Package ‘BMRSr’

September 3, 2019

Type Package
Title Wrapper Functions to the ‘BMRS API’
Version 1.0.0
Description A set of wrapper functions to better interact with the ‘Balancing Mechanism Reporting System API’.
License GPL (>= 2)
Encoding UTF-8
LazyData true
Depends R (>= 2.10)
Imports httr, xml2, stringr, tibble, readr, methods
RoxygenNote 6.1.1
URL https://github.com/ARawles/BMRSr
Suggests covr, knitr, rmarkdown, ggplot2, dplyr, tidyr, testthat
VignetteBuilder knitr
NeedsCompilation no
Author Adam Rawles [aut, cre]
Maintainer Adam Rawles <adamrawles@hotmail.co.uk>
Repository CRAN
Date/Publication 2019-09-03 07:30:13 UTC

R topics documented:

  build_b_call ....................................................... 2
  build_call .......................................................... 3
  build_legacy_call .................................................. 4
  build_remit_call ................................................... 5
  check_data_item .................................................... 7
  clean_date_columns ................................................ 7
  full_request ........................................................ 8
  generation_dataset_example ........................................ 9
build_b_call

Create an API call for B-data flows

Description
Create an API call for B-data flows

Usage
build_b_call(data_item, api_key, settlement_date = NULL, period = NULL,
year = NULL, month = NULL, week = NULL, process_type = NULL,
start_time = NULL, end_time = NULL, start_date = NULL,
end_date = NULL, service_type = "csv", api_version = "v1")

Arguments
- data_item: character string; the id of the B flow
- api_key: character string; api key retrieved from the Elexon portal
- settlement_date: character string; settlement date (automatically cleaned by format_date)
- period: character string; settlement period
- year: character string; year
- month: character string; month
- week: character string; week
- process_type: character string; process type
- start_time: character string; start time
- end_time: character string; end time
- start_date: character string; start date
- end_date: character string; end date
- service_type: character string; file format (csv or xml)
- api_version: character string; version of the api to use (currently on v1)

Value
list; created url for the call, service type and data item
build_call

See Also

Other call-building functions: build_call, build_legacy_call, build_remit_call

Examples

build_b_call(data_item = "B1730", api_key = "12345", settlement_date = "14-12-2016")
build_b_call(data_item = "B1510", api_key = "12345", start_date = "01 Jan 2019",
start_time = "00:00:00", end_date = "02 Jan 2019", end_time = "24:00:00", service_type = "csv")

build_call

Build an API call (uses the appropriate function based on the data item)

Description

Build an API call (uses the appropriate function based on the data item)

Usage

build_call(data_item, api_key, service_type = "csv",
api_version = "v1", ...)

Arguments

data_item character string; data item to be retrieved
api_key character string; user's API key
service_type character string; one of "csv" or "xml" to define return format
api_version character string; API version to use - currently only on version 1
... values to be passed to appropriate build_x_call function

See Also

build_b_call
build_remit_call
build_legacy_call

Other call-building functions: build_b_call, build_legacy_call, build_remit_call

Examples

build_call(data_item = "TEMP", api_key = "12345", from_date = "12 Jun 2018",
to_date = "13 Jun 2018", service_type = "csv")
build_call(data_item = "QAS", api_key = "12345",
settlement_date = "01 Jun 2019", service_type = "xml")
build_legacy_call  

Create an API call for legacy data

**Description**

Create an API call for legacy data

**Usage**

```r
build_legacy_call(data_item, api_key, from_date = NULL, to_date = NULL,
                  settlement_date = NULL, settlement_period = NULL,
                  bm_unit_id = NULL, bm_unit_type = NULL, lead_party_name = NULL,
                  ngc_bm_unit_name = NULL, from_cleared_date = NULL,
                  to_cleared_date = NULL, is_two_day_window = NULL,
                  from_datetime = NULL, to_datetime = NULL,
                  from_settlement_date = NULL, to_settlement_date = NULL,
                  period = NULL, fuel_type = NULL, balancing_service_volume = NULL,
                  zone_identifier = NULL, start_time = NULL, end_time = NULL,
                  trade_name = NULL, trade_type = NULL, api_version = "v1",
                  service_type = "csv")
```

**Arguments**

- `data_item` character string; the id of the legacy data
- `api_key` character string; api key retrieved from the Elexon portal
- `from_date` character string; from date (automatically cleaned by format_date)
- `to_date` character string; to date (automatically cleaned by format_date)
- `settlement_date` character string; settlement date (automatically cleaned by format_date)
- `settlement_period` character string; settlement period
- `bm_unit_id` character string; BM Unit ID
- `bm_unit_type` character string; BM Unit type
- `lead_party_name` character string; lead party name
- `ngc_bm_unit_name` character string; NGC BM Unit name
- `from_cleared_date` character string; from cleared date (automatically cleaned by format_date)
- `to_cleared_date` character string; to cleared date (automatically cleaned by format_date)
- `is_two_day_window` character string; is two day window
- `from_datetime` character string; from datetime
- `to_datetime` character string; to datetime
- `from_settlement_date` character string; from settlement date
- `to_settlement_date` character string; to settlement date
- `period` character string; period
- `fuel_type` character string; fuel type
- `balancing_service_volume` character string; balancing service volume
- `zone_identifier` character string; zone identifier
- `start_time` character string; start time
- `end_time` character string; end time
- `trade_name` character string; trade name
- `trade_type` character string; trade type
- `api_version` character string; api version (default v1)
- `service_type` character string; service type (default csv)
**build_remit_call**

- **to_datetime** character string; to datetime
- **from_settlement_date** character string; from settlement date (automatically cleaned by format_date)
- **to_settlement_date** character string; to settlement date (automatically cleaned by format_date)
- **period** character string; period
- **fuel_type** character string; fuel type
- **balancing_service_volume** character string; balancing service volume
- **zone_identifier** character string; zone identifier
- **start_time** character string; start time
- **end_time** character string; end time
- **trade_name** character string; trade name
- **trade_type** character string; trade type
- **api_version** character string; version of the api to use (currently on v1)
- **service_type** character string; file format (csv or xml)

**Value**

list; created url for the call, service type and data item

**See Also**

Other call-building functions: build_b_call, build_call, build_remit_call

**Examples**

```r
build_legacy_call(data_item = "FUELINST", api_key = "12345", from_datetime = "14-12-201613:00:00", to_datetime = "14-12-201614:00:00")
buildLegacyCall(data_item = "QAS", api_key = "12345", settlement_date = "01 Jun 2019", service_type = "xml")
```

---

**build_remit_call** *Create an API call for REMIT flows*

**Description**

Create an API call for REMIT flows
Usage

build_remit_call(data_item, api_key, event_start = NULL, event_end = NULL, publication_from = NULL, publication_to = NULL, participant_id = NULL, asset_id = NULL, event_type = NULL, fuel_type = NULL, message_type = NULL, message_id = NULL, unavailability_type = NULL, active_flag = NULL, sequence_id = NULL, service_type = "xml", api_version = "v1")

Arguments

data_item character string; the id of the REMIT flow
api_key character string; api key retrieved from the Elexon portal
event_start character string; event start (automatically cleaned by format_date)
event_end character string; event end (automatically cleaned by format_date)
publication_from character string; publication from (automatically cleaned by format_date)
publication_to character string; publication to (automatically cleaned by format_date)
participant_id character string; participant id
asset_id character string; asset id
event_type character string; event type
fuel_type character string; fuel type
message_type character string; message type
message_id character string; message id
unavailability_type character string; unavailability type
active_flag character string; active flag
sequence_id character string; sequence id
service_type character string; file format (csv or xml)
api_version character string; version of the api to use (currently on v1)

Value

list; created url for the call, service type and data item

See Also

Other call-building functions: build_b_call, build_call, build_legacy_call

Examples

build_remit_call(data_item = "MessageListRetrieval", api_key = "12345", event_start = "14-12-2016", event_end = "15-12-2016")
build_remit_call(data_item = "MessageDetailRetrieval", api_key = "12345", participant_id = 21, service_type = "xml")
check_data_item

Check the data item to ensure that it is a valid request

Description
Check the data item to ensure that it is a valid request

Usage
check_data_item(data_item, type)

Arguments
- data_item: character; the data item to check
- type: character; the type of data_item - one of "B Flow", "Legacy", or "REMIT"

Value
boolean: returns true if data_item is valid, false if it is not

Examples
check_data_item("B1720", "B Flow") #valid
check_data_item("B1720", "Legacy") #invalid - incorrect type
check_data_item("B1111", "REMIT") #invalid - incorrect data item and type

clean_date_columns
Reformat date, time, and datetime columns

Description
Reformat date, time, and datetime columns

Usage
clean_date_columns(x)

Arguments
- x: tibble/df; dataset with the columns to be formatted

Value
tibble/df; dataset with reformatted columns (if any needed reformatting)
Examples

generation_dataset_unclean <- as.data.frame(
apply(generation_dataset_example, 2, as.character)
)  # Create a version of the example generation dataset with character columns
clean_date_columns(generation_dataset_unclean)

full_request

Create an API call, send the request and retrieve the results, and parse them

Description

Create an API call, send the request and retrieve the results, and parse them

Usage

full_request(..., get_params = list(), parse = TRUE,
clean_dates = TRUE)

Arguments

... values to be passed to appropriate build_x_call function
get_params list; parameters to be passed to the send_request function (which will pass those parameters to httr::get)
parse boolean; whether the results should be parsed or returned as a response() object
clean_dates boolean; whether the csv response columns should be cleaned (reformatted to be correct date/time/datetime)

Value

If parse == TRUE, a tibble if service_type = "csv", otherwise a list. If parse == FALSE, a response() object is returned

Examples

full_request(data_item = "B1730", api_key = "12345",
settlement_date = "14-12-2016", parse = TRUE, service_type = "xml")
An example dataset from BMRS showing generation by fuel type.

Description
A dataset containing UK generation by fuel type between 1 July 2019 and 3 July 2019 at half-hourly intervals.

Usage
generation_dataset_example

Format
A data frame with 8655 rows and 6 variables:

- **record_type** data item
- **settlement_date** Settlement Date of the observation
- **settlement_period** Settlement Period of the observation
- **spot_time** Spot Time of the observation; this is essentially an amalgamation of settlement_date and settlement_period
- **ccgt** Generation from Combined Cycle Gas Turbines (MW)
- **oil** Generation from oil (MW)
- **coal** Generation from coal (MW)
- **nuclear** Generation from nuclear (MW)
- **wind** Generation from wind (MW)
- **ps** Generation from pumped storage (MW)
- **npshyd** Generation from hydro (non-pump storage; MW)
- **ocgt** Generation from Open Cycle Gas Turbines (MW)
- **other** Generation from other, not-listed sources (MW)
- **infr** Generation from the French interconnector (MW)
- **intirl** Generation from the Northern Irish interconnector (MW)
- **intned** Generation from the Dutch interconnector (MW)
- **intew** Generation from the Irish interconnector (MW)
- **biomass** Generation from biomass (MW)
- **intnem** Generation from Belgian interconnector (MW)

Source
https://www.bmreports.com/bmrs/?q-help/about-us
get_column_names  *Get the column names for a returned csv dataset*

**Description**

Get the column names for a returned csv dataset

**Usage**

get_column_names(data_item)

**Arguments**

- data_item  
  character string; data item for the dataset

**Value**

vector; a vector of character strings with the column headings

**Examples**

get_column_names("TEMP")

get_data_items  *Get a vector containing all of the permissible data items*

**Description**

Get a vector containing all of the permissible data items

**Usage**

get_data_items()

**Value**

vector; data items as character string

**Examples**

generate_data()}
**get_data_item_type**

Get the data item type of a data item

**Usage**

`get_data_item_type(data_item)`

**Arguments**

data_item character string; data item to be retrieved

**Examples**

`get_data_item_type("TEMP")`

---

**get_function**

Get the correct function to create the API call depending on the data item

**Usage**

`get_function(data_item)`

**Arguments**

data_item character string; data item to be retrieved

**Value**

function

**Examples**

`get_function("TEMP")`
get_parameters  
*Get the required parameters for a data item*

**Description**
Get the required parameters for a data item

**Usage**
```r
get_parameters(data_item)
```

**Arguments**
- `data_item` character; the data item to get the parameters for

**Value**
A list containing the named parameters required for that call

**Examples**
```r
get_parameters("TEMP")
```

parse_response  
*Parse the results of a call*

**Description**
Parse the results of a call

**Usage**
```r
parse_response(response, format, clean_dates = TRUE)
```

**Arguments**
- `response` A response object returned from the API request
- `format` character string; format of the content of the response object; either "csv" or "xml"
- `clean_dates` boolean; whether to clean date/time columns

**Value**
A tibble if format == "csv", otherwise a list
Examples

```r
list_example <- parse_response(
    send_request(
        build_call("TEMP", api_key = "12345", from_date = "01 Jun 2019",
                    to_date = "10 Jun 2019", service_type = "xml"), "xml")
)
```

Description

Send an API request (basically a wrapper to `httr::GET` that adds a marker for the data item)

Usage

```r
send_request(request, config_options = list())
```

Arguments

- `request` list; a named list with at least a url to be sent and the data item contained within (most easily generated from `build_call()`)
- `config_options` list; a named list of config options to be passed to `httr::GET`

Value

A `response()` object with an added `data_item` attribute

Examples

```r
send_request(
    build_call(data_item = "TEMP", from_date = "01 Jun 2019", to_date = "10 Jun 2019", api_key = "test")
)
```
Index

*Topic **datasets**
  - generation.dataset_example, 9

build.b.call, 2, 3, 5, 6
build.call, 3, 3, 5, 6
build_legacy.call, 3, 4, 6
build_remit.call, 3, 5, 5

check_data_item, 7
clean_date_columns, 7

full_request, 8

generation.dataset_example, 9
get_column_names, 10
get_data_item_type, 11
get_data_items, 10
get_function, 11
get_parameters, 12

parse_response, 12

send_request, 13