

# Package ‘octopusR’

May 9, 2026

**Title** Interact with the 'Octopus Energy' API

**Version** 1.0.1

**Description** A simple wrapper for the 'Octopus Energy' API  
<<https://developer.octopus.energy/docs/api/>>. It handles authentication, by storing a provided API key and meter details. Implemented endpoints include 'products' for viewing tariff details and 'consumption' for viewing meter consumption data.

**License** MIT + file LICENSE

**URL** <https://github.com/Moohan/octopusR>,  
<https://moohan.github.io/octopusR/>

**BugReports** <https://github.com/Moohan/octopusR/issues>

**Imports** askpass, cli, glue, httr2, rlang, tibble

**Suggests** covr, lubridate, spelling, testthat

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** James McMahon [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-5380-2029>>)

**Maintainer** James McMahon <jamescmahon@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-06-09 15:10:07 UTC

## Contents

get_consumption . . . . .	2
get_meter_gsp . . . . .	3
get_products . . . . .	4
set_api_key . . . . .	5
set_meter_details . . . . .	5

## Index

7

---

get_consumption	<i>List consumption for a meter</i>
-----------------	-------------------------------------

---

**Description**

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<sup>3</sup>

**Parsing dates:**

To return dates properly parsed [lubridate](#) is required. Use the `tz` parameter to specify a time zone e.g. `tz = "UTC"`, the default (`tz = NULL`) will return the dates unparsed, as characters.

**Usage**

```
get_consumption(
  meter_type = c("electricity", "gas"),
  mpan_mprn = get_meter_details(meter_type)[["mpan_mprn"]],
  serial_number = get_meter_details(meter_type)[["serial_number"]],
  api_key = get_api_key(),
  period_from = NULL,
  period_to = NULL,
  tz = NULL,
  order_by = c("-period", "period"),
  group_by = c("hour", "day", "week", "month", "quarter")
)
```

**Arguments**

<code>meter_type</code>	Type of meter-point, electricity or gas
<code>mpan_mprn</code>	The electricity meter-point's MPAN or gas meter-point's MPRN.
<code>serial_number</code>	The meter's serial number.
<code>api_key</code>	Your API key. If you are an Octopus Energy customer, you can generate an API key on the <a href="#">developer dashboard</a> .
<code>period_from</code>	Show consumption from the given datetime (inclusive). This parameter can be provided on its own.
<code>period_to</code>	Show consumption to the given datetime (exclusive). This parameter also requires providing the <code>period_from</code> parameter to create a range.
<code>tz</code>	a character string that specifies which time zone to parse the date with. The string must be a time zone that is recognized by the user's OS.

order_by	Ordering of results returned. Default is that results are returned in reverse order from latest available figure. Valid values: <ul style="list-style-type: none"><li>• period, to give results ordered forward.</li><li>• -period, (default), to give results ordered from most recent backwards.</li></ul>
group_by	Aggregates consumption over a specified time period. A day is considered to start and end at midnight in the server's time zone. The default is that consumption is returned in half-hour periods. Accepted values are: <ul style="list-style-type: none"><li>• hour</li><li>• day</li><li>• week</li><li>• month</li><li>• quarter</li></ul>

**Value**

a [tibble](#) of the requested consumption data.

---

get_meter_gsp	<i>Get the GSP of a meter-point.</i>
---------------	--------------------------------------

---

**Description**

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

**Usage**

```
get_meter_gsp(mpan = get_meter_details("electricity")[["mpan_mprn"]])
```

**Arguments**

mpan	The electricity meter-point's MPAN
------	------------------------------------

**Value**

a character of the meter-points GSP.

---

get_products	<i>Return a list of energy products</i>
--------------	---

---

### Description

By default, results will be public energy products but if authenticated organisations will also see products available to their organisation.

### Usage

```
get_products(  
  is_variable = NULL,  
  is_green = NULL,  
  is_tracker = NULL,  
  is_prepay = NULL,  
  is_business = FALSE,  
  available_at = Sys.Date(),  
  authenticate = FALSE,  
  api_key = NULL  
)
```

### Arguments

is_variable	(boolean, optional) Show only variable products.
is_green	(boolean, optional) Show only green products.
is_tracker	(boolean, optional) Show only tracker products.
is_prepay	(boolean, optional) Show only pre-pay products.
is_business	(boolean, default: FALSE) Show only business products.
available_at	Show products available for new agreements on the given datetime. Defaults to current datetime, effectively showing products that are currently available.
authenticate	(boolean, default: FALSE) Use an api_key to authenticate. Only useful for organisations.
api_key	Your API key. If you are an Octopus Energy customer, you can generate an API key on the <a href="#">developer dashboard</a> .

### Value

a [tibble](#)

### Examples

```
get_products(is_green = TRUE)
```

---

set_api_key	<i>Set the Octopus API key</i>
-------------	--------------------------------

---

### Description

Set the Octopus API key to use. This will be stored as an environment variable. You should add `OCTOPUSR_API_KEY = <api_key>` to your `.Renvi ron` otherwise you will have to call this function every session.

### Usage

```
set_api_key(api_key = NULL)
```

### Arguments

api_key	Your API key. If you are an Octopus Energy customer, you can generate an API key on the <a href="#">developer dashboard</a> .
---------	---

### Value

No return value, called for side effects.

---

set_meter_details	<i>Set the details for your gas/electricity meter</i>
-------------------	---

---

### Description

Set the details for your gas/electricity meter. These will be stored as environment variables. You should add:

- `OCTOPUSR_MPAN = <electric MPAN>`
- `OCTOPUSR_MPRN = <gas MPRN>`
- `OCTOPUSR_ELEC_SERIAL_NUM = <electric serial number>`
- `OCTOPUSR_GAS_SERIAL_NUM = <gas serial number>` to your `.Renvi ron` otherwise you will have to call this function every session. You can find your meter details (MPAN/MPRN and serial number(s)) on the [developer dashboard](#).

### Usage

```
set_meter_details(  
  meter_type = c("electricity", "gas"),  
  mpan_mprn = NULL,  
  serial_number = NULL  
)
```

**Arguments**

<code>meter_type</code>	Type of meter-point, electricity or gas
<code>mpan_mprn</code>	The electricity meter-point's MPAN or gas meter-point's MPRN.
<code>serial_number</code>	The meter's serial number.

**Value**

No return value, called for side effects.

# Index

`get_consumption`, 2  
`get_meter_gsp`, 3  
`get_products`, 4  
  
`lubridate`, 2  
  
`set_api_key`, 5  
`set_meter_details`, 5  
  
`tibble`, 3, 4