octo-api

Release 0.1.1

Python interface to the Octopus Energy API.

Dominic Davis-Foster

Jun 21, 2021

Contents

Chapter

ONE

Installation

1.1 from PyPI

```
$ python3 -m pip install octo-api --user
```

1.2 from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/octo-api@master --user
```

Chapter

TWO

API Reference

2.1 octo_api.api

The primary interface to the Octopus Energy API.

Note: The Octopus Energy API uses the term "Grid Supply Point" (GSP) to refer to what are actually the 14 former Public Electricity Suppliers. The GSP terminology has been used here to better reflect the REST API.

Classes:

OctoAPI(api_key)	The primary interface to the Octopus Energy API.

class OctoAPI(*api_key*)

Bases: object

The primary interface to the Octopus Energy API.

Parameters **api_key** (str) – API key to access the Octopus Energy API.

If you are an Octopus Energy customer, you can generate an API key from your online dashboard.

Attributes:

API_BASE	The base URL of the Octopus Energy API.
API_KEY	The API key to access the Octopus Energy API.

Methods:

get_consumption(mpan, serial_number, fuel[, ...])	Return a list of consumption values for half-hour periods for a given meter-point and meter.
get_grid_supply_point(postcode)	Returns the grid supply point for the given postcode.
get_meter_point_details(mpan)	Retrieve the details of a meter-point.
get_product_info(product_code[, ...])	Retrieve the details of a product (including all its tariffs) for a particular point in time.
get_products([is_variable, is_green, ...])	Returns a list of energy products.
get_tariff_charges(product_code, ...[, ...])	Returns a list of time periods and their associated unit rates charges.

API_BASE

Type: SlumberURL

The base URL of the Octopus Energy API.

API_KEY
Type: Secret

The API key to access the Octopus Energy API.

get_consumption(*mpan*, *serial_number*, *fuel*, *period_from=None*, *period_to=None*, *page_size=100*, *reverse=False*, *group_by=None*)
Return a list of consumption values for half-hour periods for a given meter-point and meter.

Unit of measurement:

- Electricity meters: kWh
- SMETS1 Secure gas meters: kWh
- SMETS2 gas meters: m^3

Attention: Half-hourly consumption data is only available for smart meters. Requests for consumption data for non-smart meters will return an empty response payload.

Parameters

- **mpan** (str) – The electricity meter-point's MPAN or gas meter-point's MPRN.
- **serial_number** (str) – The meter's serial number.
- **fuel** (Literal['electricity', 'gas'])
- **period_from** (Optional[datetime]) – Show consumption for periods which start at or after the given datetime. This parameter can be provided on its own. Default None.
- **period_to** (Optional[datetime]) – Show consumption for periods which start at or before the given datetime. This parameter also requires providing the period_from parameter to create a range. Default None.
- **page_size** (int) – Page size of returned results. Default is 100, maximum is 25,000 to give a full year of half-hourly consumption details.
- **reverse** (bool) – Returns the results ordered from most oldest to newest. By default the results are from most recent backwards.
- **group_by** (Optional[str]) – The grouping of the consumption data. By default the consumption is returned in half-hour periods.

  Possible alternatives are:

  - 'hour'
  - 'day'
  - 'week'
  - 'month'
  - 'quarter'

Return type *PaginatedResponse*[*Consumption*]

get_grid_supply_point(*postcode*)
Returns the grid supply point for the given postcode.

Parameters **postcode** (str)

Raises `ValueError` if the postcode cannot be mapped to a GSP.

Return type *Region*

get_meter_point_details(*mpan*)
Retrieve the details of a meter-point.

This can be used to get the GSP of a given meter-point.

Parameters **mpan** (`str`) – The electricity meter-point's MPAN.

Return type *MeterPointDetails*

Returns

get_product_info(*product_code*, *tariffs_active_at=None*)
Retrieve the details of a product (including all its tariffs) for a particular point in time.

Parameters

- **product_code** (`str`) – The code of the product to be retrieved, for example `VAR-17-01-11`.
- **tariffs_active_at** (`Optional[datetime]`) – The point in time in which to show the active charges. Defaults to current datetime. Default `None`.

Example

```
>>> api.get_product_info(product_code='VAR-17-01-11')
octo_api.products.DetailedProduct(
    available_from='2017-01-11T10:00:00+00:00',
    available_to='2018-02-15T00:00:00+00:00',
    brand='S_ENERGY',
    code='7-01-11',
    description='This variable tariff always offers great value - driven by␣
↪our'
                'belief that prices should be fair for the long term, not␣
↪just a'
                'fixed term. We aim for 50% renewable electricity on this␣
↪tariff.',
    display_name='pus',
    full_name='ctopus January 2017 v1',
    is_business=False,
    is_green=False,
    is_prepay=False,
    is_restricted=False,
    is_tracker=False,
    is_variable=True,
    links=[
        {
            'href': 'https://api.octopus.energy/v1/products/VAR-17-01-11/',
            'method': 'GET',
            'rel': 'self'
        }
    ],
    term=None,
    tariffs_active_at='2020-10-26T11:15:17.208285+00:00',
    single_register_electricity_tariffs=RegionalTariffs(['direct_debit_monthly
↪']),
    dual_register_electricity_tariffs=RegionalTariffs(['direct_debit_monthly
↪']),
```

(continues on next page)

(continued from previous page)

```
    single_register_gas_tariffs=RegionalTariffs(['direct_debit_monthly']),
    sample_quotes=RegionalQuotes([dual_fuel_dual_rate, dual_fuel_single_rate,␣
↪electricity_dual_rate, electricity_single_rate]),
    sample_consumption={
        'electricity_single_rate': {'electricity_standard': 2900},
        'electricity_dual_rate': {
            'electricity_day': 2436,
            'electricity_night': 1764
        },
        'dual_fuel_single_rate': {
            'electricity_standard': 2900,
            'gas_standard': 12000
        },
        'dual_fuel_dual_rate': {
            'electricity_day': 2436,
            'electricity_night': 1764,
            'gas_standard': 12000
        }
    },
)
```

Return type *DetailedProduct*

get_products(*is_variable=None*, *is_green=None*, *is_tracker=None*, *is_prepay=None*, *is_business=False*, *available_at=None*)

Returns a list of energy products.

By default, the results only include public energy products. Authenticated organisations will also see products available to their organisation.

Parameters

- **is_variable** (Optional[bool]) – Show only variable products. Default None.
- **is_green** (Optional[bool]) – Show only green products. Default None.
- **is_tracker** (Optional[bool]) – Show only tracker products. Default None.
- **is_prepay** (Optional[bool]) – Show only pre-pay products. Default None.
- **is_business** (bool) – Show only business products. Default False.
- **available_at** (Optional[datetime]) – Show products available for new agreements on the given datetime. Defaults to the current datetime, effectively showing products that are currently available.

Example

```
>>> api.get_products()[0]
octo_api.products.Product(
    available_from='2016-01-01T:00:00:00+00:00',
    available_to=None,
    brand='AFFECT_ENERGY',
    code='1201',
    description='Affect Standard Tariff',
    display_name='Affect Standard Tariff',
    full_name='Affect Standard Tariff',
    is_business=False,
```

(continues on next page)

(continued from previous page)

```
    is_green=False,
    is_prepay=False,
    is_restricted=False,
    is_tracker=False,
    is_variable=True,
    links=[
        {
            'href': 'https://api.octopus.energy/v1/products/1201/',
            'method': 'GET',
            'rel': 'self'
        }
    ],
    term=None,
    direction='IMPORT',
)
```

Return type *PaginatedResponse*[*Product*]

get_tariff_charges(*product_code*, *tariff_code*, *fuel*, *rate_type*, *period_from=None*, *period_to=None*, *page_size=100*)

Returns a list of time periods and their associated unit rates charges.

If the tariff has a fixed unit rate the list will only contain one element.

Parameters

- **product_code** (str) – The code of the product to be retrieved, for example VAR-17-01-11.
- **tariff_code** (str) – The code of the tariff to be retrieved, for example E-1R-VAR-17-01-11-A. From what I can tell the format is:

```
<E for electricity><optional hyphen><1R for single rate?><the product␣
↪code>-<the grid supply point>
```

- **fuel** (Literal['electricity', 'gas'])
- **rate_type** (*RateType*)
- **period_from** (Optional[datetime]) – Show charges active from the given datetime (inclusive). This parameter can be provided on its own. Default None.
- **period_to** (Optional[datetime]) – Show charges active up to the given datetime (exclusive). You must also provide the period_from parameter in order to create a range. Default None.
- **page_size** (int) – Page size of returned results. Default is 100, maximum is 1,500 to give up to a month of half-hourly prices.

Note: If you're using this API to query future unit-rates of the Agile Octopus product, note that day-ahead prices are normally created by 4pm in the Europe/London timezone. Further, the market index used to calculate unit rates is based in the CET timezone (UTC+1) and so its "day" corresponds to 11pm to 11pm in UK time. Hence, if you query today's unit rates before 4pm, you'll get 46 results back rather than 48.

Return type *PaginatedResponse*[*RateInfo*]

2.2 octo_api.consumption

Class to represent consumption data.

Classes:

Consumption(consumption, interval_start, ...)	Represents the consumption for a given period of time.

class Consumption(*consumption*, *interval_start*, *interval_end*)

Bases: object

Represents the consumption for a given period of time.

Parameters

- **consumption** (float) – The consumption.
- **interval_start** (Union[str, datetime, None]) – The start of the time period.
- **interval_end** (Union[str, datetime, None]) – The end of the time period.

Attributes:

consumption	The consumption.
interval_end	The end of the time period.
interval_start	The start of the time period.

Methods:

from_dict(d)	Construct an instance of *Consumption* from a dictionary.
to_dict([convert_values])	Returns a dictionary containing the contents of the *Consumption* object.

consumption

Type: float

The consumption.

classmethod from_dict(*d*)

Construct an instance of *Consumption* from a dictionary.

Parameters **d** (Mapping[str, Any]) – The dictionary.

interval_end

Type: datetime

The end of the time period.

interval_start

Type: datetime

The start of the time period.

to_dict(*convert_values=False*)
Returns a dictionary containing the contents of the *Consumption* object.

Parameters **convert_values** (bool) – Recursively convert values into dictionaries, lists etc. as appropriate. Default False.

Return type MutableMapping[str, Any]

2.3 octo_api.pagination

Class for handling paginated API responses.

Classes:

OctoResponse	TypedDict representing the raw JSON data returned by the Octopus Energy API.
PaginatedResponse(query_url[, query_params, ...])	Represents a multi-page response from a REST API.

typeddict OctoResponse
Bases: dict

TypedDict representing the raw JSON data returned by the Octopus Energy API.

Required Keys

- **count** (int) – The total number of responses.
- **next** (str) – The URL of the next page of results.
- **previous** (str) – The URL of the previous page of results.
- **results** (List[Dict[str, Any]]) – The current page of results.

class PaginatedResponse(*query_url*, *query_params=None*, *obj_type=<class 'dict'>*)
Bases: Iterable[~_T]

Represents a multi-page response from a REST API.

The items within the response can be iterated over or accessed by their indices. The total number of items can be accessed with len(response).

Parameters

- **query_url** (SlumberURL) – The initial query URL.
- **query_params** (Optional[MutableMapping[str, Any]]) – The parameters to the query. Default None.
- **obj_type** (Type) – The object to convert the response data to. Default dict.

Note: This class assumes the JSON response is in the format used by Django REST framework.

The response should be in the following format:

```
{
    "count": 1023,
    "next": "https://api.example.org/accounts/?page=5",
    "previous": "https://api.example.org/accounts/?page=3",
    "results": []
}
```

See https://www.django-rest-framework.org/api-guide/pagination/ for more information.

Methods:

__eq__(other)	Return `self == other`.
__getitem__(item)	Returns the item or items in the *PaginatedResponse*, as given by the index or slice.
__iter__()	Iterate over items in the *PaginatedResponse*.
__len__()	Returns the number of items in the *PaginatedResponse*.

__eq__(*other*)
Return `self == other`.

Return type `bool`

__getitem__(*item*)
Returns the item or items in the *PaginatedResponse*, as given by the index or slice.

Parameters **item** (`Union[int, slice]`)

Return type `Union[~_T, List[~_T]]`

Overloads

- *__getitem__*(**item**: `int`) -> `~_T`
- *__getitem__*(**item**: `slice`) -> `List[~_T]`

__iter__()
Iterate over items in the *PaginatedResponse*.

Return type `Iterator[~_T]`

__len__()
Returns the number of items in the *PaginatedResponse*.

Return type `int`

2.4 octo_api.products

Classes to model products and tariffs.

Classes:

BaseProduct(available_from, available_to, ...)	Represents an Octopus Energy product.
Product(available_from, available_to, brand, ...)	Represents an Octopus Energy product.
DetailedProduct(available_from, ...)	Represents an Octopus Energy product, with detailed tariff information.
Tariff(code, standing_charge_exc_vat, ...[, ...])	Represents a tariff for a product.
RateInfo(value_exc_vat, value_inc_vat, ...)	Represents the unit rate of a tariff at a particular period in time.
RegionalTariffs	Mapping of GSP regions to a mapping of payment methods to *Tariffs*.
RegionalQuotes	Mapping of GSP regions to a mapping of payment methods to a mapping of fuel types to Quotes.

class BaseProduct(*available_from*, *available_to*, *brand*, *code*, *description*, *display_name*, *full_name*, *is_business*, *is_green*, *is_prepay*, *is_restricted*, *is_tracker*, *is_variable*, *links*, *term*)

Bases: object

Represents an Octopus Energy product.

Parameters

- **available_from** (Union[str, datetime, None]) – The date from which the product is available.
- **available_to** (Union[str, datetime, None]) – The date until which the product is available.
- **brand** (str) – The brand under which the product is sold.
- **code** (str) – The tariff code.
- **description** (str) – A description of the product.
- **display_name** (str) – The display name of the product.
- **full_name** (str) – The name of the product.
- **is_business** (bool) – Whether the product is for businesses.
- **is_green** (bool) – Whether the product is green.
- **is_prepay** (bool) – Whether the product is prepay.
- **is_restricted** (bool) – Whether the product is restricted.
- **is_tracker** (bool) – Whether the product tracks the wholesale electricity rate.
- **is_variable** (bool) – Whether the product has a variable tariff.
- **links** (Iterable[MutableMapping[str, Any]]) – Links associated with this product.
- **term** (Optional[int]) – The number of months that a product lasts for if it is fixed length.

Attributes:

available_from	The date from which the product is available.
available_to	The date until which the product is available.
brand	The brand under which the product is sold.
code	The code of the product.
description	A description of the product.
display_name	The display name of the product.
full_name	The name of the product.
is_business	Whether the product is for businesses.
is_green	Whether the product is green.
is_prepay	Whether the product is prepay.
is_restricted	Whether the product is restricted.
is_tracker	Whether the product tracks the wholesale electricity rate.
is_variable	Whether the product has a variable tariff.
links	Links associated with this product.
term	The number of months that a product lasts for if it is fixed length.

Methods:

from_dict(d)	Construct an instance of *BaseProduct* from a dictionary.
to_dict([convert_values])	Returns a dictionary containing the contents of the *BaseProduct* object.

available_from
Type: Optional[datetime]

The date from which the product is available.

available_to
Type: Optional[datetime]

The date until which the product is available.

brand
Type: str

The brand under which the product is sold.

code
Type: str

The code of the product.

description
Type: str

A description of the product.

display_name
Type: str

The display name of the product.

classmethod from_dict(*d*)
Construct an instance of *BaseProduct* from a dictionary.

Parameters **d** (Mapping[str, Any]) – The dictionary.

full_name
Type: str

The name of the product.

is_business
Type: bool

Whether the product is for businesses.

is_green
Type: bool

Whether the product is green.

is_prepay
Type: bool

Whether the product is prepay.

is_restricted
Type: bool

Whether the product is restricted.

is_tracker
Type: bool

Whether the product tracks the wholesale electricity rate.

is_variable
Type: bool

Whether the product has a variable tariff.

links
Type: List[MutableMapping[str, Any]]

Links associated with this product.

term
Type: Optional[int]

The number of months that a product lasts for if it is fixed length.

to_dict(*convert_values=False*)
Returns a dictionary containing the contents of the *BaseProduct* object.

Parameters **convert_values** (bool) – Recursively convert values into dictionaries, lists etc. as appropriate. Default False.

Return type MutableMapping[str, Any]

class Product(*available_from*, *available_to*, *brand*, *code*, *description*, *display_name*, *full_name*, *is_business*, *is_green*, *is_prepay*, *is_restricted*, *is_tracker*, *is_variable*, *links*, *term*, *direction*)

Bases: *octo_api.products.BaseProduct*

Represents an Octopus Energy product.

Parameters

- **available_from** (Union[str, datetime, None]) – The date from which the product is available.
- **available_to** (Union[str, datetime, None]) – The date until which the product is available.
- **brand** (str) – The brand under which the product is sold.
- **code** (str) – The tariff code.
- **description** (str) – A description of the product.
- **display_name** (str) – The display name of the product.
- **full_name** (str) – The name of the product.
- **is_business** (bool) – Whether the product is for businesses.
- **is_green** (bool) – Whether the product is green.
- **is_prepay** (bool) – Whether the product is prepay.
- **is_restricted** (bool) – Whether the product is restricted.
- **is_tracker** (bool) – Whether the product tracks the wholesale electricity rate.
- **is_variable** (bool) – Whether the product has a variable tariff.
- **links** (Iterable[MutableMapping[str, Any]]) – Links associated with this product.
- **term** (Optional[int]) – The number of months that a product lasts for if it is fixed length.
- **direction** (str) – The direction of the product (supply to the customer or supply to the grid).

Attributes:

direction	The direction of the product (supply to the customer or supply to the grid).

direction

Type: str

The direction of the product (supply to the customer or supply to the grid).

class DetailedProduct(*available_from*, *available_to*, *brand*, *code*, *description*, *display_name*, *full_name*, *is_business*, *is_green*, *is_prepay*, *is_restricted*, *is_tracker*, *is_variable*, *links*, *term*, *tariffs_active_at*, *single_register_electricity_tariffs*, *dual_register_electricity_tariffs*, *single_register_gas_tariffs*, *sample_quotes*, *sample_consumption*)

Bases: *octo_api.products.BaseProduct*

Represents an Octopus Energy product, with detailed tariff information.

Each *_tariffs object will have up to 14 keys; one for each GSP. For each GSP the applicable tariffs are listed under their associated payment method, e.g. direct_debit_monthly.

- The `standard_unit_rate_*` values are listed in p/kWh (pence per kilowatt hour).
- The `standing_charge_*` values are listed in p/day (pence per day).
- The `annual_cost_*` values are listed in p (pence).

Parameters

- **available_from** (Union[str, datetime, None]) – The date from which the product is available.
- **available_to** (Union[str, datetime, None]) – The date until which the product is available.
- **brand** (str) – The brand under which the product is sold.
- **code** (str) – The tariff code.
- **description** (str) – A description of the product.
- **display_name** (str) – The display name of the product.
- **full_name** (str) – The name of the product.
- **is_business** (bool) – Whether the product is for businesses.
- **is_green** (bool) – Whether the product is green.
- **is_prepay** (bool) – Whether the product is prepay.
- **is_restricted** (bool) – Whether the product is restricted.
- **is_tracker** (bool) – Whether the product tracks the wholesale electricity rate.
- **is_variable** (bool) – Whether the product has a variable tariff.
- **links** (Iterable[MutableMapping[str, Any]]) – Links associated with this product.
- **term** (Optional[int]) – The number of months that a product lasts for if it is fixed length.
- **tariffs_active_at** (Union[str, datetime, None])
- **single_register_electricity_tariffs** (Dict[str, Dict[str, Dict[str, Any]]]) – Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.
- **dual_register_electricity_tariffs** (Dict[str, Dict[str, Dict[str, Any]]]) – Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.
- **single_register_gas_tariffs** (Dict[str, Dict[str, Dict[str, Any]]]) – Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.
- **sample_quotes** (Dict[str, Dict[str, Dict[str, Any]]])
- **sample_consumption** (Dict)

Attributes:

dual_register_electricity_tariffs	Mapping of GSPs to applicable tariffs for each payment method, e.g.
sample_consumption	
sample_quotes	
single_register_electricity_tariffs	Mapping of GSPs to applicable tariffs for each payment method, e.g.

continues on next page

Table 13 – continued from previous page

single_register_gas_tariffs	Mapping of GSPs to applicable tariffs for each payment method, e.g.
tariffs_active_at	

Methods:

from_dict(d)	Construct an instance of *DetailedProduct* from a dictionary.
to_dict([convert_values])	Returns a dictionary containing the contents of the *DetailedProduct* object.

dual_register_electricity_tariffs
Type: Dict

Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.

classmethod from_dict(*d*)
Construct an instance of *DetailedProduct* from a dictionary.

Parameters **d** (Mapping[str, Any]) – The dictionary.

sample_consumption
Type: Dict

sample_quotes
Type: Dict

single_register_electricity_tariffs
Type: Dict

Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.

single_register_gas_tariffs
Type: Dict

Mapping of GSPs to applicable tariffs for each payment method, e.g. direct_debit_monthly.

tariffs_active_at
Type: Optional[datetime]

to_dict(*convert_values=False*)
Returns a dictionary containing the contents of the *DetailedProduct* object.

Parameters **convert_values** (bool) – Recursively convert values into dictionaries, lists etc. as appropriate. Default False.

Return type MutableMapping[str, Any]

class RateInfo(*value_exc_vat*, *value_inc_vat*, *valid_from*, *valid_to*)
Bases: object

Represents the unit rate of a tariff at a particular period in time.

Parameters

- **value_exc_vat** (float) – In p/kWh (pence per kilowatt hour).
- **value_inc_vat** (float) – In p/kWh (pence per kilowatt hour).

- **valid_from** (Union[str, datetime, None]) – The date and time from which this rate is in effect.
- **valid_to** (Union[str, datetime, None]) – The date and time until which this rate is in effect, or None if this rate continues in perpetuity.

Methods:

from_dict(d)	Construct an instance of *RateInfo* from a dictionary.
to_dict([convert_values])	Returns a dictionary containing the contents of the *RateInfo* object.

Attributes:

valid_from	The date and time from which this rate is in effect.
valid_to	The date and time until which this rate is in effect, or None if this rate continues in perpetuity.
value_exc_vat	In p/kWh (pence per kilowatt hour).
value_inc_vat	In p/kWh (pence per kilowatt hour).

classmethod from_dict(*d*)
Construct an instance of *RateInfo* from a dictionary.

Parameters **d** (Mapping[str, Any]) – The dictionary.

to_dict(*convert_values=False*)
Returns a dictionary containing the contents of the *RateInfo* object.

Parameters **convert_values** (bool) – Recursively convert values into dictionaries, lists etc. as appropriate. Default False.

Return type MutableMapping[str, Any]

valid_from
Type: datetime

The date and time from which this rate is in effect.

valid_to
Type: Optional[datetime]

The date and time until which this rate is in effect, or None if this rate continues in perpetuity.

value_exc_vat
Type: float

In p/kWh (pence per kilowatt hour).

value_inc_vat
Type: float

In p/kWh (pence per kilowatt hour).

class Tariff(*code*, *standing_charge_exc_vat*, *standing_charge_inc_vat*, *online_discount_exc_vat*, *online_discount_inc_vat*, *dual_fuel_discount_exc_vat*, *dual_fuel_discount_inc_vat*, *exit_fees_exc_vat*, *exit_fees_inc_vat*, *links*, *standard_unit_rate_exc_vat=None*, *standard_unit_rate_inc_vat=None*, *day_unit_rate_exc_vat=None*, *day_unit_rate_inc_vat=None*, *night_unit_rate_exc_vat=None*, *night_unit_rate_inc_vat=None*)

Bases: `object`

Represents a tariff for a product.

Parameters

- **code** (`str`) – The tariff code.
- **standing_charge_exc_vat** (`float`) – In p/day (pence per day).
- **standing_charge_inc_vat** (`float`) – In p/day (pence per day).
- **online_discount_exc_vat** (`int`)
- **online_discount_inc_vat** (`int`)
- **dual_fuel_discount_exc_vat** (`int`)
- **dual_fuel_discount_inc_vat** (`int`)
- **exit_fees_exc_vat** (`int`)
- **exit_fees_inc_vat** (`int`)
- **links** (`List[Dict[str, Any]]`) – Links associated with this product.
- **standard_unit_rate_exc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.
- **standard_unit_rate_inc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.
- **day_unit_rate_exc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.
- **day_unit_rate_inc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.
- **night_unit_rate_exc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.
- **night_unit_rate_inc_vat** (`Optional[float]`) – In p/kWh (pence per kilowatt hour). Default `None`.

Methods:

from_dict(d)	Construct an instance of *Tariff* from a dictionary.
to_dict([convert_values])	Returns a dictionary containing the contents of the *Tariff* object.

classmethod from_dict(*d*)

Construct an instance of *Tariff* from a dictionary.

Parameters **d** (`Mapping[str, Any]`) – The dictionary.

to_dict(*convert_values=False*)

Returns a dictionary containing the contents of the *Tariff* object.

Parameters **convert_values** (bool) – Recursively convert values into dictionaries, lists etc. as appropriate. Default False.

Return type MutableMapping[str, Any]

class RegionalTariffs

Bases: Dict[str, Dict[str, *Tariff*]]

Mapping of GSP regions to a mapping of payment methods to *Tariffs*.

Methods:

__str__()	Return str(self).

__str__()

Return str(self).

Return type str

2.5 octo_api.utils

Utility functions.

Classes:

MeterPointDetails(mpan, gsp, profile_class)	Information about a meter point.
RateType(value)	Enumeration of different rate types.
Region(value)	Enumeration of different electricity supply regions.

Functions:

add_repr(cls)	Add a pretty-printed __repr__ function to the decorated attrs class.
from_iso_zulu(the_datetime)	Constructs a datetime.datetime object from an ISO 8601 format string.

namedtuple MeterPointDetails(*mpan*, *gsp*, *profile_class*)

Bases: NamedTuple

Information about a meter point.

Fields

0) **mpan** (str) – The meter point access number.
1) **gsp** (*Region*) – The grid supply point/region that the meter point is located in.
2) **profile_class** (int) – The profile class of the meter point.

- **Profile Class 1** – Domestic Unrestricted Customers
- **Profile Class 2** – Domestic Economy 7 Customers
- **Profile Class 3** – Non-Domestic Unrestricted Customers

- **Profile Class 4** – Non-Domestic Economy 7 Customers
- **Profile Class 5** – Non-Domestic Maximum Demand (MD) Customers with a Peak Load Factor (LF) of less than 20%
- **Profile Class 6** – Non-Domestic Maximum Demand Customers with a Peak Load Factor between 20% and 30%
- **Profile Class 7** – Non-Domestic Maximum Demand Customers with a Peak Load Factor between 30% and 40%
- **Profile Class 8** – Non-Domestic Maximum Demand Customers with a Peak Load Factor over 40%

Information from https://www.elexon.co.uk/knowledgebase/profile-classes/

See also:

Load Profiles and their use in Electricity Settlement by Elexon

__repr__()
Return a string representation of the *MeterPointDetails*.

Return type str

enum RateType(*value*)
Bases: enum_tools.custom_enums.StrEnum

Enumeration of different rate types.

Member Type str

Valid values are as follows:

StandingCharge = <RateType.StandingCharge: 'standing-charges'>

StandardUnitRate = <RateType.StandardUnitRate: 'standard-unit-rates'>

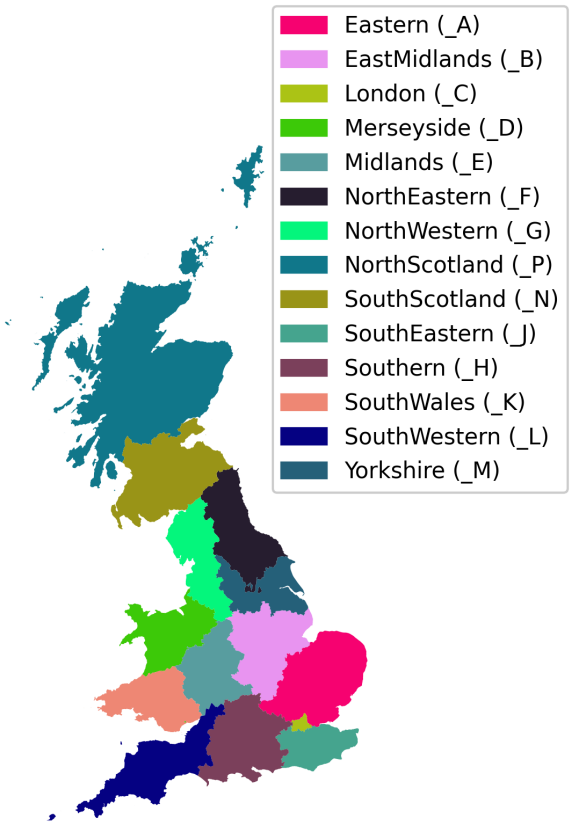
DayUnitRate = <RateType.DayUnitRate: 'day-unit-rates'>

NightUnitRate = <RateType.NightUnitRate: 'night-unit-rates'>

enum Region(*value*)
Bases: enum_tools.custom_enums.StrEnum

Enumeration of different electricity supply regions.

The different regions can be seen on the following map:



Member Type `str`

Valid values are as follows:

```
Eastern = <Region.Eastern:  '_A'>

EastMidlands = <Region.EastMidlands:  '_B'>

London = <Region.London:  '_C'>

Merseyside = <Region.Merseyside:  '_D'>

Midlands = <Region.Midlands:  '_E'>

NorthEastern = <Region.NorthEastern:  '_F'>

NorthWestern = <Region.NorthWestern:  '_G'>

Southern = <Region.Southern:  '_H'>

SouthEastern = <Region.SouthEastern:  '_J'>

SouthWales = <Region.SouthWales:  '_K'>

SouthWestern = <Region.SouthWestern:  '_L'>

Yorkshire = <Region.Yorkshire:  '_M'>

SouthScotland = <Region.SouthScotland:  '_N'>
```

NorthScotland = <Region.NorthScotland: '_P'>

add_repr(*cls*)
Add a pretty-printed `__repr__` function to the decorated attrs class.

Parameters **cls** (Type)

See also:

attr_utils.pprinter.pretty_repr().

Return type Type

from_iso_zulu(*the_datetime*)
Constructs a datetime.datetime object from an ISO 8601 format string.

This function understands the character Z as meaning Zulu time (GMT/UTC).

Parameters **the_datetime** (Union[str, datetime, None])

Return type Optional[datetime]

Chapter

THREE

Contributing

3.1 Overview

`octo-api` uses tox to automate testing and packaging, and pre-commit to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```

3.2 Coding style

formate is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```

3.3 Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```

3.4 Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```

3.5 Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```

3.6 Downloading source code

The `octo-api` source code is available on GitHub, and can be accessed from the following URL: https://github.com/domdfcoding/octo-api

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/octo-api
```

```
Cloning into 'octo-api'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a 'zip' file by clicking:
Clone or download –> Download Zip

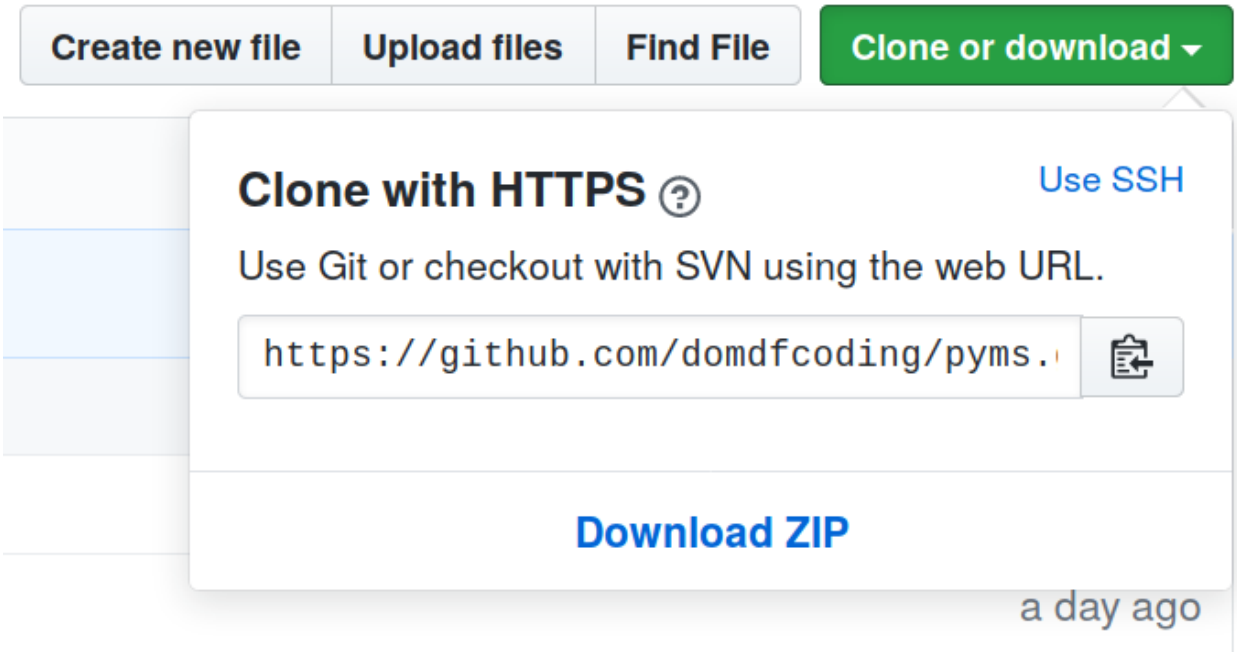


Fig. 1: Downloading a 'zip' file of the source code

3.6.1 Building from source

The recommended way to build `octo-api` is to use tox:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use pep517.build or another **PEP 517**-compatible build tool.

Parts of this documentation based on https://developer.octopus.energy/docs/api/

Python Module Index

o

Index

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

V

Y