Package ‘rt’

April 27, 2020

Type  Package
Title  Interface to the ‘Request Tracker’ API
Description  Provides a programmatic interface to the ’Request Tracker’ (RT) HTTP API <https://rt-wiki.bestpractical.com/wiki/REST>. ’RT’ is a popular ticket tracking system.
Version  1.0.0
URL  https://github.com/nceas/rt
BugReports  https://github.com/nceas/rt/issues
Maintainer  Bryce Mecum <mecum@nceas.ucsb.edu>
License  MIT + file LICENSE
Imports  httr, stringr
Suggests  askpass, knitr, rmarkdown, testthat, tibble
RoxygenNote  7.1.0
VignetteBuilder  knitr
Encoding  UTF-8
LazyData  true
NeedsCompilation  no
Author  Bryce Mecum [aut, cre] (<https://orcid.org/0000-0002-0381-3766>), Irene Steves [ctb], National Center for Ecological Analysis and Synthesis [cph]
Repository  CRAN
Date/Publication  2020-04-27 17:30:02 UTC

\textbf{R topics documented:}

- check_login .......................................................... 2
- compact ............................................................... 3
- construct_newline_pairs ........................................... 3
- parse_rt_properties ............................................. 4
check_login: Check that the login request was successful or not

**Usage**

```r
check_login(response)
```
compact

**Arguments**

response  (httr::response) RT API login response

**Value**

(logical) TRUE if login was successful, errors out otherwise

---

**Description**

Compact list.

Remove all NULL entries from a list. From plyr::compact().

**Usage**

```r
compact(l)
```

**Arguments**

1  list

---

**construct_newline_pairs**

Construct a string for params suitable for passing into an RT request

**Description**

RT’s API, in a few cases, takes a body of key value pairs that are colon separated and each key value pair is newline separated. Each pair is also run through `compact` to remove NULL elements.

**Usage**

```r
construct_newline_pairs(params)
```

**Arguments**

params  (list) One or more key value pairs

**Value**

(character)
Parse typical RT properties as contained in an RT response body

**Description**

The code gives a basic idea of the format but it’s basically newline-separated key-value pairs with a `:` between them. e.g.,

```plaintext
id: queue/1 Name: General
```

**Usage**

```python
parse_rt_properties(body)
```

**Arguments**

- **body** (character) Response body from an `rt_response`

**Details**

- `id`: queue/1 `Name`: General

**Value**

- List of properties

---

Parse an RT ticket create response body and return the ticket ID

**Description**

This function essentially parses the text: "# Ticket 1 created."

**Usage**

```python
parse_ticket_create_body(body)
```

**Arguments**

- **body** (character) The ticket create response body

**Value**

- (numeric) The ticket ID
Parse the response body from a call to \texttt{rt_user_create}

**Description**

Parse the response body from a call to \texttt{rt_user_create}

**Usage**

\begin{verbatim}
parse_user_create_body(body)
\end{verbatim}

**Arguments**

- **body** (character)

**Value**

- \text{(numeric)} The user ID

---

Print an \texttt{rt_api} object

**Description**

Print an \texttt{rt_api} object

**Usage**

\begin{verbatim}
## S3 method for class 'rt_api'
print(x, ...)
\end{verbatim}

**Arguments**

- **x** object of class \texttt{rt_api}
- **...** Other arguments passed to \texttt{head}
Description

rt provides a programming interface to the Request Tracker API.

Details

Everything should be implemented and all functions should return a reasonably useful result that’s suitable for integrating into your workflows.

Setup:

Before you can do anything useful with this package, you’ll need to do three things:

1. Determine your base URL and set it using Sys.setenv(RT_BASE_URL="your url here"). In most cases, this will be the same as the URL of the page you use to log in to RT.
2. Determine and set your credentials. You can skip setting them if you like and skip to step 3 or you can set them using the RT_USER and RT_PASSWORD environmental variables via Sys.setenv. See rt_login for more.
3. Log in using R by calling rt_login. See rt_login for more.

A typical flow for setting up your R session to work with RT might look like this:

Sys.setenv(RT_BASE_URL = "http://example.com/rt",
          RT_USER = "me@example.com",
          RT_PASSWORD = "mypassword")

If you use RT a lot, you might consider putting code like the above in your .Renviron, minus the call to rt_login() so the environmental variables are available but you aren’t logging into RT every time you start R. See ?Startup for more information.

Available Functions:

General:

• rt_login
• rt_logout

Tickets:

• rt_ticket_search
• rt_ticket_create
• rt_ticket_edit
• rt_ticket_history
• rt_ticket_history_comment
• rt_ticket_history_reply
• rt_ticket_links
• rt_ticket_links_edit
• rt_ticket_merge
• rt_ticket_properties
rt_do_login

- rt_ticket_attachments
- rt_ticket_attachment
- rt_ticket_attachment_content

Users:
- rt_user_create
- rt_user_edit
- rt_user_properties

Queues:
- rt_queue_properties

---

**rt_do_login**

*Actually do the logging in part of logging in*

**Description**

Called by `rt_login` and `rt_login_interactive` to do the work of logging in

**Usage**

```
rt_do_login(user, password, ...)
```

**Arguments**

- `user` (character) Your username.
- `password` (character) Your password.
- `...` Other arguments passed to `rt_POST`

**Value**

(logical) Either returns TRUE if successful or errors out

---

**rt_GET**

*Get an RT response*

**Description**

Get an RT response and format it into an S3 object

**Usage**

```
rt_GET(url, raw = FALSE, ...)
```
Arguments

- **url** (character) The full RT URL
- **raw** (logical) Whether or not to return the raw response from \( \text{codeGET} \) (TRUE) or not (FALSE)
- **...** Other arguments passed to \( \text{GET} \)

Value

- \( \text{(rt_api)} \) The parsed response from RT

---

**rt_login**  
*Log in to RT*

**Description**

Use this to log into RT at the start of your session. Once you call this function and successfully log in, calls of other functions within this package will re-use your login information automatically.

**Usage**

```r
rt_login(
  user = Sys.getenv("RT_USER"),
  password = Sys.getenv("RT_PASSWORD"),
  ...
)
```

**Arguments**

- **user** (character) Your username.
- **password** (character) Your password.
- **...** Other arguments passed to \( \text{rt_POST} \)

**Details**

The value of \( \text{rt_base_url} \) should be the same address you use in your web browser to log into RT (i.e., the address of the log in page).

**Value**

Either \( \text{TRUE} \), invisibly, if logged in, or throws an error.
**rt_login_interactive**  

Log in to RT interactively

## Description

Wrapper for `rt_login` to interactively log into RT at the start of your session. Keeps your log-in information private.

## Usage

```r
rt_login_interactive(rt_base_url = Sys.getenv("RT_BASE"), ...)```

## Arguments

- `rt_base_url` (character) The base URL that hosts RT for your organization. Set the base URL in your R session using `Sys.getenv("RT_BASE_URL" = "https://server.name/rt/")`
- `...` Other arguments passed to `rt_do_login`

## Examples

```r
## Not run:
Sys.setenv(RT_BASE_URL = "https://demo.bestpractical.com")
rt_login_interactive()
## End(Not run)
```
Description

Use this to log out of RT at the end of your session. Note: restarting your R session will also log you out.

Usage

```r
tax_logout(...)```

Arguments

```r
... Other arguments passed to `rt_POST`
```

Value

`(rt_api) The parsed response from RT`

Examples

```r
## Not run:
# First, log in
rt_login()

# Then log out
rt_logout()

## End(Not run)```

Description

The RT API uses overrides default HTTP behavior with their own set of status codes, messages, and response formats. This function parses that custom implementation and presents it into something that’s easier to build a package with.

Usage

```r
rt_parse_response(response, verbose = FALSE)```
Arguments

response (character) Parsed response from content
verbose (logical) Optional, defaults to TRUE. Prints more information during parsing.

Details

For example, a response like:

"RT/4.4.3 200 Ok

# Ticket 2 created.

is turned into the list:

$status
[1] 200

$message
[1] "Ok"

$body
[1] "# Ticket 2 created."

Value

(list) List with named elements status, message, and body

Description

POST an RT request

Usage

rt_POST(url, raw = FALSE, ...)

Arguments

url (character) The full RT URL
raw (logical) Whether or not to return the raw response from POST (TRUE) or not (FALSE)
... Other arguments passed to POST

Value

(rt_api) The parsed response from RT
### rt_queue_properties

**Description**

Get the properties of a queue

**Usage**

```r
ticket_properties(queue, ...)
```

**Arguments**

- `queue` (character) The queue
- `...` Other arguments passed to `rt_GET`

**Value**

(list) A list of queue properties

**Examples**

```r
## Not run:
# By default, RT installations come with a General queue
# We can get its properties like this
rt_queue_properties("General")
## End(Not run)
```

### rt_ticket_attachment

**Description**

Retrieves attachment metadata. To get the attachment itself, see `rt_ticket_attachment_content`.

**Usage**

```r
ticket_attachment(ticket_id, attachment_id, ...)
```

**Arguments**

- `ticket_id` (numeric) The ticket identifier
- `attachment_id` (numeric) The attachment identifier
- `...` Other arguments passed to `rt_GET`
rt_ticket_attachments

Value

(rt_api) An rt_api object with the response

Examples

```r
## Not run:
# Before running rt_ticket_attachment, you'll probably want to get a list of
# the attachments for a given ticket, like:
attachments <- rt_ticket_attachments(1) # Ticket ID 1

# And then you can get information about a specific attachment:
rt_ticket_attachment(1, 3) # Attachment 3 on ticket 1

## End(Not run)
```

---

rt_ticket_attachments  Get a ticket's attachments

Description

Retrieves attachment metadata for a ticket in a tabular form.

Usage

```r
rt_ticket_attachments(ticket_id, ...)
```

Arguments

- `ticket_id` (numeric) The ticket identifier
- `...` Other arguments passed to `rt_POST`

Value

Either a `data.frame` or `tibble` of the attachments.

Examples

```r
## Not run:
# Given a ticket exists with id '2', we can get its attachments as a table
rt_ticket_attachments(2)

## End(Not run)
```
rt_ticket_attachment_content

Description

Gets the content of the specified attachment for further processing or manipulation. You’ll almost always want to call a second function like `content` to make the content of the attachment usable from R.

Usage

```
rt_ticket_attachment_content(ticket_id, attachment_id, ...)
```

Arguments

- `ticket_id` (numeric) The ticket identifier
- `attachment_id` (numeric) The attachment identifier
- `...` Other arguments passed to `rt_GET`

Value

`(rt_api) An `rt_api` object with the response

Examples

```
## Not run:
# First, get the attachment content which gives is the raw response
att <- rt_ticket_attachment_content(2, 1)

# Then process it directly in R
httr::content(att)

# Or write it to disk
out_path <- tempfile()
writeBin(httr::content(x, as = 'raw'), out_path)

## End(Not run)
```
rt_ticket_create

Create a ticket

Description

Create a ticket

Usage

rt_ticket_create(
    queue,
    requestor = NULL,
    subject = NULL,
    cc = NULL,
    admin_cc = NULL,
    owner = NULL,
    status = NULL,
    priority = NULL,
    initial_priority = NULL,
    final_priority = NULL,
    time_estimated = NULL,
    starts = NULL,
    due = NULL,
    text = NULL,
    custom_field = NULL,
    ...
)

Arguments

queue (character) The queue
requestor (character) Requestor email address
subject (character) Ticket subject
cc (character) Email address to cc
admin_cc (character) Admin email address to cc
owner (character) Owner username or email
status (character) Ticket status; typically "open", "new", "stalled", or "resolved"
priority (numeric) Ticket priority
initial_priority (numeric) Ticket initial priority
final_priority (numeric) Ticket final priority
time_estimated (character) Time estimated
starts (character) Starts
rt_ticket_edit

Edit a ticket

Description

Updates an existing ticket with new information.

Usage

rt_ticket_edit(
  ticket_id,
  queue = NULL,
  requestor = NULL,
  subject = NULL,
  cc = NULL,
  admin_cc = NULL,
  owner = NULL,
  status = NULL,
  priority = NULL,
  initial_priority = NULL,
  final_priority = NULL,
  time_estimated = NULL,
  starts = NULL,
  due = NULL,
  text = NULL,
  custom_field = NULL,
  ...)

Value

(numeric) The ID of the ticket

Examples

## Not run:
# We can create an empty ticket
rt_ticket_create("General")

# Or we can provide some of the fields
rt_ticket_create("General",
                requestor = "requestor@example.com",
                subject = "An example ticket")

## End(Not run)
rt_ticket_edit

custom_field = NULL,
...
)

Arguments

ticket_id   (numeric|character) The ticket number
queue      (character) The queue
requestor  (character) Requestor email address
subject    (character) Ticket subject
cc         (character) Email address to cc
admin_cc   (character) Admin email address to cc
owner      (character) Owner username or email
status     (character) Ticket status; typically "open", "new", "stalled", or "resolved"
priority   (numeric) Ticket priority
initial_priority (numeric) Ticket initial priority
final_priority (numeric) Ticket final priority
time_estimated (character) Time estimated
starts     (character) Starts
due        (character) Due date
text       (character) Ticket content; if multi-line, prefix every line with a blank
custom_field (vector) Takes a named vector of the custom field name and custom field value
...
Other arguments passed to rt_POST

Value

(numeric) The ID of the ticket

Examples

## Not run:
# First, create a ticket
ticket <- rt_ticket_create("General")

# Then we can update its fields
rt_ticket_edit(ticket,
requestor = "me@example.com",
s
subject = "My subject")

## End(Not run)
rt_ticket_history

Get a ticket’s history

Description
Get a ticket’s history

Usage
rt_ticket_history(ticket_id, format = "l", ...)

Arguments
- **ticket_id** (numeric) The ticket identifier
- **format** (character) The format of the ticket history response. Either s (ticket ID and subject) or l (full ticket metadata). Defaults to l.
- ... Other arguments passed to rt_GET

Value
(rt_api) An rt_api object with the response

Examples

```r
## Not run:
# Get the full ticket history for ticket 992
rt_ticket_history(992)

# Get just the ticket ID and subject for ticket 992
rt_ticket_history(992, format = "s")

## End(Not run)
```

---

rt_ticket_history_comment

Comment on a ticket

Description
Comment on a ticket

Usage
rt_ticket_history_comment(ticket_id, comment_text, ...)

---
rt_ticket_history_reply

Arguments

ticket_id  (numeric) The ticket identifier
comment_text  (character) Text that to add as a comment
...  Other arguments passed to rt_POST

Value

(numeric) The ID of the ticket

Examples

## Not run:
rt_ticket_history_comment(1, “Your comment here...”)

## End(Not run)

rt_ticket_history_reply

Reply to a ticket

Description

Reply to a ticket

Usage

rt_ticket_history_reply(  ticket_id,  text,  cc = NULL,  bcc = NULL,  time_worked = "0",  attachment_path = NULL,  ...  )

Arguments

ticket_id  (numeric) The ticket identifier
text  (character) Text that to add as a comment
cc  (character) Email for cc
bcc  (character) Email for bcc
time_worked  (character)
attachment_path  (character) Path to a file to upload
...  Other arguments passed to rt_POST
**rt_ticket_links**

**Value**

(numeric) The ID of the ticket

**Examples**

```
## Not run:
# Reply to ticket 11 with a courteous message
rt_ticket_history_reply(11,
     "Thank you.
     Have a great day!")

## End(Not run)
```

---

**rt_ticket_links**  
*Get a ticket’s links*

**Description**

Gets the ticket links for a single ticket. If applicable, the following fields will be returned: HasMember, ReferredToBy, DependedOnBy, MemberOf, RefersTo, and DependsOn.

**Usage**

```
rt_ticket_links(ticket_id, ...)
```

**Arguments**

- `ticket_id` (numeric) The ticket identifier
- `...` Other arguments passed to `rt_GET`

**Value**

(rt_api) An rt_api object with the response

**Examples**

```
## Not run:
# Assuming have a ticket with id 1007, we can get it links by calling
rt_ticket_links(1007)

## End(Not run)
```
rt_ticket_links_edit

Edit the links on a ticket

Description

Edit the links on a ticket

Usage

rt_ticket_links_edit(
   ticket_id,
   referred_to_by = NULL,
   depended_on_by = NULL,
   member_of = NULL,
   refers_to = NULL,
   depends_on = NULL,
   ...
)

Arguments

ticket_id (numeric) The ticket identifier
referred_to_by Tickets that are referred to
depended_on_by Tickets that are depended on
member_of Ticket groups?
refers_to Tickets that are referred to
depends_on Tickets that are depended on
...
Other arguments passed to rt_POST

Value

(numeric) The ID of the ticket

Examples

## Not run:
# Assuming we have tickets 20 and 21, we can make ticket 20 depend on ticket
# 21
rt_ticket_links_edit(20, depends_on = 21)

## End(Not run)
rt_ticket_merge  

**Merge two tickets**

**Description**
Merge two tickets

**Usage**
rt_ticket_merge(origin, into)

**Arguments**

- **origin** (character|numeric) Ticket ID to merge into into
- **into** (character|numeric) Ticket ID to merge origin into

**Value**
(numeric) The ID of ticket both tickets were merged into

**Examples**

```r
## Not run:
# First, create two tickets
ticket_one <- rt_ticket_create("General")
ticket_two <- rt_ticket_create("General")

# Then merge them together
ticket_merge(ticket_one, ticket_two)

## End(Not run)
```

---

rt_ticket_properties  

**Get a ticket’s properties**

**Description**
Retrieves ticket properties

**Usage**
rt_ticket_properties(ticket_id, ...)

**Arguments**

- **ticket_id** (numeric) The ticket identifier
- **...** Other arguments passed to rt_GET
### Description
Search RT for tickets using RT’s query syntax which is documented at [https://docs.bestpractical.com/rt/4.4.4/query_builder.html](https://docs.bestpractical.com/rt/4.4.4/query_builder.html).

### Usage

```r
rt_ticket_search(query, orderby = NULL, format = "l", fields = NULL, ...)
```

### Arguments

- `query` (character) Your query (See Details)
- `orderby` (character) How to order your search results. Should be a ticket property name preceded by either a + or a - character.
- `format` (character) Either i (ticket ID only), s (ticket ID and subject), or l (full ticket metadata). Defaults to l.
- `fields` (character) Comma-separated list of fields to include in the results.
- `...` Other arguments passed to `rt_GET`

### Details
The `query` parameter conforms to RT’s query syntax and requires you to build the query yourself. A query will have one or more parameters of the form `$FIELD='VALUE'` where `$FIELD` is an RT ticket property like Subject, Requestor, etc and `VALUE` (surrounded by single quotes) is the value to filter by. See Examples for examples.

### Value
Either a data.frame or tibble (when format is l or s) or a numeric vector when it’s i.
Examples

```r
## Not run:
# To return all un-owned tickets on a queue:
rt_ticket_search("Queue='General' AND (Status='new')")

# We can sort by date created, increasing
rt_ticket_search("Queue='General' AND (Status='new')",
  orderby = "+Created")

# If we just need a vector of ticket ids
rt_ticket_search("Queue='General' AND (Status='new')",
  orderby = "+Created",
  format = "i")

## End(Not run)
```

---

**rt_url**  
*Generate an RT API URL*

**Description**

Create an RT API URL based on the server URL and any arguments provided

**Usage**

```r
rt_url(..., query_params = NULL, base_url = Sys.getenv("RT_BASE_URL"))
```

**Arguments**

- `...` Parts of the URL to be joined by "/
- `query_params` (list) A named list of query parameters where the names of the list map to the query parameter names and the values of the list map to the query parameter values. e.g., `list(a=1)` maps to "?a=1".
- `base_url` (character) The base URL that hosts RT for your organization

---

**rt_user_agent**  
*Get the user agent for the package.*

**Description**

This is used by `rt_GET` and `rt_POST` to provide HTTP requests with an appropriate user agent.

**Usage**

```r
rt_user_agent()
```
Value

(character) The user agent string for the package

---

**rt_user_create**  
*Create a user*

Description

Create a user

Usage

```r
rt_user_create(
  name,
  password = NULL,
  email_address = NULL,
  real_name = NULL,
  organization = NULL,
  privileged = NULL,
  disabled = NULL,
  ...
)
```

Arguments

- `name` (character) Optional. User name
- `password` (character) The password
- `email_address` (character) Optional. User email
- `real_name` (character) Optional. User real name
- `organization` (character) Optional. User organization
- `privileged` (numeric) Optional. User privilege status
- `disabled` (numeric) Optional. User disabled status
- `...` Other arguments passed to `rt_POST`

Value

(numeric) The ID of the newly-created user

Examples

```r
## Not run:
# Create a barebones user with just a name
rt_user_create("Some Person")

# Create user that also has an email address
rt_user_create("Person", email_address = "person@example.com")

## End(Not run)
```
rt_user_edit

Description
Edit a user’s information.

Usage
rt_user_edit(
  user_id,
  password = NULL,
  name = NULL,
  email_address = NULL,
  real_name = NULL,
  organization = NULL,
  privileged = NULL,
  disabled = NULL,
  ...
)

Arguments
user_id (numeric) The ID of the User to edit
password (character) The password
name (character) Optional. User name
e-mail_address (character) Optional. User email
real_name (character) Optional. User real name
organization (character) Optional. User organization
privileged (numeric) Optional. User privilege status
disabled (numeric) Optional. User disabled status
... Other arguments passed to rt_POST

Value
The ID of the edited user

Examples
## Not run:
# First, create a user
user_id <- rt_user_create("Example", "password", "me@example.com")

# Then we can edit it
rt_user_edit(user_id, real_name = "Example User")

## End(Not run)
rt_user_properties  

Get a user's properties

Description

Get a user's properties

Usage

\texttt{rt_user_properties(user_id, \ldots)}

Arguments

user_id  \hspace{1cm} \text{(numeric)} The ID of the User to edit
\ldots \hspace{1cm} \text{Other arguments passed to \texttt{rt\_GET}}

Value

\text{(list)} A list of the user's properties

Examples

\texttt{## Not run:}
# Assuming we have a user with id 1, we can get its properties
\texttt{rt_user_properties(1)}

\texttt{## End(Not run)}

rt_version_string  

Get the version of the currently installed version of this package as a character vector

Description

Get the version of the currently installed version of this package as a character vector

Usage

\texttt{rt_version_string()}

Value

\text{(character)} The version is a character vector, e.g. "1.2.3"
stopforstatus

Description

Throw an error if the RT status code is an error status

Usage

stopforstatus(response)

Arguments

response (response) An \texttt{httr} response object

Value

Either nothing, or throws an error

tidy_long_search_result

tidy_long_search_result

Description

tidy_long_search_result

Usage

tidy_long_search_result(result)

Arguments

result (list) List of lists from search results

Value

A \texttt{data.frame} or \texttt{tibble}
try_tibble

Try to make a tibble

Description
Try to make a tibble

Usage
try_tibble(df, coerce = TRUE)

Arguments

- df (data.frame) The data.frame to attempt to coerce to a tibble
- coerce (logical) Whether or not to try coercion. Provided for upstream calling functions.

Value
Either a data.frame or a tibble

warn_user_edit_warnings
Warn if a user edit response body contains warnings

Description
Warn if a user edit response body contains warnings

Usage
warn_user_edit_warnings(body)

Arguments

- body (character)

Value
None.
Index

check_login, 2
compact, 3, 3
construct_newline_pairs, 3
content, 11, 14

GET, 8
head, 5

parse_rt_properties, 4
parse_ticket_create_body, 4
parse_user_create_body, 5
POST, 11
print.rt_api, 5

rt, 6
rt_do_login, 7, 9
rt_GET, 7, 12, 14, 18, 20, 22–24, 27
rt_login, 6, 7, 8, 9
rt_login_interactive, 7, 9
rt_logout, 6, 10
rt_parse_response, 10
rt_POST, 7, 8, 10, 11, 13, 16, 17, 19, 21, 24–26
rt_queue_properties, 7, 12
rt_ticket_attachment, 7, 12
rt_ticket_attachment_content, 7, 12, 14
rt_ticket_attachments, 7, 13
rt_ticket_create, 6, 15
rt_ticket_edit, 6, 16
rt_ticket_history, 6, 18
rt_ticket_history_comment, 6, 18
rt_ticket_history_reply, 6, 19
rt_ticket_links, 6, 20
rt_ticket_links_edit, 6, 21
rt_ticket_merge, 6, 22
rt_ticket_properties, 6, 22
rt_ticket_search, 6, 23
rt_url, 24
rt_user_agent, 24
rt_user_create, 5, 7, 25
rt_user_edit, 7, 26
rt_user_properties, 7, 27
rt_version_string, 27
stopforstatus, 28
tidy_long_search_result, 28
try_tibble, 29
warn_user_edit_warnings, 29