging_range
Add explanation here

Returns add type of variable here

property charging_time
Add explanation here

Returns add type of variable here

property energy_consumption
Add explanation here

Returns add type of variable here

property investment_cost
Add explanation here

Returns add type of variable here

property maintenance_cost
Add explanation here

Returns add type of variable here

property maintenance_time
Add explanation here

Returns add type of variable here

property maximum_travel_range
Add explanation here

Returns add type of variable here

property recovery_time
Add explanation here

Returns add type of variable here

property trips_schedule → *Schedule*
Add explanation here

Returns add type of variable here

property vehicle_type
Add explanation here

Returns add type of variable here

2.3.61 Bus Depot

class BusDepot(*name*, *coordinates*, *edges=None*)
Inherit: BusNode
BusDepot class

property number_of_available_buses
Add explanation here

Returns add type of variable here

property number_of_charging_poles
Add explanation here

Returns add type of variable here

2.3.62 Bus Edge

class BusEdge(*name*, *nodes*, *edge_type='BusEdge'*)
Inherit: Edge
BusEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

property average_travel_time
Add explanation here

Returns add type of variable here

property edge_type
Get the edge type

Returns str

property nodes → [*BusNode*]
Get delimiting nodes for the edge

Returns [BusNode]

2.3.63 Bus Network

class BusNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network
BusNetwork(Network) class

property edges → [*BusEdge*]
Get network edges

Returns [BusEdge]

property nodes → [*BusNode*]
Get network nodes

Returns [BusNode]

property type
Get network type

Returns str

2.3.64 Bus Node

class BusNode(*name*, *coordinates*, *node_type='BusNode'*, *edges=None*)
Inherit: Node
BusNode class

property coordinates → *Point*
Get node coordinates

Returns Point

property edges → [*BusEdge*]
get edges delimited by the node

Returns [BusEdge]

property node_type
Get node type

Returns str

2.3.65 Bus Stop

class BusStop(*name*, *coordinates*, *edges=None*)
Inherit: BusNode
BusStop class

property average_hourly_passengers_demand → *Schedule*
Add explanation here

Returns Schedule

property fast_charging_infrastructure → Optional[*FastChargingInfrastructure*]
Add explanation here

Returns FastChargingInfrastructure

property time_table
Add explanation here

Returns add type of variable here

property waiting_time
Add explanation here

Returns add type of variable here

2.3.66 Fast Charging Infrastructure

class FastChargingInfrastructure
FastChargingInfrastructure class

property electrical_demand
Add explanation here

Returns add type of variable here

property losses_in_grid
Add explanation here

Returns add type of variable here

2.3.67 Origin Destination Network

class OriginDestinationNetwork(*name*, *edges=None*, *nodes=None*)
Inherit: Network
OriginDestinationNetwork(Network) class

property edges → [*OriginDestinationEdge*]
Get network edges

Returns [OriginDestinationEdge]

property nodes → [*OriginDestinationNode*]
Get network nodes

Returns [OriginDestinationNode]

property type
Get network type

Returns str

2.3.68 Origin Destination Edge

class OriginDestinationEdge(*name*, *nodes*, *edge_type='OriginDestinationEdge'*)
Inherit: Edge
OriginDestinationEdge class Each edge is unidirectional and starts at the "from" node and ends at the "to" node

property edge_type
Get the edge type

Returns str

property movement_schedule → *Schedule*
Get the schedule of the movement of people along this edge

Returns Schedule

property nodes → [*OriginDestinationNode*]
Get delimiting nodes for the edge

Returns [OriginDestinationNode]

2.3.69 Origin Destination Node

class OriginDestinationNode(*name*, *coordinates*, *node_type='OriginDestinationNode'*, *edges=None*, *polygon=None*)

Inherit: Node

OriginDestinationNode class

property city_objects → [*CityObject*]

Get the list of city objects place inside the zone

Returns List[CityObject]

property coordinates → *Point*

Get node coordinates

Returns Point

property edges → [*OriginDestinationEdge*]

get edges delimited by the node

Returns [OriginDestinationEdge]

property land_use_types → dict

Get land use types inside the node polygon. It returns a dictionary with the types of land use together with the

percentage of the land that corresponds to each type

Returns {string : float}

property node_type

Get node type

Returns str

property polygon → *Polygon*

Get node polygon that defines the zone represented by the node

Returns Polygon

CHAPTER

THREE

FACTORIES

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

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

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

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

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

[development manual]

3.1 Folder structure

3.1.1 Imports

../hub/imports/ [error opening dir]

0 directories, 0 files

3.1.2 Exports

../hub/exports/ [error opening dir]

0 directories, 0 files

3.2 Imports Classes

3.2.1 Construction Factory

class ConstructionFactory(*handler*, *city*, *base_path=None*)

ConstructionFactory class

_nrcan()

Enrich the city by using NRCAN information

_nrel()

Enrich the city by using NREL information

enrich()

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

Returns None

enrich_debug()

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

Returns None

3.2.2 Geometry Factory

class GeometryFactory(*file_type*, *path=None*, *data_frame=None*, *height_field=None*, *year_of_construction_field=None*, *function_field=None*, *function_to_hub=None*)

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 _gpandas → *City*

Enrich the city by using GeoPandas information as data source

Returns City

property _obj → *City*

Enrich the city by using OBJ information as data source

Returns City

property _rhino → *City*

Enrich the city by using Rhino information as data source

Returns City

property city → *City*

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

Returns City

property city_debug → *City*
Enrich the city given to the class using the class given handler

Returns City

3.2.3 Sensors Factory

class SensorsFactory(*handler*, *city*, *end_point*, *base_path=None*)
UsageFactory class

_cec()
Enrich the city by using concordia energy consumption sensors as data source

_cgf()
Enrich the city by using concordia gas flow sensors as data source

_ct()
Enrich the city by using concordia temperature sensors as data source

enrich()
Enrich the city given to the class using the given sensor handler

Returns None

3.2.4 Usage Factory

class UsageFactory(*handler*, *city*, *base_path=None*)
UsageFactory class

_comnet()
Enrich the city with COMNET usage library

_nrcan()
Enrich the city with NRCAN usage library

enrich()
Enrich the city given to the class using the usage factory given handler

Returns None

3.2.5 Weather Factory

class WeatherFactory(*handler*, *city*, *base_path=None*, *file_name=None*)
WeatherFactory class

_epw()
Enrich the city with energy plus weather file

_xls()
Enrich the city with ISO_52016_1_BESTEST_ClimData_2016.08.24 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(*export_type*, *city*, *path*, *target_buildings=None*, *adjacent_buildings=None*, *weather_file=None*, *weather_format=None*)

Exports factory class

property _citygml

Export to citygml

Returns None

property _grounded_obj

Export the city geometry to obj with grounded coordinates

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

export_debug()

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

Returns None

CHAPTER

FOUR

CATALOGS

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

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

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

[catalogs manual]

4.1 Folder structure

4.1.1 Catalogs

4.2 Catalog Classes

4.2.1 Catalog

`class Catalog`

Catalogs base class not implemented instance of the Catalog 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

4.2.2 Greenery Catalog Factory

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

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

4.2.3 Construction Catalog Factory

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

property _nrcan
Retrieve NREL catalog

property _nrel
Retrieve NREL catalog

property catalog → *Catalog*
Enrich the city given to the class using the class given handler

Returns Catalog

CHAPTER

FIVE

HELPERS

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

clearpage

5.1 Folder structure

```
../hub/hub/helpers/
├── data
│   ├── alkis_function_to_hub_function.py
│   ├── hft_function_to_hub_function.py
│   ├── hub_function_to_nrcan_construction_function.py
│   ├── hub_function_to_nrel_construction_function.py
│   ├── hub_usage_to_comnet_usage.py
│   ├── hub_usage_to_hft_usage.py
│   ├── hub_usage_to_nrcan_usage.py
│   ├── montreal_function_to_hub_function.py
│   └── pluto_function_to_hub_function.py
├── auth.py
├── configuration_helper.py
├── constants.py
├── dictionaries.py
├── enrich_city.py
├── geometry_helper.py
├── __init__.py
├── location.py
├── monthly_values.py
├── utils.py
└── yearly_from_daily_schedules.py

1 directory, 20 files
```

5.2 Configuration Helper

class ConfigurationHelper

Configuration class

property comnet_lighting_convective → float

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

Returns 0.5

property comnet_lighting_latent → float

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

Returns 0

property comnet_lighting_radiant → float

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

Returns 0.5

property comnet_occupancy_sensible_convective → float

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

used for Comnet (ASHRAE) standard

Returns 0.9

property comnet_occupancy_sensible_radiant → float

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

used for Comnet (ASHRAE) standard

Returns 0.1

property comnet_plugs_convective → float

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

Returns 0.75

property comnet_plugs_latent → float

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

Returns 0

property comnet_plugs_radiant → float

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

Returns 0.25

property convective_heat_transfer_coefficient_exterior → float

Get configured convective heat transfer coefficient for surfaces outside the building

Returns 20 W/m2K

property convective_heat_transfer_coefficient_interior → float

Get configured convective heat transfer coefficient for surfaces inside the building

Returns 3.5 W/m2K

property max_coordinate → float

Get configured maximal coordinate value

Returns 1.7976931348623157e+308

property max_location_distance_for_shared_walls → float
Get configured maximal distance between attributes to consider that they may share walls in meters

Returns 5.0

property min_coordinate → float
Get configured minimal coordinate value

Returns -1.7976931348623157e+308

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

5.3 Constants

KELVIN = 273.15
HOUR_TO_MINUTES = 60
MINUTES_TO_SECONDS = 60
METERS_TO_FEET = 3.28084
BTU_H_TO_WATTS = 0.29307107
KILO_WATTS_HOUR_TO_JULES = 3600000
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'
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'
LIGHTING = 'Lights'
OCCUPANCY = 'Occupancy'
APPLIANCES = 'Appliances'
HVAC_AVAILABILITY = 'HVAC Avail'
INFILTRATION = 'Infiltration'
COOLING_SET_POINT = 'ClgSetPt'
HEATING_SET_POINT = 'HtgSetPt'
EQUIPMENT = 'Equipment'
ACTIVITY = 'Activity'
PEOPLE_ACTIVITY_LEVEL = 'People Activity Level'
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'

5.4 Geometry Helper

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

static adjacent_locations(*location1*, *location2*)
Determine when two attributes may be adjacent or not based in the dis

Parameter location1

Parameter location2

Returns Boolean

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 get_location(*latitude*, *longitude*) → *Location*
Get Location from latitude and longitude

static segment_list_to_trimesh(*lines*) → Trimesh
Transform a list of segments into a Trimesh

5.5 Location

class Location(*country*, *city*)

property city
City name

property country
Country code

5.6 Monthly to Hourly Demand

CHAPTER

SIX

ADDITIONAL FILES

6.1 Readme

README.md

6.2 License

LICENSE.md

6.3 Code of conduct

CODE_OF_CONDUCT.md

6.4 How to contribute

CONTRIBUTING.md

6.5 Coding style

PYGUIDE.md

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Y