Package 'OnboardClient'

January 20, 2025

Type Package

Title Bindings for Onboard Data's Building Data API

Version 1.0.0

Description Provides a wrapper for the Onboard Data building data API <https://api.onboarddata.io/swagger>. Along with streamlining access to the API, this package simplifies access to sensor time series data, metadata (sensors, equipment, and buildings), and details about the Onboard data model/ontology.

License Apache License (>= 2)

Imports data.table (>= 1.14.2), dplyr (>= 1.0.10), httr (>= 1.4.4), jsonlite (>= 1.8.0), lubridate (>= 1.8.0), plyr (>= 1.8.7), rrapply (>= 1.2.5), rstudioapi (>= 0.14), tibble (>= 3.1.8), tidyr (>= 1.2.1), tidyselect (>= 1.1.2), stringr (>= 1.4.1)

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Pranay Shah [aut], Christopher Dudas-Thomas [cre, aut], Onboard Data [cph, fnd]

Maintainer Christopher Dudas-Thomas <christopher@onboarddata.io>

Repository CRAN

Date/Publication 2022-10-29 09:14:31 UTC

Contents

`api.access` *Access API keys and URL from System Environment*

Description

Returns the API url and API key.

Usage

```
api.access()
```

Value

A named list of API information, containing elements 'url' and 'key'.

`api.get` *API GET call*

Description

Uses http GET call to return an object from the API.

Usage

```
api.get(endpoint)
```

Arguments

`endpoint`	A character string containing a valid Onboard API endpoint.

Value

A list or data.frame of the API output.

Examples

```
## Not run:  whoami <- api.get('whoami')
```

---

api.post *API POST call*

Description

Uses http POST call to post objects to the API.

Usage

```
api.post(endpoint, json_body, output = "list")
```

Arguments

endpoint	A character string containing a valid Onboard API endpoint.
json_body	A JSON payload to give to the POST call.
output	A character string, either "list" (default) or "dataframe", to specify the API output format.

Value

A list or data.frame of the API output.

---

api.setup *Set up Onboard API keys and URL in system environment*

Description

Set the Onboard API URL and API keys in the system environment.

Usage

```
api.setup(api_type = "prod")
```

Arguments

api_type	Provide the API client name.

Value

No return value, sets API url and API key in the system environment.

api.status *Check the status of your connection with the Onboard API*

Description

Provides a status code and message for the API connection.

Usage

```
api.status()
```

Value

A character string of the API server status and message.

get_buildings *Buildings*

Description

Retrieve buildings that you have access to.

Usage

```
get_buildings(id)
```

Arguments

id	(Optional) An integer if you want information on a particular entity. Returns all entities unless this argument is provided.

Value

A data.frame of all building information.

get_building_info *Building Info*

Description

Retrieves building id(s) and name(s). Assigns each to list variables in the parent environment called "id" and "name", and prints each list.

Usage

```
get_building_info(buildings, verbose = TRUE)
```

Arguments

buildings	Integer, character, or vectors of those types, providing building id(s) or name(s). You can provide multiple buildings at once.
verbose	Logical. If TRUE (default), print status messages.

Value

A data.frame of building info with two columns, 'id' and 'name'.

get_deployments *Deployments*

Description

Get all deployments in your organization.

Usage

```
get_deployments(org_id)
```

Arguments

org_id	organization id

Value

A data.frame of all deployments.

get_equipment_by_ids *Equipment by ID*

Description

Queries equipment by their ids.

Usage

```
get_equipment_by_ids(id)
```

Arguments

id	Integer or integer vector, containing one or many equipment ids.

Value

A data.frame of the requested equipment, or an empty list if no equipment matches those ids.

Examples

```
## Not run:
equipment <- get_equipment_by_ids(c(1000,1001))

# If you are using the point selector function:
query <- PointSelector()

query$buildings <- 101
query$equipment_types <- 'ahu'

selection <- select_points(query)

equipment <- get_equipment_by_ids(selection$equipment)

## End(Not run)
```

get_equip_types *Equipment Types*

Description

Query all equipment types from Onboard's Data Model.

Usage

```
get_equip_types()
```

Value

A data.frame containing all equipment types.

`get_metadata` *Metadata*

Description

Retrieves points and equipment for a given building or selection and outputs a clean metadata data.frame.

Usage

```
get_metadata(buildings, selection, verbose = TRUE)
```

Arguments

`buildings`	Integer, character, or vectors of those types, providing building id(s) or name(s). You can provide multiple buildings at once.
`selection`	Selection list from point selector.
`verbose`	Logical. If TRUE (default), print status messages.

Value

A data.frame of clean metadata for the requested points.

Examples

```
## Not run:
metadata <- get_metadata(buildings=c(427,"Laboratory"))

OR

query <- PointSelector()

query$buildings <- 427
query$equipment_types <- 'ahu'
query$point_types <- c('Supply Air Temperature','Supply Air Static Pressure')

selection <- select_points(query)

metadata <- get_metadata(selection)

## End(Not run)
```

get_orgs *Organizations*

Description

Retrieve Organizations that you have access to.

Usage

```
get_orgs(id)
```

Arguments

id	(Optional) An integer if you want information on a particular entity. Returns all entities unless this argument is provided.

Value

A data.frame of organization information.

get_points_by_ids *Points by ID*

Description

Queries data points by their ids.

Usage

```
get_points_by_ids(id)
```

Arguments

id	Integer or list of integers. One or many point ids.

Value

A data.frame of the requested points, or an empty list if there are no points with those ids.

Examples

```
## Not run:
points <- get_points_by_ids(c(10000,10001))

# If you are using the point selector function:
query <- PointSelector()

query$buildings <- 101
query$equipment_types <- 'ahu'
query$point_types <- c('Supply Air Temperature','Supply Air Static Pressure')

selection <- select_points(query)

points <- get_points_by_ids(selection$points)

## End(Not run)
```

---

get_point_types *Point Types*

Description

Queries all point types, measurements and their units from Onboard's Data Model and returns a clean output.

Usage

```
get_point_types()
```

Value

A data.frame containing all point types.

---

get_staged_data *Get Staged Data*

Description

Gets metadata from the staging area.

Usage

```
get_staged_data(building, verbose = TRUE)
```

Arguments

building	Character vector or integer corresponding to the building name or id. If you enter multiple building ids or names, only the first entry is considered.
verbose	Logical. If TRUE (default), prints status and progress messages.

Value

A data.frame of metadata from the staging area.

---

get_timeseries *Time-Series Data*

Description

Provides clean time-series

Usage

```
get_timeseries(start_time, end_time, point_ids)
```

Arguments

start_time	Start Time in UTC.
end_time	End Time in UTC.
point_ids	Point IDs for which timeseries data needs to be queried.

Value

A wide data.frame of time-series data, with timestamp and all requested point IDs as columns.

---

get_timeseries_raw *Raw Time-Series Data*

Description

Retrieves timeseries data in raw format.

Usage

```
get_timeseries_raw(start_time, end_time, point_ids)
```

Arguments

start_time	Start Time in UTC.
end_time	End Time in UTC.
point_ids	Point IDs for which timeseries data needs to be queried.

Value

A long data.frame of time series data, with point id, timestamp, and raw point values as columns.

`get_users` *Users*

Description

Retrieve all user info in your organization.

Usage

```
get_users(id)
```

Arguments

`id`	(Optional) An integer if you want information on a particular entity. Returns all entities unless this argument is provided.

Value

A data.frame of all user information.

`PointSelector` *PointSelector*

Description

A list of parameters to query metadata.

Usage

```
PointSelector()
```

Value

An empty named list of possible point selection criteria.

Examples

```
## Not run:
query <- PointSelector()

query$buildings <- 101
query$equipment_types <- 'ahu'
query$point_types <- c('Supply Air Temperature','Supply Air Static Pressure')

## End(Not run)
```

promote_staged_data *Promote data on Staging Area*

Description

Promote valid data on the staging area to the live building.

Usage

```
promote_staged_data(building, data_to_promote, verbose = TRUE)
```

Arguments

building	Character vector or integer corresponding to the building name or id. If you enter multiple building ids or names, only the first entry is considered.
data_to_promote	(Optional) If missing, all valid topics are promoted. A data.frame containing columns 'e.equip_id' & 'p.topic'.
verbose	Logical. If TRUE (default), prints status and progress messages.

Value

A named list containing any errors that may have occured during data promotion.

select_points *Select Points*

Description

Returns a list of ids based on the input query from PointSelector. Uses http POST call to query data.

Usage

```
select_points(query)
```

Arguments

query	query supplied from PointSelector.

Value

A named list of all the points requested by the query.

Examples

```
## Not run:
query <- PointSelector()

query$buildings <- 427
query$equipment_types <- 'ahu'
query$point_types <- c('Supply Air Temperature','Supply Air Static Pressure')

selection <- select_points(query)

## End(Not run)
```

---

upload_staging *Upload to Staging Area*

Description

Uploads data to the staging area.

Usage

```
upload_staging(building, data_to_upload, skip_topics = FALSE, verbose = TRUE)
```

Arguments

building	Character vector or integer corresponding to the building name or id. If you enter multiple building ids or names, only the first entry is considered.
data_to_upload	A data.frame to upload to the staging area. Must contain e.equip_id and p.topic columns.
skip_topics	Logical. If True, the uploaded topics will be assigned '__SKIP__' equip_id.
verbose	Logical. If TRUE (default), prints status and progress messages.

Value

A named list containing any errors that may have occured during data upload.

Index