Package ‘trelloR’

April 22, 2021

Title Access the Trello API

Version 0.7.1

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 httr (>= 1.4.0), jsonlite (>= 1.7.0)
License GPL-3

URL https://github.com/jchrom/trelloR

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

Date 2021-04-20

RoxygenNote 7.1.1

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 2021-04-22 00:50:02 UTC

R topics documented:

add_board ............................................................... 2
add_card ............................................................... 3
add_card_attachment .................................................. 4
add_checkitem .......................................................... 4
add_checklist .......................................................... 5
add_comment .......................................................... 6
add_board

Description

Add board.
add_card

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

Description

Add checklist to a card.

Usage

```r
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, ...)
add_field_option

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Board ID</td>
</tr>
<tr>
<td>type</td>
<td>Custom field type</td>
</tr>
<tr>
<td>name</td>
<td>Custom field name</td>
</tr>
<tr>
<td>body</td>
<td>Named list with additional parameters</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to <code>create_resource()</code></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Custom field ID (see <code>get_board_fields()</code>)</td>
</tr>
<tr>
<td>text</td>
<td>Option text</td>
</tr>
<tr>
<td>color</td>
<td>Option color</td>
</tr>
<tr>
<td>position</td>
<td>Option position</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to <code>create_resource()</code></td>
</tr>
</tbody>
</table>

See Also

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

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

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

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

```r
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](https://trello.com/api) for more info about what elements can be included in POST request body.
delete_card

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

```r
# 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, ...)
**delete_checklist**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>card</td>
<td>Card id</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to delete_resource()</td>
</tr>
</tbody>
</table>

**Description**

DELETE checklist.

**Usage**

delete_checklist(checklist, ...)
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**: Deprecated.
- **response**: Deprecated.

Details
See Trello API reference for more info about DELETE requests.

Value
See response.
Examples

```r
## 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 functions

<table>
<thead>
<tr>
<th>Deprecated function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>trello_get(...)</code></td>
<td>Use <code>get_resource()</code> instead of <code>get_model()</code> or <code>trello_get()</code></td>
</tr>
<tr>
<td><code>get_model(...)</code></td>
<td>Use <code>update_resource()</code> instead of <code>put_model()</code></td>
</tr>
<tr>
<td><code>post_model(...)</code></td>
<td>Use <code>create_resource()</code> instead of <code>post_model()</code></td>
</tr>
<tr>
<td><code>put_model(...)</code></td>
<td>Use <code>delete_resource()</code> instead of <code>delete_model()</code></td>
</tr>
<tr>
<td><code>delete_model(...)</code></td>
<td>Use <code>get_token()</code> instead of <code>trello_get_token()</code></td>
</tr>
<tr>
<td><code>trello_search(...)</code></td>
<td>Use <code>search_resource()</code> instead of <code>trello_search()</code></td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>extract_id</th>
<th>Extract ID</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>get_board</th>
<th>Get Board</th>
</tr>
</thead>
</table>

Description

Returns a flat data frame with board-related data.
### get_card

**Usage**

```r
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**

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

---

**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, ...)

Arguments

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

See Also

get_resource()

get_id Board 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**

```r
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**

```r
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

class

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 \texttt{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 \texttt{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, i.e. newest results are retrieved first (e.g. 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, e.g. `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/wWPK9I4/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, ...)
get_token

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.
get_token

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Can be one or several of &quot;read&quot;, &quot;write&quot; or &quot;account&quot;. Defaults to &quot;read&quot;.</td>
</tr>
<tr>
<td>expiration</td>
<td>Can be &quot;1hour&quot;, &quot;1day&quot;, &quot;30days&quot; or &quot;never&quot;. Defaults to &quot;30days&quot;.</td>
</tr>
<tr>
<td>cache</td>
<td>Passed to <code>httr::oauth1.0_token()</code>. 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.</td>
</tr>
</tbody>
</table>
| 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:

```r
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**

```r
# 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:

```r
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" possess 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**

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

**Arguments**

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

Search for resources.

Usage

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

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

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

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

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

```r
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

```r
## Not run:
search_resource("Anything with this text")
search_cards("A card with this text")

## End(Not run)
```

---

trelloR | trelloR: R API for Trello

Description

Access the Trello API. Free-tier features are fully supported, including authentication and administration of private boards, teams, creating and updating cards and other resources, and downloading data in a structured way.

---

update_card | Update card

Description

Update name, description, assigned members and other fields.

Usage

```r
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**

```r
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.
update_resource

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, description and position
update_resource("card", id = cid, body = payload, token = token)

## End(Not run)
Index

* functions to create resources
  add_board, 2
  add_card, 3
  add_card_attachment, 4
  add_checkitem, 4
  add_checklist, 5
  add_comment, 6
  add_label, 8
  add_list, 8
  add_member, 9
  create_resource, 10

* functions to handle custom fields
  add_field, 6
  add_field_option, 7
  delete_field, 7
  delete_field_option, 7
  update_card_field, 29
  update_field, 31

add_board, 2, 3–6, 8, 9, 11
add_card, 3, 4–6, 8, 9, 11
add_card_attachment, 3, 4, 5, 6, 8, 9, 11
add_checkitem, 3, 4, 5, 6, 8, 9, 11
add_checklist, 3–5, 5, 6, 8, 9, 11
add_comment, 3–5, 6, 8, 9, 11
add_field, 6, 7, 13, 29, 31
add_field_checkbox (add_field), 6
add_field_date (add_field), 6
add_field_dropdown (add_field), 6
add_field_number (add_field), 6
add_field_option, 7, 7, 13, 29, 31
add_field_text (add_field), 6
add_label, 3–6, 8, 9, 11
add_list, 3–6, 8, 9, 11
add_member, 3–6, 8, 9, 11
as_POSIXct_hex, 9

clear_card_field (update_card_field), 29
create_resource, 3–6, 8, 9, 10
create_resource(), 3–9, 13, 15

delete_card, 11
delete_checkitem, 12
delete_checklist, 12
delete_field, 7, 13, 29, 31
delete_field_option, 7, 13, 29, 31
delete_model (Deprecated), 15
delete_resource, 14
delete_resource(), 12, 15
Deprecated, 15

eextract_id, 16
get_board, 16
get_board_actions (get_board), 16
get_board_cards (get_board), 16
get_board_checklists (get_board), 16
get_board_comments (get_board), 16
get_board_fields (get_board), 16
get_board_fields(), 7, 13
get_board_labels (get_board), 16
get_board_lists (get_board), 16
get_board_members (get_board), 16
get_board_prefs (get_board), 16
get_card, 17
get_card_actions (get_card), 17
get_card_checklists (get_card), 17
get_card_comments (get_card), 17
get_card_fields (get_card), 17
get_card_labels (get_card), 17
get_card_members (get_card), 17
get_id, 18
get_id(), 22
get_id_board (get_id), 18
get_id_card (get_id), 18
get_list, 19
get_list_actions (get_list), 19
get_list_cards (get_list), 19
get_list_comments (get_list), 19
get_member_boards, 19
get_member_fields, 20
get_model (Deprecated), 15
get_my_boards, 20
get_resource, 21
get_resource(), 15, 17–20, 24, 27, 28
get_team, 23
get_team_boards (get_team), 23
get_team_members (get_team), 23
get_token, 24
get_token(), 15, 18, 22, 28

httr::GET(), 21, 22
httr::oauth1.0_token(), 25
httr::oauth_app(), 25
httr::oauth_endpoint(), 25
httr::RETRY(), 10, 14, 22, 32
httr::verbose(), 10, 14, 32

jsonlite::fromJSON(), 22

move_card, 26

post_model (Deprecated), 15
print.Trello_API_token, 26
put_model (Deprecated), 15

search_actions (search_resource), 27
search_boards (search_resource), 27
search_cards (search_resource), 27
search_members (search_resource), 27
search_model (Deprecated), 15
search_resource, 27
search_resource(), 15
search_teams (search_resource), 27

trello_get (Deprecated), 15
trello_get_token (Deprecated), 15
trello_search (Deprecated), 15
trello_search_actions (Deprecated), 15
trello_search_boards (Deprecated), 15
trello_search_cards (Deprecated), 15
trello_search_members (Deprecated), 15
trello_search_teams (Deprecated), 15
trelloR, 28

update_card, 28
update_card_checkbox
  (update_card_field), 29
update_card_date (update_card_field), 29
update_card_dropdown
  (update_card_field), 29
update_card_labels, 30
update_card_members, 30
update_card_number (update_card_field), 29
update_card_text (update_card_field), 29
update_checkitem, 31
update_field, 7, 13, 29, 31
update_resource, 32
update_resource(), 15, 26, 28–31