Package ‘trelloR’

August 27, 2023

Title Access the Trello API

Version 0.8.0

Description An R client for the Trello API. Supports free-tier features such as access to private boards, creating and updating cards and other resources, and downloading data in a structured way.

Depends R (>= 3.6.0)

Imports curl, httr (>= 1.4.0), jsonlite (>= 1.7.0)

License GPL-3

URL https://github.com/jchrom/trelloR,
http://jchrom.github.io/trelloR/

BugReports https://github.com/jchrom/trelloR/issues

Date 2023-08-27

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, tibble, testthat

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Jakub Chromec [aut, cre]

Maintainer Jakub Chromec <chromec.jakub@gmail.com>

Repository CRAN

Date/Publication 2023-08-27 18:30:02 UTC

R topics documented:

`add_board` *Add board*

Description

Add board.

Usage

```
add_board(name, body = NULL, ...)
```

Arguments

`name`	Name of the board.
`body`	A named list of query parameters.
`...`	Additional arguments passed to `create_resource()`.

See Also

Other functions to create resources: `add_card_attachment()`, `add_card()`, `add_checkitem()`, `add_checklist()`, `add_comment()`, `add_label()`, `add_list()`, `add_member()`, `create_resource()`

`add_card` *Add card*

Description

Add card to a list.

Usage

```
add_card(list, body = list(name = "New card"), ...)
```

Arguments

`list`	List id.
`body`	A named list of query parameters.
`...`	Additional arguments passed to `create_resource()`.

See Also

Other functions to create resources: `add_board()`, `add_card_attachment()`, `add_checkitem()`, `add_checklist()`, `add_comment()`, `add_label()`, `add_list()`, `add_member()`, `create_resource()`

add_card_attachment *Add card attachment*

Description

Add attachment to a card.

Usage

```
add_card_attachment(
  card,
  file = NULL,
  url = NULL,
  cover = FALSE,
  name = NULL,
  ...
)
```

Arguments

card	Card id.
file, url	Path to a file to be attached, or a URL.
cover	Whether the attached file should be set as cover.
name	Name of the attachment, shown inside the card.
...	Additional arguments passed to create_resource().

See Also

Other functions to create resources: add_board(), add_card(), add_checkitem(), add_checklist(), add_comment(), add_label(), add_list(), add_member(), create_resource()

add_checkitem *Add checklist item*

Description

Add item to a checklist.

Usage

```
add_checkitem(checklist, name, checked = FALSE, position = "bottom", ...)
```

Arguments

`checklist`	Checklist id.
`name`	Item name (text).
`checked`	Whether item should be checked; defaults to `FALSE`.
`position`	Position in the checklist; defaults to `"bottom"`.
`...`	Additional arguments passed to `create_resource()`.

See Also

Other functions to create resources: `add_board()`, `add_card_attachment()`, `add_card()`, `add_checklist()`, `add_comment()`, `add_label()`, `add_list()`, `add_member()`, `create_resource()`

`add_checklist` *Add checklist*

Description

Add checklist to a card.

Usage

```
add_checklist(card, name, source = NULL, ...)
```

Arguments

`card`	Card id.
`name`	Checklist name.
`source`	Items from this checklist id will be copied to the new one. Defaults to `NULL`.
`...`	Additional arguments passed to `create_resource()`.

See Also

Other functions to create resources: `add_board()`, `add_card_attachment()`, `add_card()`, `add_checkitem()`, `add_comment()`, `add_label()`, `add_list()`, `add_member()`, `create_resource()`

add_comment *Add comment*

Description

Add comment to a card.

Usage

```
add_comment(card, text, ...)
```

Arguments

card	Card id.
text	Comment text.
...	Additional arguments passed to create_resource().

See Also

Other functions to create resources: add_board(), add_card_attachment(), add_card(), add_checkitem(), add_checklist(), add_label(), add_list(), add_member(), create_resource()

add_field *Add a custom field*

Description

Create a new custom field definition and attach it to the board.

Usage

```
add_field(id, type, name = "New field", body = NULL, ...)

add_field_checkbox(id, name = "New checkbox", body = NULL, ...)

add_field_date(id, name = "New date", body = NULL, ...)

add_field_dropdown(id, name = "New dropdown", body = NULL, ...)

add_field_number(id, name = "New number", body = NULL, ...)

add_field_text(id, name = "New text", body = NULL, ...)
```

Arguments

`id`	Board ID
`type`	Custom field type
`name`	Custom field name
`body`	Named list with additional parameters
`...`	Additional arguments passed to `create_resource()`

Details

For `add_field_dropdown` - options can be provided using a single (optionally named) vector: `body = list(options = c(red = "Alert", green = "Ok", "Nothing"))`

See Also

Other functions to handle custom fields: `add_field_option()`, `delete_field_option()`, `delete_field()`, `update_card_field()`, `update_field()`

`add_field_option` *Add dropdown option*

Description

Add dropdown (custom field) option.

Usage

```
add_field_option(id, text, color = "none", position = "bottom", ...)
```

Arguments

`id`	Custom field ID (see `get_board_fields()`)
`text`	Option text
`color`	Option color
`position`	Option position
`...`	Additional arguments passed to `create_resource()`

See Also

Other functions to handle custom fields: `add_field()`, `delete_field_option()`, `delete_field()`, `update_card_field()`, `update_field()`

add_label *Add label*

Description

Add label to a card.

Usage

```
add_label(card, color, name = NULL, ...)
```

Arguments

card	Card id.
color	Label color.
name	Label name; choosing different non-existing name will create new label. Defaults to NULL.
...	Additional arguments passed to create_resource().

See Also

Other functions to create resources: add_board(), add_card_attachment(), add_card(), add_checkitem(), add_checklist(), add_comment(), add_list(), add_member(), create_resource()

add_list *Add list*

Description

Add list to a board.

Usage

```
add_list(board, name, position = NULL, ...)
```

Arguments

board	Board id.
name	List name.
position	List position. One of "top", "bottom" or NULL.
...	Additional arguments passed to create_resource().

See Also

Other functions to create resources: add_board(), add_card_attachment(), add_card(), add_checkitem(), add_checklist(), add_comment(), add_label(), add_member(), create_resource()

add_member *Add member*

Description

Add member to a card.

Usage

```
add_member(card, member, ...)
```

Arguments

card	Card id.
member	Member id.
...	Additional arguments passed to `create_resource()`.

See Also

Other functions to create resources: `add_board()`, `add_card_attachment()`, `add_card()`, `add_checkitem()`, `add_checklist()`, `add_comment()`, `add_label()`, `add_list()`, `create_resource()`

as_POSIXct_hex *Convert hex string into POSIXct*

Description

Convert hex string into POSIXct

Usage

```
as_POSIXct_hex(x)
```

Arguments

x	Vector of strings, each 8 characters long

create_resource *Create Resources*

Description

Create resources via Trello API.

Usage

```
create_resource(
  resource,
  id = NULL,
  path = NULL,
  body = list(name = "New"),
  token = NULL,
  on.error = c("stop", "warn", "message"),
  verbose = FALSE,
  handle = NULL,
  encode,
  response
)
```

Arguments

`resource` Model name, eg. `"card"`.

`id` Model id.

`path` Path.

`body` A named list.

`token` An object of class `"Trello_API_token"`, a path or `NULL`.

- If a `Token`, it is passed as is.
- If `NULL` and a cache file called `".httr-oauth"` exists, the newest token is read from it. If the file is not found, an error is thrown.
- If a character vector of length 1, it will be used as an alternative path to the cache file.

`on.error` Behavior when HTTP status >= 300, defaults to `"stop"`.

`verbose` Whether to pass `httr::verbose()` to `httr::RETRY()`.

`handle` Passed to `httr::RETRY()`.

`encode, response` Deprecated.

Details

See Trello API reference for more info about what elements can be included in POST request body.

See Also

Other functions to create resources: `add_board()`, `add_card_attachment()`, `add_card()`, `add_checkitem()`, `add_checklist()`, `add_comment()`, `add_label()`, `add_list()`, `add_member()`

Examples

```
## Not run:

# Get token with write access
key = Sys.getenv("MY_TRELLO_KEY")
secret = Sys.getenv("MY_TRELLO_SECRET")

token = get_token("my_app", key = key, secret = secret,
                  scope = c("read", "write"))

# Get board ID
url = "Your board URL"
bid = get_id_board(url, token)

# Get lists on that board, extract ID of the first one
lid = get_board_lists(bid, token)$id[1]

# Content for the new card
payload = list(
  idList = lid,
  name = "A new card",
  desc = "#This card has been created by trelloR",
  pos = "bottom"
)

# Create card and store the response (to capture the ID
# of the newly created resource)
r = create_resource("card", body = payload, token = token)

# Get ID of the new card
r$id

## End(Not run)
```

`delete_card` *Delete card*

Description

DELETE card.

Usage

```
delete_card(card, ...)
```

Arguments

`card`	Card id
`...`	Additional arguments passed to `delete_resource()`

`delete_checkitem` *Delete item*

Description

DELETE checklist item.

Usage

```
delete_checkitem(checklist, checkitem, ...)
```

Arguments

`checklist`	Checklist id
`checkitem`	Checklist item id
`...`	Additional arguments passed to `delete_resource()`

`delete_checklist` *Delete checklist*

Description

DELETE checklist.

Usage

```
delete_checklist(checklist, ...)
```

Arguments

`checklist`	Checklist id
`...`	Additional arguments passed to `delete_resource()`

delete_field *Delete custom field*

Description

Delete custom field - this will remove it from all cards on the board.

Usage

```
delete_field(id, ...)
```

Arguments

`id`	Custom field ID (see `get_board_fields()`)
`...`	Additional arguments passed to `create_resource()`

See Also

Other functions to handle custom fields: `add_field_option()`, `add_field()`, `delete_field_option()`, `update_card_field()`, `update_field()`

delete_field_option *Delete dropdown option*

Description

Delete dropdown (custom field) option - this will remove it from all cards on the board.

Usage

```
delete_field_option(id, option, ...)
```

Arguments

`id`	Custom field ID (see `get_board_fields()`
`option`	Dropdown option ID
`...`	Additional arguments passed to `create_resource()`

See Also

Other functions to handle custom fields: `add_field_option()`, `add_field()`, `delete_field()`, `update_card_field()`, `update_field()`

`delete_resource` *Delete Resources*

Description

Delete resources via Trello API.

Usage

```
delete_resource(
  resource,
  id = NULL,
  path = NULL,
  token = NULL,
  on.error = c("stop", "warn", "message"),
  verbose = FALSE,
  handle = NULL,
  encode,
  response
)
```

Arguments

`resource, id`	Resource name (eg. `"card"`) and id.
`path`	Path (optional).
`token`	An object of class `"Trello_API_token"`, a path or `NULL`. • If a `Token`, it is passed as is. • If `NULL` and a cache file called `".httr-oauth"` exists, the newest token is read from it. If the file is not found, an error is thrown. • If a character vector of length 1, it will be used as an alternative path to the cache file.
`on.error`	Behavior when HTTP status >= 300, defaults to `"stop"`.
`verbose`	Whether to pass `httr::verbose()` to `httr::RETRY()`.
`handle`	Passed to `httr::RETRY()`.
`encode, response`	Deprecated.

Details

See Trello API reference for more info about DELETE requests.

Value

See `response`.

Examples

```
## Not run:

# Get token with write access
key = Sys.getenv("MY_TRELLO_KEY")
secret = Sys.getenv("MY_TRELLO_SECRET")

token = get_token("my_app", key = key, secret = secret,
                  scope = c("read", "write"))

# Get board ID
url = "Your board URL"
bid = get_id_board(url, token)

# Get cards and extract ID of the first one
cid = get_board_cards(bid, token)$id[1]

# Delete it
delete_resource(resource = "card", id = cid, token = token)

## End(Not run)
```

Deprecated *Deprecated functions*

Description

- Use get_resource() instead of get_model() or trello_get()
- Use update_resource() instead of put_model()
- Use create_resource() instead of post_model()
- Use delete_resource() instead of delete_model()
- Use get_token() instead of trello_get_token()
- Use search_resource() instead of trello_search()

Usage

```
trello_get(...)

get_model(...)

post_model(...)

put_model(...)

delete_model(...)
```

```
trello_get_token(...)

trello_search(...)

search_model(...)

trello_search_actions(...)

trello_search_cards(...)

trello_search_boards(...)

trello_search_members(...)

trello_search_teams(...)
```

Arguments

`...`	See current functions for argument names and defaults.

`extract_id` *Extract ID*

Description

Extract resource ID from its URL. If input is not a valid URL, it is returned as is.

Usage

```
extract_id(x)
```

Arguments

`x`	character vector of length 1

`get_batch` *Dispatch a Batch Request*

Description

Perform 1 to 10 GET requests at once.

Usage

```
get_batch(
  parent,
  child,
  ids,
  token = NULL,
  query = NULL,
  on.error = c("stop", "warn", "message"),
  retry.times = 1,
  handle = NULL,
  verbose = FALSE
)
```

Arguments

`parent` Parent resource, e.g. `"board"`.

`child` Child resource, eg. `"card"`.

`ids` A vector of resource IDs, with maximum length of 10.

`token` An object of class `"Trello_API_token"`, a path or `NULL`.

- If a `Token`, it is passed as is.
- If `NULL` and a cache file called `".httr-oauth"` exists, the newest token is read from it. If the file is not found, an error is thrown.
- If a character vector of length 1, it will be used as an alternative path to the cache file.

`query` Named list of key-value pairs, appended to each component of the batch request. See `httr::GET()` for details. NOTE: If you require different query with each batch component, build the list of URLs yourself using `get_resource()`, as per Trello API reference.

`on.error` Whether to `"stop"`, `"warn"` or `"message"` on API error.

`retry.times` How many times to re-try when a request fails. Defaults to 1.

`handle` The handle to use with this request, see `httr::RETRY()`.

`verbose` Set to `TRUE` for verbose output.

Value

A data frame (or a tibble, if installed).

See Also

`get_token()`, `get_id()`, `httr::GET()`, `jsonlite::fromJSON()`

Examples

```
demo_board <- "https://trello.com/b/wVWPK9I4/r-client-for-the-trello-api"
```

```
if (FALSE) {
# Download custom field values for 10 cards.
cards <- get_board_cards(demo_board, limit = 10)
values <- get_batch(
  parent = "card",
  child = "customFieldItems",
  ids = cards$id
)
values
}
```

get_board *Get Board*

Description

Returns a flat data frame with board-related data.

Usage

```
get_board_actions(id, ...)

get_board_cards(id, ...)

get_board_checklists(id, ...)

get_board_comments(id, ...)

get_board_labels(id, ...)

get_board_lists(id, ...)

get_board_members(id, ...)

get_board_fields(id, ...)

get_board_prefs(id, ...)
```

Arguments

`id`	Board ID
`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

Examples

```
## Not run:
#Get cards with custom fields
cards = get_board_cards(board_id, query = list(customFieldItems = "true"))

## End(Not run)
```

`get_card` *Get Card*

Description

Returns a flat data frame with card-related data.

Usage

```
get_card_actions(id, ...)

get_card_checklists(id, ...)

get_card_comments(id, ...)

get_card_labels(id, ...)

get_card_members(id, ...)

get_card_fields(id, ...)

get_card_fields_values(id, ...)
```

Arguments

`id`	Card ID
`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

`get_id` *Get model ID*

Description

Get ID of a resource.

Usage

```
get_id_board(url, token = NULL)

get_id_card(url, token = NULL)
```

Arguments

`url`	Complete url, short url or just the url ID part of a Trello board
`token`	Secure token, see `get_token()`

Examples

```
# Get Trello Development Roadmap board ID
url = "https://trello.com/b/nC8QJJoZ/trello-development-roadmap"
tdr_id = get_id_board(url)

# Also works:
url = "nC8QJJoZ"
tdr_id = get_id_board(url)
```

`get_list` *Get List*

Description

Returns a flat data frame with list-related data.

Usage

```
get_list_actions(id, ...)

get_list_cards(id, ...)

get_list_comments(id, ...)
```

Arguments

`id`	Board ID
`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

`get_member_boards` *Get Member's Boards*

Description

Returns a flat data frame with member-related data.

Usage

```
get_member_boards(id, ...)
```

Arguments

`id`	member ID or username
`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

`get_member_fields` *Get Member Info*

Description

Returns a flat data frame with member-related data.

Usage

```
get_member_fields(
  id,
  fields = c("fullName", "username", "memberType", "bio"),
  ...
)
```

Arguments

`id`	member ID or username
`fields`	by default fetches fullName, username, memberType, bio
`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

get_my_boards *Get Own Boards*

Description

Returns a flat data frame, containing the authenticated user's boards.

Usage

```
get_my_boards(...)
```

Arguments

`...`	Additional arguments passed to `get_resource()`

See Also

`get_resource()`

get_resource *Get Data From Trello API*

Description

Fetch resources using Trello API.

Usage

```
get_resource(
  parent = NULL,
  child = NULL,
  id = NULL,
  token = NULL,
  query = NULL,
  url = NULL,
  filter = NULL,
  limit = 100,
  on.error = c("stop", "warn", "message"),
  retry.times = 1,
  handle = NULL,
  verbose = FALSE,
  response,
  paging,
  bind.rows
)
```

Arguments

`parent` Parent resource, e.g. `"board"` or `NULL`.

`child` Child resource, eg. `"card"` or `NULL`.

`id` Resource ID or `NULL`.

`token` An object of class `"Trello_API_token"`, a path or `NULL`.

- If a `Token`, it is passed as is.
- If `NULL` and a cache file called `".httr-oauth"` exists, the newest token is read from it. If the file is not found, an error is thrown.
- If a character vector of length 1, it will be used as an alternative path to the cache file.

`query` Named list of key-value pairs, see `httr::GET()` for details.

`url` Url for the GET request. Can be `NULL` if `parent` is specified, or a combination of `parent`, `child` and `id` is provided.

`filter` Defaults to `"all"` which includes both open and archived cards or all action types, depending on what resource is requested.

`limit` Defaults to `100`. Set to `Inf` to get everything.

`on.error` Whether to `"stop"`, `"warn"` or `"message"` on API error.

`retry.times` How many times to re-try when a request fails. Defaults to 1.

`handle` The handle to use with this request, see `httr::RETRY()`.

`verbose` Set to `TRUE` for verbose output.

`response, paging, bind.rows`
Deprecated.

Value

A data frame with API responses.

Request limits

At maximum, the API can retrieve 1000 results in a single call. Setting `limit > 1000` will activate paging. When paging is used, the request will be issued repeatedly, retrieving new batch of results each time until the `limit` is reached or there is nothing else to fetch. Results are fetched chronologically, ie. newest results are retrieved first (eg. newest cards). Use `limit = Inf` to make sure you get all.

Errors

If the request fails, server error messages are reprinted on the console. Depending on the value of `on.error`, the request call can throw an error in R (this is the default), or can issue a warning/message. If the latter, the function returns a data frame containing the failed URL, HTTP status and an informative message (produced by the server).

Results

The API returns JSON objects which are parsed using jsonlite::fromJSON(). Non-JSON results throw an error, but these should never happen anyway. The result is always a data frame, or a tibble if the package is installed.

Filter

Both filter and limit exist as explicitly defined arguments, but you can ignore them in favor of supplying their values as query parameters, eg. query = list(filter = "filter_value", limit = "limit_value").

See Also

get_token(), get_id(), httr::GET(), jsonlite::fromJSON()

Examples

```
# Public boards can be accessed without authorization, so there is no need
# to create a token, just the board id:
url = "https://trello.com/b/wVWPK9I4/r-client-for-the-trello-api"
bid = get_id_board(url)

# Getting resources from the whole board. `filter="all"` fetches archived
# cards as well.
labels = get_board_labels(bid)
cards = get_board_cards(bid, filter = "all")

# It is possible to call `get_resource()` directly:
lists = get_resource(parent = "board", child = "lists", id = bid)

# As with boards, cards can be queried for particular resources, in this case
# to obtain custom fields:
card = cards$id[5]
acts = get_card_fields(card)

# Set `limit` to specify the number of results. Pagination will be used
# whenever limit exceeds 1000. Use `limit=Inf` to make sure you get all.

## Not run:
all_actions = get_board_actions(bid, limit = Inf)

## End(Not run)

# For private and team boards, a secure token is required:

## Not run:
key = Sys.getenv("MY_TRELLO_KEY")
secret = Sys.getenv("MY_TRELLO_SECRET")

token = get_token("my_app", key = key, secret = secret,
                  scope = c("read", "write"))

# Token is now cached, no need to pass it explicitly.
cards_open = get_board_cards(board_id, filter = "open")

## End(Not run)
```

get_team *Get Team*

Description

Returns a flat data frame with team/organization-related data

Usage

```
get_team_members(id, ...)

get_team_boards(id, ...)
```

Arguments

`id`	team ID, short name or URL
`...`	Additional arguments passed to `get_resource()`

Details

Previously, teams were called "organizations", and the correct parent/child name in API calls remains "organization", "organizations".

See Also

`get_resource()`

get_token *Get OAuth1.0 Token*

Description

Authorize access to Trello API, required for private boards and write access (see details).

Usage

```
get_token(
  app = NULL,
  key = NULL,
  secret = NULL,
  scope = c("read", "write", "account"),
  expiration = c("30days", "1day", "1hour", "never"),
  cache = getOption("httr_oauth_cache"),
  appname
)
```

Arguments

`app` A string, `NULL` or a Token.

- If `key` and `secret` are set, a new token will be initialized and its name set to this value. A meaningful name can help you to spot your token on the settings page `https://trello.com/username/account`. A token cannot be initialized without a name, so it will default to `"trello-app"` if it needs to.
- If `key` and `secret` are not set, the value of `app` will be used as a path from which an existing token is read. If `NULL`, the token will be read from the working directory. `NULL` is returned if nothing is found.
- If a Token, return as is.

`key, secret` Developer credentials from `https://trello.com/app-key` (see details). If `NULL` and a cache file exists, the newest token is read from it, otherwise an error is thrown.

`scope` Can be one or several of `"read"`, `"write"` or `"account"`. Defaults to `"read"`.

`expiration` Can be `"1hour"`, `"1day"`, `"30days"` or `"never"`. Defaults to `"30days"`.

`cache` Passed to `httr::oauth1.0_token()`. Can specify whether the token should be cached locally (will ask the first time and then `TRUE` by default) or choose an alternative path for the cache file.

`appname` Deprecated, use `app`.

Value

An object of class `"Trello_API_token"` (a Token).

Getting developer access

To access private data, a secure token is required. In order to create it, you will need your developer `key` and `secret` which you can get by visiting `https://trello.com/app-key` after logging in to Trello. You may also want to set at least one *allowed origin* there. If you are using trelloR locally (ie. from your laptop or PC), `http://localhost:1410` is a good value to use.

Creating tokens

Once back in R, run this function the following way:

my_token = get_token("my_app", key = key, secret = secret)

passing the values you have obtained on the developer page. First time you create a token, you will be prompted to confirm the authorization in a browser. If you chose to store the token locally as prompted, you won't have to do this anymore until your token expires (see expiration) or your local cache file is deleted.

Tokens are stored inside a hidden .httr-oauth cache file and automatically read when any function in the package is called. Optionally, you can specify a different cache path using the cache argument, or avoid caching the token completely with cache = FALSE. See httr::oauth1.0_token() for details.

If you opt out of storing the token, then it will only be held until your R session is over, and you will have to pass it to the token argument, eg. get_my_boards(token = my_token) each time you are fetching data.

Remember to store your credentials in a **secure, non-shared** location. To minimize the risk of unwanted disclosure when using remote code repositories, .httr-oauth (or whatever cache path you have specified using the cache argument) is automatically added to .gitignore.

See Also

httr::oauth_app(), httr::oauth_endpoint(), httr::oauth1.0_token()

Examples

```
# This example assumes you are reading your key and secret from environment
# variables. This is not obligatory, but wherever you read them from, make
# sure it is a secure, non-shared location.

## Not run:

key = Sys.getenv("MY_TRELLO_KEY")
secret = Sys.getenv("MY_TRELLO_SECRET")

token = get_token("my_app", key = key, secret = secret,
                  scope = c("read", "write"))

## End(Not run)
```

move_card *Move card*

Description

Move card to another list.

Usage

```
move_card(card, list, ...)
```

Arguments

`card`	Card id.
`list`	List id.
`...`	Additional arguments passed to `update_resource()`.

`print.Trello_API_token` *Print Trello API Tokens*

Description

Objects of class "`Trello_API_token`" posses some additional attributes to make it easier to see the app name, its scope and expiration. If you want to print the bare token object as generated by `[httr::oauth1.0_token()]`, call its own print method, eg: `my_token$print()`.

Usage

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

Arguments

`x`	An object of class "`Trello_API_token`".
`...`	Unused.

`search_resource` *Search Trello*

Description

Search for resources.

Usage

```
search_resource(
  string,
  resource = "all",
  boards = "mine",
  partial = FALSE,
  query = list(),
  modeltype,
  ...
)

search_cards(string, boards = "mine", partial = FALSE, query = list(), ...)

search_actions(string, boards = "mine", partial = FALSE, query = list(), ...)

search_boards(string, boards = "mine", partial = FALSE, query = list(), ...)

search_members(string, boards = "mine", partial = FALSE, query = list(), ...)

search_teams(string, boards = "mine", partial = FALSE, query = list(), ...)
```

Arguments

`string` Text to search for.

`resource` Type of resource to return; typically `"cards"`. Defaults to `"all"`.

`boards` Boards to limit the search to - defaults to `"mine"`.

`partial` Should partial matching be used? Defaults to `FALSE`.

`query` Name list of additional query parameters; consult search API reference

`modeltype` Deprecated, use `resource` instead.

`...` Additional arguments passed to `get_resource()`.

Details

Search can be narrowed down by resource type and will return a single data frame per each type (if anything is found). The value of the `resource` argument is passed on `modelTypes` - see other attributes at search API reference.

Value

A data frame.

See Also

`get_resource()`, `get_token()`

Examples

```
## Not run:

search_resource("Anything with this text")
search_cards("A card with this text")


## End(Not run)
```

`update_card` *Update card*

Description

Update name, description, assigned members and other fields.

Usage

```
update_card(card, body = NULL, ...)
```

Arguments

`card`	Card id
`body`	A named list of query parameters.
`...`	Additional arguments passed to `update_resource()`.

`update_card_field` *Update card field value*

Description

Set custom field value on a single card.

Usage

```
update_card_field(card, field, key, value, ...)

update_card_checkbox(card, field, value, ...)

update_card_date(card, field, value, ...)

update_card_dropdown(card, field, value, ...)

update_card_number(card, field, value, ...)
```

```
update_card_text(card, field, value, ...)

clear_card_field(card, field, ...)
```

Arguments

`card`	Card ID
`field`	Custom field ID
`key`	Key for the value, e.g. `"number"` or `"checked"`
`value`	New value
`...`	Additional arguments passed to `update_resource()`

Details

`update_card_date` requires ISO Formatted Datetime String. `"YYYY-MM-DD"` is fine, but if you want also hour and timezone, use `"YYYY-MM-DD hh:mm UTC+X"`, e.g. `"2018-12-24 16:00 UTC+1"`

`clear_card_field` does not remove the field, but replaces its value with the equivalent of "No selection"

See Also

Other functions to handle custom fields: `add_field_option()`, `add_field()`, `delete_field_option()`, `delete_field()`, `update_field()`

`update_card_labels` *Update card labels*

Description

Replace currently assigned labels.

Usage

```
update_card_labels(card, labels, ...)
```

Arguments

`card`	Card id.
`labels`	A character vector of one or more label id.
`...`	Additional arguments passed to `update_resource()`.

update_card_members *Update card members*

Description

Replace currently assigned members.

Usage

```
update_card_members(card, members, ...)
```

Arguments

`card`	Card id.
`members`	A character vector of one or more member id.
`...`	Additional arguments passed to `update_resource()`.

update_checkitem *Update item*

Description

Update checklist item state, name, position and which checklist it is in.

Usage

```
update_checkitem(card, checkitem, body = list(state = "complete"), ...)
```

Arguments

`card`	Card id.
`checkitem`	Checklist item id.
`body`	A named list of query parameters. Defaults to `list(state = "complete")`, which makes the item complete.
`...`	Additional arguments passed to `update_resource()`.

update_field *Update custom field*

Description

Update custom field definition.

Usage

```
update_field(id, body = list(name = "New name"), ...)
```

Arguments

`id`	Board ID
`body`	Named list with additional parameters
`...`	Additional arguments passed to `update_resource()`

See Also

Other functions to handle custom fields: `add_field_option()`, `add_field()`, `delete_field_option()`, `delete_field()`, `update_card_field()`

update_resource *Update Resources*

Description

Update resources via Trello API.

Usage

```
update_resource(
  resource,
  id = NULL,
  path = NULL,
  body = NULL,
  token = NULL,
  on.error = c("stop", "warn", "message"),
  verbose = FALSE,
  handle = NULL,
  encode,
  response
)
```

Arguments

resource Model name, eg. "card".

id Model id.

path Path.

body A named list.

token An object of class "Trello_API_token", a path or NULL.

- If a Token, it is passed as is.
- If NULL and a cache file called ".httr-oauth" exists, the newest token is read from it. If the file is not found, an error is thrown.
- If a character vector of length 1, it will be used as an alternative path to the cache file.

on.error Behavior when HTTP status >= 300, defaults to "stop".

verbose Whether to pass httr::verbose() to httr::RETRY().

handle Passed to httr::RETRY().

encode, response
Deprecated.

Details

See Trello API reference for more info about what elements can be included in PUT request body.

Examples

```
## Not run:

# Get token with write access
key = Sys.getenv("MY_TRELLO_KEY")
secret = Sys.getenv("MY_TRELLO_SECRET")

token = get_token("my_app", key = key, secret = secret,
                  scope = c("read", "write"))

# Get board ID
url = "Your board URL"
bid = get_id_board(url, token)

# Get cards and extract ID of the first one
cid = get_board_cards(bid, token)$id[1]

# Content for the new card
payload = list(
  id = cid,
  name = "A new card name",
  desc = "Description - updated by trelloR",
  pos = "top" #put card on the top of a list
)
```

```
# Update card's name, descriptionand position
update_resource("card", id = cid, body = payload, token = token)

## End(Not run)
```

Index