Package: r311 (via r-universe)

January 30, 2025

Type Package

Title Interface to the 'open311' Standard

Version 0.4.0

Maintainer Jonas Lieth <jonas.lieth@gesis.org>

Description Access and handle APIs that use the international
'open311' 'GeoReport v2' standard for civic issue tracking
<https://wiki.open311.org/GeoReport_v2/>. Retrieve civic
service types and request data. Select and add available
'open311' endpoints and jurisdictions. Implicitly supports
custom queries and 'open311' extensions. Requires a minimal
number of hard dependencies while still allowing the
integration in common R formats ('xml2', 'tibble', 'sf').

License MIT + file LICENSE

URL https://github.com/jslth/r311, https://jslth.github.io/r311/,
https://github.com/JsLth/r311,
https://ropengov.github.io/r311/,
https://github.com/rOpenGov/r311

BugReports https://github.com/rOpenGov/r311/issues

Depends R (>= 4.0.0)

Imports curl, jsonlite, tools, utils

Suggests httptest2, knitr, rmarkdown, sf, testthat (>= 3.0.0), tibble,
xml2, xmlconvert

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

X-schema.org-isPartOf http://ropengov.org/

X-schema.org-keywords ropengov

Config/pak/sysreqs libssl-dev

Repository https://ropengov.r-universe.dev

RemoteUrl https://github.com/rOpenGov/r311

RemoteRef HEAD

RemoteSha 26e949e06239e9e82a19c29af655f25494992b72

Contents

`o311_add_endpoint` *Endpoints*

Description

Modify and examine defined open311 endpoints. `o311_endpoints()` retrieves a list of endpoints including additional information. `o311_add_endpoint` adds to this list to define a new endpoint that can be used for queries. `o311_reset_endpoints` restores the initial state of the endpoints list.

Usage

```
o311_add_endpoint(
  name,
  root,
  jurisdiction = NULL,
  key = FALSE,
  pagination = FALSE,
  limit = NULL,
  json = TRUE,
  dialect = NULL
)

o311_reset_endpoints()

o311_endpoints(...)
```

Arguments

`name`	`[character]` Name of an endpoint / city. This name can be arbitrary and only serves for identification in `[o311_api]`.
`root`	`[character]` Base URL of the endpoint for sending production-grade requests. The root URL commonly points to `"georeport/v2/"`.
`jurisdiction`	`[character]` Unique identifier of the jurisdiction. The jurisdiction is typically defined as the domain of the respective city website. It is optional as most endpoints only serve one jurisdiction.
`key`	`[logical]` Is an API key mandatory?
`pagination`	`[logical]` Are `requests` responses paginated?
`limit`	`[integer]` If paginated, how many requests does one page contain?
`json`	`[logical]` Are JSON responses supported? If `FALSE`, defaults to `"XML"` responses. See also `o311_api`.
`dialect`	`[character]` open311 extension that the endpoint is built on. Common dialects include CitySDK, Connected Bits, SeeClickFix and Mark-a-Spot. Currently, this argument does nothing, but it could be used in the future to adjust response handling based on dialect.
`...`	List of key-value pairs where each pair is a filter. The key represents the column and the value the requested column value. All keys must be present in the column names of `o311_endpoints()`.

Details

`o311_endpoints()` returns a static list defined in the package installation directory. This list contains a limited number of endpoints that were proven to work at the time of package development. It does not include newer/smaller/less known endpoints or test APIs. These can be manually added using `o311_add_endpoint`.

Value

For `o311_endpoints`, a dataframe containing all relevant information on an endpoint. For `o311_add_endpoint`, the new endpoint, invisibly. `o311_reset_endpoints` returns `NULL` invisibly. If the new endpoint is a duplicate, `NULL` is returned invisibly.

Note

This function uses `R_user_dir` to persistently store custom endpoints data between sessions. To set a different directory, you may use `options("o311_user_dir")`. To clean up, run `o311_reset_endpoints()` which deletes the package-specific user directory and defaults back to `system.file("endpoints.json", package = "r311")`.

See Also

`o311_api`

Examples

```
# read default endpoints
o311_endpoints()

# get all endpoints powered by Connected Bits
o311_endpoints(dialect = "Connected Bits")

# add a new endpoint
o311_add_endpoint(name = "test", root = "test.org/georeport/v2")

# read new endpoints
o311_endpoints()

# reset endpoints back to default
o311_reset_endpoints()
```

`o311_api` *Select an open311 API*

Description

Select an open311 API and attach it to the active session. An open311 API is an implementation of the open311 standard. It consists of an endpoint name (e.g. a city), a root URL, and a jurisdiction ID. To unambiguously identify an API, you can provide an endpoint, a jurisdiction ID, or both. The input is matched with `o311_endpoints` to select an API. The selected API is available to other `o311_*` functions until the session is terminated or until it is overwritten.

Usage

```
o311_api(
  endpoint = NULL,
  jurisdiction = NULL,
  key = NULL,
  format = c("json", "xml")
)
```

Arguments

`endpoint`	`[character]` Name of an endpoint that runs an open311 API. This is usually a city, but can be any provider of an open311 API.
`jurisdiction`	`[character]` ID of a jurisdiction that is served by an open311 API. A jurisdiction ID is usually the root URL of the jurisdiction website, e.g. `"sandiego.gov"` for San Diego.
`key`	`[character]` If a key is required by the selected API, this argument can be used to store the key in the R session. The API key is automatically used in API requests. If `key` is `NULL` although a key is required, a warning is emitted.
`format`	`[character]` Response format. Must be one of `"json"` or `"xml"`. Defaults to `"json"` because simplification is more difficult and unsafe for `xml2` objects. It is advisable to use `"json"` whenever possible and applicable. Additionally, `"xml"` requires the `xml2` package for queries and the `xmlconvert` package for simplification.

Details

In theory, several jurisdictions can exist for a single endpoints, e.g. if a region serves multiple jurisdictions. Similarly, multiple endpoints can exist for a single jurisdiction, e.g. if a provider has set up both production and test endpoints for a jurisdictions. Providing both endpoint and jurisdiction is thus the most safe way to identify an API.

By default, only a handful of endpoints are supported. For a list of currently supported endpoints, run `o311_endpoints`. You can add non-default endpoints using `o311_add_endpoint`.

Value

A list containing the most important information on a given jurisdiction, invisibly. This list is attached to the session and can be retrieved by calling `o311_api()` without arguments. Passing no arguments returns the currently attached API object.

See Also

`o311_requests`, `o311_request`, `o311_services`

Examples

```
# cities are matched using regex
o311_api("Cologne")

# passing a jurisdiction is more explicit
o311_api(jurisdiction = "stadt-koeln.de")

# calls without arguments return the current API
o311_api()
```

`o311_discovery` *API discovery*

Description

Retrieve discovery information about the mounted endpoint.

Usage

```
o311_discovery()
```

Value

A list containing details on the given open311 API.

Examples

```
o311_api("zurich")

if (o311_ok()) {
  o311_discovery()
}
```

`o311_ok` *Is open311 API ok?*

Description

Checks whether an open311 API mounted by `o311_api` is reachable and returns a valid requests response.

Usage

```
o311_ok(error = FALSE)
```

Arguments

`error`	`[logical]` Whether to return a logical or the error message describing why the API is not ok.

Value

A logical describing whether the API is reachable or not. If `error = TRUE`, returns the corresponding error object if one occurs.

Examples

```
# check if Bonn API is reachable
o311_api("Bonn")
o311_ok()

# check if Helsinki API is reachable - fails
o311_add_endpoint(
  name = "Helsinki",
  root = "asiointi.hel.fi/palautews/rest/v1/"
)

o311_api("Helsinki")
o311_ok()

# return error message
try(o311_ok(error = TRUE))

# reset endpoints database
o311_reset_endpoints()
```

`o311_query` *Query an open311 endpoint*

Description

Low-level function to perform a generic request to the API currently attached by `o311_api`. Some open311 implementations support unique operations that are not included in the official documentation. This function can be used to access these URL paths.

Usage

```
o311_query(path, ..., simplify = TRUE)
```

Arguments

`path`	Path appendix used to access endpoint-specific operations.
`...`	Additional query parameters.
`simplify`	Whether to simplify the output using `jsonlite::toJSON(..., simplify = TRUE)`.

Details

You can set `options(r311_echo = TRUE)` to display all requests sent using `o311_query`.

Value

The parsed query output, either as a list or dataframe.

Examples

```
o311_api("rostock")

# manually query discovery
o311_query(path = "discovery", simplify = FALSE)

# query a custom path defined by the Klarschiff API
o311_query(path = "areas")
```

---

o311_requests *Get civic service request data*

Description

Get civic service request data from a registered open311 endpoint. `o311_request` queries a single service request by ID. `o311_requests` queries a single page of service requests. `o311_request_all` tries to iterate through all pages of an endpoint to return a complete dataset of service requests.

Usage

```
o311_requests(
  service_code = NULL,
  start_date = NULL,
  end_date = NULL,
  status = NULL,
  page = NULL,
  ...
)

o311_request(service_request_id, ...)

o311_request_all(
  service_code = NULL,
  start_date = NULL,
  end_date = NULL,
  status = NULL,
  ...,
  max_pages = Inf,
  progress = TRUE
)
```

Arguments

`service_code`	`[character]` IDs of the service types to be queried. Defaults to all available codes of an endpoint. A list of all available service codes can be retrieved using `o311_services`.

`start_date`, `end_date`	`[POSIXt]` Start date and end date of the query results. Must be date-time objects. If not specified, defaults to the last 90 days.
`status`	`[character]` Status of the public service ticket. Can be one of `"open"` or `"closed"`. If `NULL`, returns all types of tickets.
`page`	`[integer]` Page of the response. Most endpoints paginate their responses in a way that only a limited number of tickets are returned with each query. To retrieve all data, consider using `o311_request_all`.
`...`	Further endpoint-specific parameters as documented in the respective endpoint reference.
`service_request_id`	`[character]` Identifier of a single service request. Request IDs can usually be retrieved from `o311_requests`.
`max_pages`	`[integer]` Number of pages to search until the result is returned.
`progress`	`[logical]` Whether to show a waiter indicating the current page iteration.

Details

`o311_request_all` applies a number of checks to determine when to stop searching. First, many endpoints return an error if the last page is exceeded. Thus, if the last page request failed, break. Second, if exceeding the pagination limit does not return an error, the response is compared with the previous response. If identical, the response is discarded and all previous responses returned. Finally, if the page exceeds `max_pages`, the responses up to this point are returned.

open311 leaves space for endpoints to implement their own request parameters. These parameters can be provided using dot arguments. These arguments are not validated or pre-processed. Date-time objects must be formatted according to the w3c standard. Some more common parameters include:

- `q`: Perform a text search across all requests.
- `update_after`/`updated_before`: Limit request according to request update dates.
- `per_page`: Specifiy the maximum number of requests per page.
- `extensions`: Adds a nested attribute `"extended_attributes"` to the response.
- `long`/`lat`/`radius`: Searches for requests in a fixed radius around a coordinate.

As dot arguments deviate from the open311 standard, they are not guaranteed to be available for every endpoint and might be removed without further notice. Refer to the endpoint docs to learn more about custom parameters (`o311_endpoints()$docs`).

Value

A dataframe containing data on civic service requests. The dataframe can contain varying columns depending on the open311 implementation.

See Also

o311_api

Examples

```
o311_api("zurich")

if (o311_ok()) {
  # retrieve requests from the last two days
  now <- Sys.time()
  two_days <- 60 * 60 * 24 * 2
  o311_requests(end_date = now, start_date = now - two_days)

  # retrieve only open tickets
  tickets <- o311_requests(status = "open")

  # request the first ticket of the previous response
  rid <- as.character(tickets$service_request_id[1])
  o311_request(rid)

  if (interactive()) {
    # request all data
    o311_request_all()
  }

  # request data of the first 5 pages
  o311_request_all(max_pages = 5)
}
```

o311_services *Get service list*

Description

Get a list of available services. Services are unique to the endpoint / city and thus require an attached jurisdiction using o311_api.

Usage

```
o311_services(...)

o311_service(service_code, ...)
```

Arguments

`...`	Further endpoint-specific parameters as documented in the respective endpoint reference.
`service_code`	Identifier of a single service definition. Service codes can usually be retrieved from `o311_services`.

Value

A dataframe or list containing information about each service.

Examples

```
# set up a jurisdiction
o311_api("san francisco")

if (o311_ok()) {
  # get a list of all services
  services <- o311_services()

  # inspect a service code
  o311_service(services$service_code[1])
}
```

`validate_endpoints` *Validate endpoints*

Description

Checks whether and which endpoints are correctly defined, reachable, and/or valid. Iterates through all endpoints defined in `o311_endpoints` and returns their status along with a reason, if applicable.

Usage

```
validate_endpoints(
  idx = NULL,
  checks = c("discovery", "services", "requests"),
  methods = c("formal", "down", "valid")
)
```

Arguments

`idx`	`[integer]` Index numbers of endpoints to check. Index numbers follow row numbers in `o311_endpoints`.
`checks`	`[character]` Which open311 method to check. By default, checks all methods.

methods	[character] Which checks to apply. formal checks whether an endpoint is uniquely identifiable through given names and jurisdictions in o311_endpoints. down checks whether an endpoint is reachable and ready for requests. valid checks whether a method returns a valid output, i.e. a list or dataframe with more than 0 rows/elements. By default, applies all methods.

Value

A dataframe containing the name of the endpoint, one to three columns on check results, and one to three columns on reasons if a check turned out to be negative.

Examples

```
# check the first three endpoints in o311_endpoints()
validate_endpoints(1:3)

# check only requests
validate_endpoints(1:3, checks = "requests")

# check only whether an endpoint is down
validate_endpoints(1:3, methods = "down")
```

Index