Package 'rapiclient'

May 9, 2026

Type Package

Title Dynamic OpenAPI/Swagger Client

Version 0.1.8

URL https://github.com/bergant/rapiclient

BugReports https://github.com/bergant/rapiclient/issues

Description Access services specified in OpenAPI (formerly Swagger) format. It is not a code generator. Client is generated dynamically as a list of R functions.

Depends R (>= 3.3.0)

License MIT + file LICENSE

Imports jsonlite, httr, tools, yaml

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat

NeedsCompilation no

Author Darko Bergant [aut],
Marcel Ramos [cre, ctb] (ORCID:
<https://orcid.org/0000-0002-3242-0582>),
Alexandru Mahmoud [ctb],
Sean Davis [ctb],
Martin Morgan [ctb]

Maintainer Marcel Ramos <marcel.ramos@sph.cuny.edu>

Repository CRAN

Date/Publication 2024-09-30 21:10:02 UTC

Contents

`rapiclient-package` *rapiclient: Open API (Swagger) Client*

Description

Create R functions directly from OpenAPI (formerly Swagger) specification.

Creating a client

Use `get_api` to read the specification, `get_operations` to get client functions and `get_schemas` to create functions for additional schemas.

See usage example at https://github.com/bergant/rapiclient#rapiclient

Check out https://github.com/OAI/OpenAPI-Specification for additional information about Open API specification

Support

Please use https://github.com/bergant/rapiclient/issues for issues

Author(s)

Maintainer: Marcel Ramos <marcel.ramos@sph.cuny.edu> (ORCID) [contributor]

Authors:

- Darko Bergant <darko.bergant@gmail.com>

Other contributors:

- Alexandru Mahmoud [contributor]
- Sean Davis [contributor]
- Martin Morgan [contributor]

See Also

Useful links:

- https://github.com/bergant/rapiclient
- Report bugs at https://github.com/bergant/rapiclient/issues

Examples

```
## Not run:
# Read API description
api <- get_api(api_url)

# create operation and schema functions
operations <- get_operations(api)
schemas <- get_schemas(api)

# call service
operations$some_operation(x, y, schemas$some_structure(u, v, ...))

## End(Not run)
```

`get_api` *Get API*

Description

Create API object from Swagger specification

Usage

```
get_api(url, config = NULL, ext)
```

Arguments

`url`	API URL or file (can be json or yaml format)
`config`	httr::config() curl options.
`ext`	the file extension of the API file (either 'yaml' or 'json'). By default, it is obtained from the URL with 'tools::file_ext' and should be provided when the file URL is missing an extension.

Value

API object

See Also

See also `get_operations` and `get_schemas`

Examples

```
## Not run:
# create operation and schema functions
api_url <- "http://petstore.swagger.io/v2/swagger.json"
api <- get_api(api_url)
operations <- get_operations(api)
schemas <- get_schemas(api)

## End(Not run)
```

`get_operations` *Get operations*

Description

Creates a list of functions from API operations definition. Names in a list are operationIDs from API.

Usage

```
get_operations(
  api,
  .headers = NULL,
  path = NULL,
  handle_response = identity,
  auto_unbox = FALSE
)
```

Arguments

`api`	API object (see `get_api`)
`.headers`	Optional headers passed to httr functions. See `add_headers` documentation
`path`	(optional) filter by path from API specification
`handle_response`	(optional) A function with a single argument: httr response
`auto_unbox`	automatically `unbox()` all atomic vectors of length 1. It is usually safer to avoid this and instead use the `unbox()` function to unbox individual elements. An exception is that objects of class `AsIs` (i.e. wrapped in `I()`) are not automatically unboxed. This is a way to mark single values as length-1 arrays.

Details

All functions return a response object from httr package or a value returned by `handle_response` function if specified. When `path` is defined, only operations with the specified API path root are created. Use `.headers` parameters to send additional headers when sending a request.

Value

A list of functions.

Handling response

If no response handler function is defined, operation functions return response object (**httr** package). See httr content documentation for extracting content from a request, and functions http_error and http_status how to handle http errors and error messages.

When using simple `result_handlers`, operations will return the content of response instead of httr response object (or handle error as exception or warning in case of error).

To handle response automatically with custom function, define a function with httr response object as argument and pass it as `handle_response` argument to `get_operations` function.

Examples

```
## Not run:
# create operation and schema functions
api_url <- "http://petstore.swagger.io/v2/swagger.json"
api <- get_api(api_url)
operations <- get_operations(api)
schemas <- get_schemas(api)

# get operations which return content or stop on error
operations <- get_operations(api, handle_response = content_or_stop)

# use .headers when operations must send additional headers when sending
operations <-
  get_operations(api, .headers = c("api-key" = Sys.getenv("SOME_API_KEY")))

## End(Not run)
```

`get_schemas` *Get schemas*

Description

Returns a list of functions with arguments from API schemas. Elements are named by schema names, each function returns a named list.

Usage

```
get_schemas(api)
```

Arguments

`api`	Api object

Value

A list of functions

`result_handlers` *Simple functions to handle http response*

Description

When creating operations from api one can define how the response from http should be handled. These functions can be used for simple result handling.

Usage

```
content_or_stop(x)

content_or_warning(x)

content_or_message(x)
```

Arguments

x	A response object from httr package (see response object in **httr** package documentation)

Details

See `get_operations` for details.

Value

Content of http response

Functions

- `content_or_warning()`: Returns content or issues a warning
- `content_or_message()`: Returns content or prints a message

Examples

```
api_file <- system.file(
  "extdata", "sample_specs", "petstore.yaml",
  package = "rapiclient", mustWork = TRUE
)
api <- get_api(api_file)
operations <- get_operations(api, handle_response = content_or_stop)
```

Index