eco-connect Documentation

ecorithm

Oct 10, 2019

CONTENTS

Welcome to documentation for eco-connect, a thin python wrapper for Ecorithm's API Platform.

CHAPTER

ONE

REQUIREMENTS

- python >= 3.6

CHAPTER

TWO

INSTALLATION

eco-connect can be installed via pip from PyPI.

```
pip install eco-connect
```

This will likely require the installation of a number of dependencies, including Pandas, will require a compiler to compile required bits of code, and can take a few minutes to complete.

All supported connectors require authentication via your provided ecorithm username and password. These credentials are set through environment variables and can be set by running the following commands in shell:

```
export ECO_CONNECT_USER="MY_ECORITHM_USERNAME"
export ECO_CONNECT_PASSWORD="MY_ECORITHM_PASSWORD"
```

To verify your credentials and that the package has been installed correctly, run the following in a pythoninterpreter of your choice:

```
from eco_connect import validate_credentials

validate_credentials()
```

If everything was setup properly, you should see the print out:

Ecorithm credentials successfully validated!

CHAPTER

THREE

USAGE

All connectors can be imported directly from eco_connect. For example, to import the FactsService connector:

```
from eco_connect import FactsService
```

CHAPTER

FOUR

SUPPORTED CONNECTORS

4.1 Facts-Service

This connector is used to access the facts-service api via python. For direct api documentation, please refer to https://facts.prod.ecorithm.com/api/v1/doc.

4.1.1 Example Usage

```
from eco_connect import FactsService

facts_service = FactsService()

data = facts_service.get_facts(building_id=26,
                               start_date='2017-12-20 00:00',
                               end_date='2017-12-21 00:00',
                               result_format='pandas')
```

4.1.2 FactsService

class eco_connect.**FactsService**(*environment_name='prod'*, *version='v1'*)

A class to connect to Ecorithm's facts-service API (https://facts.prod.ecorithm.com/api/v1/).

Args:

Kwargs: **environment_name** (str): Api environment. Supported environments ('Prod', 'Qa')

Example: 'prod'

version (str): Api version. Supported versions: ('v1')

Example: 'v1'

get_facts

FactsService.**get_facts**(*building_id*, *start_date*, *end_date*, *start_hour='00:00'*, *end_hour='23:55'*, *equipment_names=[]*, *equipment_types=[]*, *excluded_days=[]*, *excluded_dates=[]*, *point_classes=[]*, *eco_point_ids=[]*, *display_names=[]*, *native_names=[]*, *point_class_expression=[]*, *native_name_expression=[]*, *display_name_expression=[]*, *result_format='pandas'*)

Return the sensor facts for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Facts/Facts_get

Args:

building_id (str): Building id to get facts for.

Example: 1

start_date (str): Start date to return facts for.

Example: '2017-12-20 00:00'

end_date (str): End date to return facts for.

Example: '2017-12-21 23:55'

Kwargs:

start_hour (str): Start hour to filter facts for.

Example: '08:00'

end_hour (str): End hour to filter facts for.

Example: '17:00'

excluded_days (list): Specific days to filter data for. Monday=1, Sunday=7

Example: [6, 7] (Filters Saturday / Sunday)

excluded_dates (list): Specific dates to filter data for.

Example: ['2017-12-20', '2017-12-25']

equipment_names (list): List of equipment names to filter facts for.

Example: ['VAV_01', 'VAV_02']

equipment_types (list): List of equipment types to filter facts for.

Example: ['VAV', 'AHU']

point_classes (list): List of point_classes to filter facts for.

Example: ['SpaceAirTemperature', 'CoolingCoilUnitFeedback']

eco_point_ids (list): List of eco_point_ids to filter facts for.

Example: [1, 2, 3]

display_names (list): List of display_names to filter facts for.

Example: ['SpaceTemp', 'AirFlow']

native_names (list): List of native_names to filter facts for.

Example: ['name-1', 'name-2']

point_class_expression (list): List of equipment / equipment type + point class regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited. for.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

native_name_expression (list): List of native-name regex expressions to filter facts.

Example: ['nam.*', 'nati-.*']

display_name_expression (list): List of equipment / equipment type + display name regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

result_format (str): Output format type. (Pandas, tuple, csv, json)

Example: 'pandas'

Returns: (DataFrame or list or csv or json depending on the requested result format).

DataFrame Example:

```
index     fact_time       fact_value  eco_point_id  display_name   native_
↪name          point_class         equipment_name  equipment_type
=====  ================   ==========  ============  ============  ␣
↪===========  ==========================  ==============  ===============
0      2017-12-20 00:00       1           192       'SpaceTemp'
↪'name-1'    'SpaceAirTemperature'         'VAV-01'        'VAV'
1      2017-12-21 00:00       2           304       'CoolingCoil'
↪'name-2'    'CoolingCoilUnitFeedback'     'AHU-01'        'AHU'
```

Json Example:

```
{
  "data": [
    {
      "87238": {
        "data": {
          "2017-08-01 00:00": 67.5,
          "2017-08-01 00:05": 68.5
        },
        "meta": {
          "display_name": "SpaceTemp",
          "eco_point_id": 85743,
          "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T",
          "equipment": "VAV-301",
          "equipment_type": "VAV",
          "point_class": "SpaceAirTemperature"
        }
      },
      "87239": {
        "data": {
          "2017-08-01 00:00": 0,
          "2017-08-01 00:05": 100
        },
        "meta": {
          "display_name": "Cooling",
          "eco_point_id": 87239,
          "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.CV",
          "equipment": "VAV_301",
          "equipment_type": "VAV",
          "point_class": "CoolingCoilUnitFeedback"
        }
      }
    }
  ]
}
```

Csv Example:

```
',fact_time,fact_value,point_class,display_name,native_name,
equipment_type,equipment_name,eco_point_id


0,2017-12-20 00:05,70.1923,SpaceAirTemperature,SpaceTemp,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T,VAV,VAV_301,85743,


2017-12-20 00:05,76.08,SpaceAirTemperatureSetPointWhenCooling,
SpaceTempSetPoint_ActualCooling,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET,VAV,VAV_301,
85744,

2017-12-20 00:05,70.08,
SpaceAirTemperatureSetPointWhenHeating,
SpaceTempSetPoint_ActualHeating,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.AHTG-SET,VAV,VAV_301,85745'
```

Tuple Example:

```
[response_tuple(fact_time='2017-12-20 00:05', fact_value=70.1923,
                point_class='SpaceAirTemperature', display_name='SpaceTemp
↪',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T
↪',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85743),
 response_tuple(fact_time='2017-12-20 00:05', fact_value=76.08,
                point_class='SpaceAirTemperatureSetPointWhenCooling',
                display_name='SpaceTempSetPoint_ActualCooling',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-
↪SET',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85744)]
```

Note: And invalid api request will return back the raw api response.

Example: {"message": {"NoData": "No data found for the provided filters. This building has data from '2017-12-20 00:00' to '2017-12-21 00:00'"}}

Example Usage:

```
>>> from eco_connect import FactsService
>>> facts_service = FactsService()
>>> facts_service.get_facts(building_id=26,
                            start_date='2017-12-20 00:00',
                            end_date='2017-12-21 00:00',
                            result_format='json')
    {
      "data": [
        {
          "87238": {
            "data": {
              "2017-08-01 00:00": 67.5,
```

(continues on next page)

(continued from previous page)


```
            "2017-08-01 00:05": 68.5
          },
          "meta": {
            "display_name": "SpaceTemp",
            "eco_point_id": 85743,
            "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T",
            "equipment": "VAV-301",
            "equipment_type": "VAV",
            "point_class": "SpaceAirTemperature"
          }
        },
        "87239": {
          "data": {
            "2017-08-01 00:00": 0,
            "2017-08-01 00:05": 100
          },
          "meta": {
            "display_name": "Cooling",
            "eco_point_id": 87239,
            "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.CV",
            "equipment": "VAV_301",
            "equipment_type": "VAV",
            "point_class": "CoolingCoilUnitFeedback"
          }
        }
      }
    ]
  }
```


get_avg_facts

`FactsService.`**`get_avg_facts`**(*building_id*, *start_date*, *end_date*, *start_hour='00:00'*, *end_hour='23:55'*, *period='day'*, *equipment_names=[]*, *equipment_types=[]*, *excluded_days=[]*, *excluded_dates=[]*, *point_classes=[]*, *eco_point_ids=[]*, *display_names=[]*, *native_names=[]*, *point_class_expression=[]*, *native_name_expression=[]*, *display_name_expression=[]*, *result_format='pandas'*)

Return the average sensor facts for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Facts/get_avg

Args:

building_id (str): Building id to get facts for.

Example: 1

start_date (str): Start date to return facts for.

Example: '2017-12-20 00:00'

end_date (str): End date to return facts for.

Example: '2017-12-21 23:55'

Kwargs:

start_hour (str): Start hour to filter facts for.

Example: '08:00'

end_hour (str): End hour to filter facts for.

Example: '17:00'

excluded_days (list): Specific days to filter data for. Monday=1, Sunday=7

Example: [6, 7] (Filters Saturday / Sunday)

excluded_dates (list): Specific dates to filter data for.

Example: ['2017-12-20', '2017-12-25']

equipment_names (list): List of equipment names to filter facts for.

Example: ['VAV_01', 'VAV_02']

equipment_types (list): List of equipment types to filter facts for.

Example: ['VAV', 'AHU']

point_classes (list): List of point_classes to filter facts for.

Example: ['SpaceAirTemperature', 'CoolingCoilUnitFeedback']

eco_point_ids (list): List of eco_point_ids to filter facts for.

Example: [1, 2, 3]

display_names (list): List of display_names to filter facts for.

Example: ['SpaceTemp', 'AirFlow']

native_names (list): List of native_names to filter facts for.

Example: ['name-1', 'name-2']

point_class_expression (list): List of equipment / equipment type + point class regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited. for.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

native_name_expression (list): List of native-name regex expressions to filter facts.

Example: ['nam.*', 'nati-.*']

display_name_expression (list): List of equipment / equipment type + display name regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

period (string): Aggregate period to average on. Supports the following aggregates [minute, hour, day, week, month, year]

Example: 'hour'

Default: 'day'

result_format (str): Output format type. (Pandas, tuple, csv, json)

Example: 'pandas'

Returns: (DataFrame or list or csv or json depending on the requested result format).

DataFrame Example:

```
index     fact_time       fact_value  eco_point_id  display_name   native_
↪name          point_class         equipment_name  equipment_type
=====  ================   ==========  ============  ============  ⌴
↪===========  =========================  ==============  ==============
0      2017-12-20 00:00       1            192       'SpaceTemp'
↪'name-1'    'SpaceAirTemperature'          'VAV-01'        'VAV'
1      2017-12-21 00:00       2            304       'CoolingCoil'
↪'name-2'    'CoolingCoilUnitFeedback'     'AHU-01'        'AHU'
```

Json Example:

```
{
  "data": [
    {
      "87238": {
        "data": {
          "2017-08-01 00:00": 67.5,
          "2017-08-01 00:05": 68.5
        },
        "meta": {
          "display_name": "SpaceTemp",
          "eco_point_id": 85743,
          "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T",
          "equipment": "VAV-301",
          "equipment_type": "VAV",
          "point_class": "SpaceAirTemperature"
        }
      },
      "87239": {
        "data": {
          "2017-08-01 00:00": 0,
          "2017-08-01 00:05": 100
        },
        "meta": {
          "display_name": "Cooling",
          "eco_point_id": 87239,
          "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.CV",
          "equipment": "VAV_301",
          "equipment_type": "VAV",
          "point_class": "CoolingCoilUnitFeedback"
        }
      }
    }
  ]
}
```

Csv Example:

```
',fact_time,fact_value,point_class,display_name,native_name,
equipment_type,equipment_name,eco_point_id


0,2017-12-20 00:05,70.1923,SpaceAirTemperature,SpaceTemp,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T,VAV,VAV_301,85743,


2017-12-20 00:05,76.08,SpaceAirTemperatureSetPointWhenCooling,
```

(continues on next page)

(continued from previous page)

```
SpaceTempSetPoint_ActualCooling,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET,VAV,VAV_301,
85744,

2017-12-20 00:05,70.08,
SpaceAirTemperatureSetPointWhenHeating,
SpaceTempSetPoint_ActualHeating,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.AHTG-SET,VAV,VAV_301,85745'
```

Tuple Example:

```
[response_tuple(fact_time='2017-12-20 00:05', fact_value=70.1923,
                point_class='SpaceAirTemperature', display_name='SpaceTemp
↪',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T
↪',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85743),
 response_tuple(fact_time='2017-12-20 00:05', fact_value=76.08,
                point_class='SpaceAirTemperatureSetPointWhenCooling',
                display_name='SpaceTempSetPoint_ActualCooling',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-
↪SET',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85744)]
```

Note: And invalid api request will return back the raw api response.

Example: {"message": {"NoData": "No data found for the provided filters. This building has data from '2017-12-20 00:00' to '2017-12-21 00:00'"}}

Example Usage:

```
>>> from eco_connect import FactsService
>>> facts_service = FactsService()
>>> facts_service.get_avg_facts(building_id=26,
                                start_date='2017-12-20 00:00',
                                end_date='2017-12-21 00:00',
                                period='minute',
                                result_format='json')
```


```
{
  "data": [
    {
      "87238": {
        "data": {
          "2017-08-01 00:00": 67.5,
          "2017-08-01 00:05": 68.5
        },
        "meta": {
          "display_name": "SpaceTemp",
          "eco_point_id": 85743,
          "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T",
          "equipment": "VAV-301",
          "equipment_type": "VAV",
          "point_class": "SpaceAirTemperature"
```


(continues on next page)

(continued from previous page)

```
            }
          },
          "87239": {
            "data": {
              "2017-08-01 00:00": 0,
              "2017-08-01 00:05": 100
            },
            "meta": {
              "display_name": "Cooling",
              "eco_point_id": 87239,
              "native_name": "UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.CV",
              "equipment": "VAV_301",
              "equipment_type": "VAV",
              "point_class": "CoolingCoilUnitFeedback"
            }
          }
        }
      ]
    }
```

put_facts

`FactsService.`**`put_facts`**(*building_id*, *data=Empty DataFrame Columns: [fact_time, fact_value, native_name] Index: []*)

Insert facts for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Facts/put

Args:

building_id (str): Building id to get facts for.

Example: 1

data (DataFrame): DataFrame with data to upload.

Example:

index	fact_time	fact_value	native-name
0	2017-12-20 00:00	1	native-name-1
1	2017-12-21 00:00	2	native-name-2

Note: If errors are present, a *message* key will be returned.

Example: { 'message': {'field_errors': ['native-name-1']} }

Example Usage:

```
>>> from eco_connect import FactsService
>>> import pandas as pd
>>> facts_service = FactsService()
>>> data = pd.DataFrame(data=[['2017-12-20 00:00', 1, 'native-name-1'],
                              ['2017-12-21 00:00', 2, 'native-name-2']],
                        columns=['fact_time', 'fact_value', 'native_name'])
>>> fs.put_facts(building_id=26, data=data)
```

(continues on next page)

(continued from previous page)

```
    {'data': {'building_id': 26,
              'max_process_timestamp': '2017-12-21 00:00',
              'min_process_timestamp': '2017-12-21 00:00',
              'records_stored': 1},
     'message': {'field_errors': ['native-name-1']}}
```

get_point_mapping

FactsService.**get_point_mapping**(*building_id*, *equipment_names=[]*, *equipment_types=[]*, *point_classes=[]*, *eco_point_ids=[]*, *display_names=[]*, *native_names=[]*, *point_class_expression=[]*, *native_name_expression=[]*, *display_name_expression=[]*, *is_active=True*, *result_format='pandas'*)

Return the point mapping for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Point-Mapping/point_mapping_get

Args:

building_id (str): Building id to get facts for.

Example: 1

Kwargs:

equipment_names (list): List of equipment names to filter facts for.

Example: ['VAV_01', 'VAV_02']

equipment_types (list): List of equipment types to filter facts for.

Example: ['VAV', 'AHU']

point_classes (list): List of point_classes to filter facts for.

Example: ['SpaceAirTemperature', 'CoolingCoilUnitFeedback']

eco_point_ids (list): List of eco_point_ids to filter facts for.

Example: [1, 2, 3]

display_names (list): List of display_names to filter facts for.

Example: ['SpaceTemp', 'AirFlow']

native_names (list): List of native_names to filter facts for.

Example: ['name-1', 'name-2']

point_class_expression (list): List of equipment / equipment type + point class regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited. for.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

native_name_expression (list): List of native-name regex expressions to filter facts.

Example: ['nam.*', 'nati-.*']

display_name_expression (list): List of equipment / equipment type + display name regex expressions to filter facts. Equipment / equipment type and point class regex expressions are space delimited.

Example: ['VAV.* SpaceAirTemperature', 'AHU Space.*']

is_active (boolean): Return the active / in-active native-names

Example: True

result_format (str): Output format type. (Pandas, tuple, csv, json)

Example: 'pandas'

Returns: (DataFrame or list or csv or json depending on the requested result format).

DataFrame Example:

```
index    eco_point_id     display_name   native_name        point_class
↪        equipment_name   equipment_type        last_updated
=====  ================   ============  ============
↪=========================   ==============  ================
↪======================
0            192          'SpaceTemp'     'name-1'    'SpaceAirTemperature
↪'         'VAV-01'         'VAV'            "2017-12-07T19:04:18Z"

1            304          'CoolingCoil'   'name-2'
↪'CoolingCoilUnitFeedback'     'AHU-01'        'AHU'            "2017-12-
↪07T19:04:18Z"
```

Json Example:

```
{
  "data": [
    {
      "equipment_name": "VAV_01",
      "equipment_type": "VAV",
      "native_name": "Native-Name-1",
      "eco_point_id": 3,
      "point_class": "SpaceAirTemperature",
      "native_name_id": 1283,
      "display_name": "SpaceTemp",
      "last_updated": "2017-11-17T17:44:04Z"
    },
    {
      "equipment_name": "VAV_02",
      "equipment_type": "VAV",
      "native_name": "Native-Name-1",
      "eco_point_id": 4,
      "point_class": "SpaceAirTemperature",
      "native_name_id": 1283,
      "display_name": "SpaceTemp",
      "last_updated": "2017-11-17T17:44:04Z"
    }
  ]
}
```

Csv Example:

```
'point_class,display_name,native_name,
equipment_type,equipment_name,eco_point_id


SpaceAirTemperature,SpaceTemp,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T,VAV,VAV_301,85743,
```

(continues on next page)

(continued from previous page)

```
SpaceAirTemperatureSetPointWhenCooling,
SpaceTempSetPoint_ActualCooling,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET,VAV,VAV_301,
85744,

SpaceAirTemperatureSetPointWhenHeating,
SpaceTempSetPoint_ActualHeating,
UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.AHTG-SET,VAV,VAV_301,85745'
```

Tuple Example:

```
[response_tuple(point_class='SpaceAirTemperature', display_name='SpaceTemp
↪',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T
↪',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85743),
 response_tuple(point_class='SpaceAirTemperatureSetPointWhenCooling',
                display_name='SpaceTempSetPoint_ActualCooling',
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-
↪SET',
                equipment_type='VAV', equipment_name='VAV_301', eco_point_
↪id=85744)]
```

Note: And invalid api request will return back the raw api response.

***Example*:** {"message": {"NoData": "No data for provided parameters"}}

Example Usage:

```
>>> from eco_connect import FactsService
>>> facts_service = FactsService()
>>> facts_service.get_point_mapping(building_id=26,
                                    result_format='json')
        {
          "data": [
            {
              "equipment_name": "VAV_01",
              "equipment_type": "VAV",
              "native_name": "Native-Name-1",
              "eco_point_id": 3,
              "point_class": "SpaceAirTemperature",
              "native_name_id": 1283,
              "display_name": "SpaceTemp",
              "last_updated": "2017-11-17T17:44:04Z"
            },
            {
              "equipment_name": "VAV_02",
              "equipment_type": "VAV",
              "native_name": "Native-Name-1",
              "eco_point_id": 4,
              "point_class": "SpaceAirTemperature",
              "native_name_id": 1283,
```

(continues on next page)

(continued from previous page)

```
            "display_name": "SpaceTemp",
            "last_updated": "2017-11-17T17:44:04Z"
          }
        ]
      }
```

get_native_names

`FactsService.`**`get_native_names`**(*building_id*, *native_name=None*, *is_active=True*, *result_format='pandas'*)

Return the native names for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Native-Names/native_names_get

Args:

building_id (str): Building id to get facts for.

Example: 1

Kwargs:

native_name (string): Filter on this native_name

Example: 'native-name-1'

is_active (boolean): Return the active / in-active native-names

Example: True

result_format (str): Output format type. (Pandas, tuple, csv, json)

Example: 'pandas'

Returns: (DataFrame or list or csv or json depending on the requested result format).

DataFrame Example:

```
index   native_name_id     native_name   native_name   expecting_data
↪    origin       trend_period      last_updated
=====  ================   ============  ============  ================
↪==============  ================  ====================
0            192          'name-1'        'name-1'          True
↪  'CLIENT'        300 seconds    2017-12-19T20:36:36Z

1            304          'name-2'        'name-2'          False
↪ 'ECORITHM'       300 seconds    2017-12-19T20:36:36Z
```

Json Example:

```
{'data': [{'expecting_data': True,
   'last_updated': '2017-12-19T20:36:36Z',
   'native_name': 'UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T',
   'native_name_id': 88826,
   'origin': 'CLIENT',
   'trend_period': '300 seconds',
   'trend_type': 'INTERVAL'},
  {'expecting_data': True,
   'last_updated': '2017-12-19T20:36:36Z',
```

(continues on next page)

(continued from previous page)

```
   'native_name': 'UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET',
   'native_name_id': 88827,
   'origin': 'CLIENT',
   'trend_period': '300 seconds',
   'trend_type': 'INTERVAL'}
   }
```

Csv Example:

```
'native_name_id,native_name,expecting_data,
origin,trend_type,trend_period,last_updated


88826,UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T,
True,CLIENT,INTERVAL,300 seconds,2017-12-19T20:36:36Z,

88827,UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET,True,
CLIENT,INTERVAL,300 seconds,2017-12-19T20:36:36Z'
```

Tuple Example:

```
[response_tuple(native_name_id=88826,
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T
↪',
                expecting_data=True, origin='CLIENT',
                trend_type='INTERVAL',
                trend_period='300 seconds',
                last_updated='2017-12-19T20:36:36Z'),
 response_tuple(native_name_id=88827,
                native_name='UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-
↪SET',
                expecting_data=True, origin='CLIENT',
                trend_type='INTERVAL',
                trend_period='300 seconds',
                last_updated='2017-12-19T20:36:36Z')
```

Note: And invalid api request will return back the raw api response.

Example: {"message": {"NoData": "No data for provided parameters"}}

Example Usage:

```
>>> from eco_connect import FactsService
>>> facts_service = FactsService()
>>> facts_service.get_native_names(building_id=26,
                                   result_format='json')
        {'data': [{'expecting_data': True,
           'last_updated': '2017-12-19T20:36:36Z',
           'native_name': 'UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ZN-T',
           'native_name_id': 88826,
           'origin': 'CLIENT',
           'trend_period': '300 seconds',
           'trend_type': 'INTERVAL'},
          {'expecting_data': True,
           'last_updated': '2017-12-19T20:36:36Z',
```

(continues on next page)

(continued from previous page)

```
            'native_name': 'UCSB/275/VAV_301/NAE11/N2-2.275-VAV-301.ACLG-SET',
            'native_name_id': 88827,
            'origin': 'CLIENT',
            'trend_period': '300 seconds',
            'trend_type': 'INTERVAL'}
            }
```

get_equipment

`FactsService.`**`get_equipment`**(*building_id*, *equipment_name=None*, *equipment_type=None*, *is_active=True*, *result_format='pandas'*)

Return the equipments for a building.

API documentation: http://facts.prod.ecorithm.com/api/v1/#/Equipment/equipment_get

Args:

building_id (str): Building id to get facts for.

Example: 1

Kwargs:

equipment_name (string): Equipment name to filter on.

Example: 'VAV_01'

equipment_type (string): equipment_type to filter on.

Example: 'AHU'

is_active (boolean): Return the active / in-active native-names

Example: True

result_format (str): Output format type. (Pandas, tuple, csv, json)

Example: 'pandas'

Returns: (DataFrame or list or csv or json depending on the requested result format).

DataFrame Example:

```
index    equipment_id     equipment_type    equipment_name       last_
↪updated
=====  ================   ==============   ================  ⌙
↪=====================
0            192              'AHU'            'VAV-01'       '2017-12-
↪07T19:04:18Z'
1            304              'VAV'            'AHU-01'       '2017-12-
↪07T19:04:18Z'
```

Json Example:

```
{'data': [{'equipment_id': 1092,
           'equipment_name': 'AHU_G1',
           'equipment_type': 'AHU',
           'last_updated': '2017-12-07T19:04:18Z'},
           {'equipment_id': 1093,
```

(continues on next page)

(continued from previous page)

```
            'equipment_name': 'AHU_G1_ExhaustFan',
            'equipment_type': 'AHU',
            'last_updated': '2017-12-07T19:04:18Z'}]
}
```

Csv Example:

```
'equipment_id,equipment_type,equipment_name,last_updated


1092,AHU,AHU_G1,2017-12-07T19:04:18Z


1094,AHU,AHU_G1_ReturnFanVFD,2017-12-07T19:04:18Z
```

Tuple Example:

```
[response_tuple(equipment_id=1092, equipment_type='AHU',
                equipment_name='AHU_G1',
                last_updated='2017-12-07T19:04:18Z'),
 response_tuple(equipment_id=1093, equipment_type='AHU',
                equipment_name='AHU_G1_ExhaustFan',
                last_updated='2017-12-07T19:04:18Z')]
```

Note: And invalid api request will return back the raw api response.

***Example*:** {“message”: {“NoData”: “No data for provided parameters”}}

Example Usage:

```
>>> from eco_connect import FactsService
>>> facts_service = FactsService()
>>> facts_service.get_equipment(building_id=26,
                                result_format='json')
        {'data': [{'equipment_id': 1092,
                   'equipment_name': 'AHU_G1',
                   'equipment_type': 'AHU',
                   'last_updated': '2017-12-07T19:04:18Z'},
                   {'equipment_id': 1093,
                    'equipment_name': 'AHU_G1_ExhaustFan',
                    'equipment_type': 'AHU',
                    'last_updated': '2017-12-07T19:04:18Z'}]
        }
```

INDEX

F

G

P