Package ‘aiRly’

October 12, 2022

Type Package
Title R Wrapper for 'Airly' API
Version 0.1.0
Maintainer Piotr Janus <piotr_janus@icloud.com>
Description Get information about air quality using 'Airly' <https://airly.eu/> API through R.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Imports utils, httr, jsonlite, reshape2, tibble
URL https://github.com/piotrekjanus/aiRly
BugReports https://github.com/piotrekjanus/aiRly/issues
Language en-US
Suggests testthat, httptest, covr
NeedsCompilation no
Author Piotr Janus [cre, aut]
Repository CRAN
Date/Publication 2020-03-19 14:00:02 UTC

R topics documented:

.base_url ................................................................. 2
.get_apikey ............................................................. 3
.send_request ............................................................ 3
add_json_extension ...................................................... 4
add_path ................................................................. 4
assert ................................................................. 5
assert_apikey ........................................................... 5
assert_coordinates ...................................................... 5
.base_url

Return base url of Airly API v2

Usage

.base_url()
.get_apikey

*Get Airly apikey*

**Description**

Get apikey that was set by user

**Usage**

```python
.get_apikey()
```

**Value**

- apikey value of set api key

---

.send_request

*Sends a request to the specified url and retrieves it's content.*

**Description**

Sends a request to the specified url and retrieves it’s content.

**Usage**

```python
.send_request(request_url, apikey, query = NULL)
```

**Arguments**

- `request_url` url to be used
- `apikey` airly apikey
- `query` Default value is NULL. Optional argument if you want to add query to request

**Value**

- parsed content of the response object
add_json_extension | Adds the json extension to the given url

**Description**

Adds the json extension to the given url

**Usage**

```
add_json_extension(url)
```

**Arguments**

- **url**: base url to which the json extension should be added

**Value**

- url with the json extension added

---

add_path | Adds the given path to the given url

**Description**

Adds the given path to the given url

**Usage**

```
add_path(url, path)
```

**Arguments**

- **url**: base url to which the path should be added
- **path**: path that should be added to the url

**Value**

- url with the given path added
assert

**Description**

Asserts a given expression and throws an error if it returns FALSE

**Usage**

```
assert(expression, error)
```

**Arguments**

- **expression**: R expression to be evaluated
- **error**: message to be displayed when the expression is not fulfilled

---

assert_apikey

**Description**

Checks whether apikey is correctly set

**Usage**

```
assert_apikey(key)
```

**Arguments**

- **key**: airly apikey

---

assert_coordinates

**Description**

Checks whether apikey is correctly set

**Usage**

```
assert_coordinates(lat, lng)
```

**Arguments**

- **lat**: latitude as decimal degree
- **lng**: longitude as decimal degree
### assert_ids

**Checks whether ids are correctly defined. If not throws an error**

**Description**

Checks whether ids are correctly defined. If not throws an error

**Usage**

```python
assert_ids(ids)
```

**Arguments**

- `ids` maximum number of ids to retrieve

---

### build_current_df

**Creates an object representing Airly measurement**

**Description**

Creates an object representing Airly measurement

**Usage**

```python
build_current_df(item)
```

**Arguments**

- `item` list returned by Airly API

**Value**

object representing a airly_measurement
**build_forecast_df**

<table>
<thead>
<tr>
<th>Description</th>
<th>Creates object containing information about history data for given API response</th>
</tr>
</thead>
</table>

**Usage**

`build_forecast_df(item)`

**Arguments**

- **item** list returned by Airly API

**Value**

tibble representing a `airly_measurement` with time, measures and indexes fields

---

**build_history_df**

<table>
<thead>
<tr>
<th>Description</th>
<th>Creates object containing information about history data for given API response</th>
</tr>
</thead>
</table>

**Usage**

`build_history_df(item)`

**Arguments**

- **item** list returned by Airly API

**Value**

tibble representing a `airly_measurement` with time, measures and indexes fields
create_airly_api_response

*Create an object representing a response from the Airly API. Also every API call return information about current limits What is used to assign variables in pkg.env*

**Description**

Creates an object representing a response from the Airly API. Also every API call return information about current limits What is used to assign variables in pkg.env

**Usage**

```r
create_airly_api_response(response)
```

**Arguments**

- `response`: response object

**Value**

- object representing a response from the Airly API

create_airly_location

*Creates an object representing Airly location*

**Description**

Creates an object representing Airly location

**Usage**

```r
create_airly_location(item)
```

**Arguments**

- `item`: list returned by Airly API

**Value**

- tibble representing an airly_location
**create_airly_measurement**

*Creates an object representing Airly measurement*

**Description**

Creates an object representing Airly measurement

**Usage**

```r
create_airly_measurement(item)
```

**Arguments**

- `item` : list returned by Airly API

**Value**

object representing a `airly_measurement`

---

**create_airly_meta**

*Creates a data.frame representing Airly meta*

**Description**

Creates a data.frame representing Airly meta

**Usage**

```r
create_airly_meta(item)
```

**Arguments**

- `item` : list returned by Airly API

**Value**

data.frame representing an `airly_meta`
create_request_url

Creates a request url based on the given base url and passed paths. The json extensions is added automatically.

Description

Creates a request url based on the given base url and passed paths. The json extensions is added automatically.

Usage

create_request_url(url, paths, add_json_ext = TRUE)

Arguments

url base url of the request
paths vector of paths that should be added to the url
add_json_ext boolean indicating if include "json" at the end of request

Value

request url with added paths and the json extension

get_content

Retrieves the response content

Description

Retrieves the response content

Usage

get_content(x)

Arguments

x airly_api_response object to retrieve content from

Value

content of the given airly_api_response object
### get_indexes

**Get Airly available indexes**

**Description**

Endpoint returns a list of all the index types supported in the API along with lists of levels defined per each index type.

**Usage**

```python
get_indexes()
```

**Value**

object of airly_meta class

**Examples**

```python
get_indexes()
```

### get_installation_by_id

**Get Airly installation by id**

**Description**

Endpoint returns single installation metadata, given by id

**Usage**

```python
get_installation_by_id(id)
```

**Arguments**

- **id**
  - integer

**Value**

airly_location item

**Examples**

```python
get_installation_by_id(2137)
```
get_installation_measurements

Get Airly measurements for any geographical location given installation id

Description

Endpoint returns measurements for concrete installation given by installation Id

Usage

get_installation_measurements(id)

Arguments

id integer, installation identifier

Value

object of airly_measurements class

Examples

get_installation_measurements(8077)

get_measurements_info

Get measures used in Airly

Description

Endpoint returns list of all the measurement types supported in the API along with their names and units.

Usage

get_measurements_info()

Value

data.frame with measure names and units
get_nearest_installations

Examples

get_measurements_info()

get_nearest_installations

Get Airly nearest installations to given point

Description

Endpoint returns list of installations which are closest to a given point, sorted by distance to that point.

Usage

get_nearest_installations(lat, lng, max_distance = NULL, max_results = NULL)

Arguments

lat latitude as decimal degree
lng longitude as decimal degree
max_distance default value 3.0. All the returned installations must be located within this limit from the given point (in km). Negative value means no limit
max_results default value 1. Maximum number of installations to return. Negative value means no limit

Value

data.frame of airly_location items

Examples

get_nearest_installations(50.11670, 19.91429, max_distance = 20)
get_nearest_measurements

Get Airly nearest measurements to given point

Description
Endpoint returns measurements for an installation closest to a given location

Usage
get_nearest_measurements(lat, lng, max_distance = NULL)

Arguments
- lat: latitude as decimal degree
- lng: longitude as decimal degree
- max_distance: default value 3.0. All the returned installations must be located within this limit from the given point (in km). Negative value means no limit

Value
data.frame of airly_measurements items

Examples
get_nearest_measurements(50.11670, 19.91429, max_distance = 10)

get_point_measurements

Get Airly measurements for any geographical location

Description
Endpoint returns measurements for any geographical location

Usage
get_point_measurements(lat, lng)

Arguments
- lat: latitude as decimal degree
- lng: longitude as decimal degree
**is_airly_api_response**  

**Value**  
object of airly_measurements class  

**Examples**  

get_point_measurements(50.11670, 19.91429)

---

**is_airly_api_response** Checks whether the given object is of the class airly_api_response

**Description**  
Checks whether the given object is of the class airly_api_response

**Usage**  
is_airly_api_response(x)

**Arguments**  
x object to test if it is of the class airly_api_response

**Value**  
TRUE if the object is of the class airly_api_response

---

**is_airly_location** Checks whether the given object is of the class airly_location

**Description**  
Checks whether the given object is of the class airly_location

**Usage**  
is_airly_location(x)

**Arguments**  
x object to test if it is of the class airly_location

**Value**  
TRUE if the object is of the class airly_location
is_airly_measurement  Checks whether the given object is of the class airly_measurement

Description
Checks whether the given object is of the class airly_measurement

Usage
is_airly_measurement(x)

Arguments
x  object to test if it is of the class airly_measurement

Value
TRUE if the object is of the class airly_measurement

parse_json  Parses a json response

Description
Parses a json response

Usage
parse_json(response)

Arguments
response  response object to parse

Value
parsed content of the given response
print.airly_measurement

Print for "airly_measurement" type objects

Description

Print for "airly_measurement" type objects

Usage

## S3 method for class 'airly_measurement'
print(x, ...)

Arguments

x "airly_measurement" type list

... further arguments passed to or from other methods

remaining_requests Get information about remaining API requests

Description

Default rate limit per apikey is 100 API requests per day for all users. In order to get information, user has to make at least one request.

Usage

remaining_requests()

Value

list containing information about remaining requests and daily limit

Examples

# Make any request before calling this function
remaining_requests()
**replace_null**

*Replaces NULL with NA for nested lists. Useful when NULL value leads to error while object casting*

**Description**

Replaces NULL with NA for nested lists. Useful when NULL value leads to error while object casting

**Usage**

`replace_null(x)`

**Arguments**

- `x` nested list

**Value**

same list with NULL replaced with NA

---

**set_apikey**

*Set Airly apikey*

**Description**

On a free plan, API consumer is required to use our API only in non-commercial projects. More details are available in under [https://airly.eu/docs/tos-en.pdf](https://airly.eu/docs/tos-en.pdf).

**Usage**

`set_apikey(key)`

**Arguments**

- `key` string. Get your api key [https://developer.airly.eu/](https://developer.airly.eu/)

**Examples**

`set_apikey("abctest")`
validate_airly_api_response

Checks if the given response is not empty and that it did not return an error http code.

Description
Checks if the given response is not empty and that it did not return an error http code.

Usage
validate_airly_api_response(airly_api_response)

Arguments
airly_api_response
airly_api_response object to be checked

validate_airly_location
Checks whether the given object is correctly defined airly_location class

Description
Checks whether the given object is correctly defined airly_location class

Usage
validate_airly_location(airly_location)

Arguments
airly_location tibble airly_location
validate_airly_measurement

Checks whether the given object is correctly defined airly_measurement class

**Description**
Checks whether the given object is correctly defined airly_measurement class

**Usage**
validate_airly_measurement(airly_measurement)

**Arguments**
- **airly_measurement**
  object of the class airly_measurement

validate_airly_meta

Checks whether the given object is correctly correctly defined

**Description**
Checks whether the given object is correctly correctly defined

**Usage**
validate_airly_meta(airly_meta)

**Arguments**
- **airly_meta**
  object of the class airly_meta