Package ‘gmailr’

November 29, 2021

Title Access the 'Gmail' 'RESTful' API

Version 1.0.1

Description An interface to the 'Gmail' 'RESTful' API. Allows access to your 'Gmail' messages, threads, drafts and labels.

License MIT + file LICENSE


BugReports https://github.com/r-lib/gmailr/issues

Depends R (>= 3.0.0)

Imports base64enc,
crayon,
gargle,
htr,
jsonlite,
lifecycle,
magrittr,
mime,
rematch2

Suggests covr,
knitr,
methods,
rmarkdown,
sodium,
testthat

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

R topics documented:

as.character.mime .......................................................... 2
gmailr ................................................................. 3
gm_attachment .......................................................... 3
gm_attachments ...................................................... 4
gm_auth_configure .................................................... 4
as.character.mime

Convert a mime object to character representation

Description

This function converts a mime object into a character vector
**gmailr**

**Usage**

```r
## S3 method for class 'mime'
as.character(x, newline = "\r\n", ...)
```

**Arguments**

- `x` object to convert
- `newline` value to use as newline character
- `...` further arguments ignored

---

**gmailr**  
*gmailr makes gmail access easy.*

---

**Description**

gmailr provides an interface to the gmail api [https://developers.google.com/gmail/api/](https://developers.google.com/gmail/api/)

---

**gm_attachment**  
*Retrieve an attachment to a message*

---

**Description**

This is a low level function to retrieve an attachment to a message by id of the attachment and message. Most users are better off using `gm_save_attachments()` to automatically save all the attachments in a given message.

**Usage**

```r
gm_attachment(id, message_id, user_id = "me")
```

**Arguments**

- `id` id of the attachment
- `message_id` id of the parent message
- `user_id` gmail user_id to access, special value of 'me' indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.messages.attachments/get](https://developers.google.com/gmail/api/reference/rest/v1/users.messages.attachments/get)

**See Also**

Other message: `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_trash_message()`, `gm_untrash_message()`
Examples

```r
## Not run:
my_attachment = attachment('a32e324b', '12345')
# save attachment to a file
gm_save_attachment(my_attachment, 'photo.jpg')

## End(Not run)
```

`gm_attachments`

Retrieve information about attachments

**Usage**

```r
gm_attachments(x, ...)
```

**Arguments**

- `x`: An object from which to retrieve the attachment information.
- `...`: other parameters passed to methods

**Value**

A data.frame with the filename, type, size and id of each attachment in the message.

`gm_auth_configure`

Edit auth configuration

**Description**

These functions give more control over and visibility into the auth configuration than `gm_auth()` does. `gm_auth_configure()` lets the user specify their own:

- OAuth app, which is used when obtaining a user token. See the vignette How to get your own API credentials for more. If the user does not configure these settings, internal defaults are used. `gm_oauth_app()` retrieves the currently configured OAuth app.

**Usage**

```r
gm_auth_configure(
  key = "",
  secret = "",
  path = Sys.getenv("GMAILR_APP"),
  appname = "gmailr",
  ...
)
```

```r
gm_oauth_app()
```
Arguments

key consumer key, also sometimes called the client ID
secret consumer secret, also sometimes called the client secret.
path JSON downloaded from Google Cloud Platform Console, containing a client id (aka key) and secret, in one of the forms supported for the `txt` argument of `jsonlite::fromJSON()` (typically, a file path or JSON string).
appname name of the application. This is not used for OAuth, but is used to make it easier to identify different applications.
app OAuth app, in the sense of `http::oauth_app()`

Value

- `gm_auth_configure()`: An object of R6 class `gargle::AuthState`, invisibly.
- `gm_oauth_app()`: the current user-configured `http::oauth_app()`.

See Also

Other auth functions: `gm_deauth()`, `gm_scopes()`

Examples

```R
## Not run:
# see the current user-configured OAuth app (probably `NULL`)
gm_oauth_app()
if (require(httr)) {
  # store current state, so we can restore
  original_app <- gm_oauth_app()
  
  # bring your own app via client id (aka key) and secret
  google_app <- http::oauth_app(
    "my-awesome-google-api-wrapping-package",
    key = "123456789.apps.googleusercontent.com",
    secret = "abcdefghijklmnopqrstuvwxyz"
  )
  gm_auth_configure(app = google_app)

  # confirm current app
  gm_oauth_app()

  # restore original state
  gm_auth_configure(app = original_app)
  gm_oauth_app()
}

# bring your own app via JSON downloaded from Google Developers Console
gm_auth_configure(
  path = "/path/to/the/JSON/you/downloaded/from/google/dev/console.json"
)
## End(Not run)
```
**gm_body**

Get the body text of a message or draft

**Description**

Get the body text of a message or draft

**Usage**

```r
gm_body(x, ...)```

**Arguments**

- `x` the object from which to retrieve the body
- `...` other parameters passed to methods

**Examples**

```r
## Not run:
gm_body(my_message)
gm_body(my_draft)
## End(Not run)
```

---

**gm_create_draft**

Create a draft from a mime message

**Description**

Create a draft from a mime message

**Usage**

```r
gm_create_draft(mail, user_id = "me")```

**Arguments**

- `mail` mime mail message created by mime
- `user_id` gmail user_id to access, special value of 'me' indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/create](https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/create)

**Examples**

```r
## Not run:
gm_create_draft(gm_mime(From="you@me.com", To="any@one.com", Subject="hello", "how are you doing"))
## End(Not run)
```
**gm_create_label**  
*Create a new label*

**Description**  
Function to create a label.

**Usage**

```r
gm_create_label(
  name,
  label_list_visibility = c("show", "hide", "show_unread"),
  message_list_visibility = c("show", "hide"),
  user_id = "me"
)
```

**Arguments**

- `name` name to give to the new label
- `label_list_visibility` The visibility of the label in the label list in the Gmail web interface.
- `message_list_visibility` The visibility of messages with this label in the message list in the Gmail web interface.
- `user_id` gmail user_id to access, special value of ’me’ indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.labels/create](https://developers.google.com/gmail/api/reference/rest/v1/users.labels/create)

**See Also**

Other label: `gm_delete_label()`, `gm_labels()`, `gm_label()`, `gm_update_label()`

---

**gm_deauth**  
*Clear current token*

**Description**

Clears any currently stored token. The next time gmair needs a token, the token acquisition process starts over, with a fresh call to `gm_auth()` and, therefore, internally, a call to `gargle::token_fetch()`. Unlike some other packages that use gargle, gmair is not usable in a de-authorized state. Therefore, calling `gm_deauth()` only clears the token, i.e. it does NOT imply that subsequent requests are made with an API key in lieu of a token.

**Usage**

```r
gm_deauth()
```
gm_delete_draft

See Also

Other auth functions: \texttt{gm_auth_configure()}, \texttt{gm_scopes()}

Examples

\begin{verbatim}
## Not run:
gm_deauth()

## End(Not run)
\end{verbatim}

gm_delete_draft \hspace{1cm} \textit{Permanently delete a single draft}

Description

Function to delete a given draft by id. This cannot be undone!

Usage

\texttt{gm_delete_draft(id, user_id = "me")}

Arguments

- \texttt{id}: message id to access
- \texttt{user_id}: gmail user id to access, special value of 'me' indicates the authenticated user.

References

\url{https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/delete}

See Also

Other draft: \texttt{gm_drafts()}, \texttt{gm_draft()}, \texttt{gm_send_draft()}

Examples

\begin{verbatim}
## Not run:
delete_draft('12345')

## End(Not run)
\end{verbatim}
gm_delete_label

Permanently delete a label

Description
Function to delete a label by id. This cannot be undone!

Usage
gm_delete_label(id, user_id = "me")

Arguments
id
label id to retrieve
user_id
gmail user_id to access, special value of 'me' indicates the authenticated user.

References
https://developers.google.com/gmail/api/reference/rest/v1/users.labels/delete

See Also
Other label: gm_create_label(), gm_labels(), gm_label(), gm_update_label()

gm_delete_message

Permanently delete a single message

Description
Function to delete a given message by id. This cannot be undone!

Usage
gm_delete_message(id, user_id = "me")

Arguments
id
message id to access
user_id
gmail user_id to access, special value of 'me' indicates the authenticated user.

References
https://developers.google.com/gmail/api/reference/rest/v1/users.messages/delete

See Also
Other message: gm_attachment(), gm_import_message(), gm_insert_message(), gm_messages(),
gm_message(), gm_modify_message(), gm_save_attachments(), gm_save_attachment(), gm_send_message(),
gm_trash_message(), gm_untrash_message()
Examples

## Not run:
gm_delete_message('12345')

## End(Not run)

---

**gm_delete_thread**

*Permanently delete a single thread.*

Description

Function to delete a given thread by id. This cannot be undone!

Usage

```r
gm_delete_thread(id, user_id = "me")
```

Arguments

- `id` : thread id to access
- `user_id` : gmail user_id to access, special value of 'me' indicates the authenticated user.

References

[https://developers.google.com/gmail/api/reference/rest/v1/users.threads/delete](https://developers.google.com/gmail/api/reference/rest/v1/users.threads/delete)

See Also

Other thread: `gm_modify_thread()`, `gm_threads()`, `gm_thread()`, `gm_trash_thread()`, `gm_untrash_thread()`

Examples

## Not run:
delete_thread(12345)

## End(Not run)

---

**gm_draft**

*Get a single draft*

Description

Function to retrieve a given draft by '<-

Usage

```r
gm_draft(id, user_id = "me", format = c("full", "minimal", "raw"))
```
Arguments

- **id**: draft id to access
- **user_id**: gmail user_id to access, special value of 'me' indicates the authenticated user.
- **format**: format of the draft returned

References

https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/get

See Also

Other draft: `gm_delete_draft()`, `gm_drafts()`, `gm_send_draft()`

Examples

```r
## Not run:
my_draft = gm_draft('12345')

## End(Not run)
```

---

```r
gm_drafts

Get a list of drafts
```

Description

Get a list of drafts possibly matching a given query string.

Usage

```r
gm_drafts(num_results = NULL, page_token = NULL, user_id = "me")
```

Arguments

- **num_results**: the number of results to return.
- **page_token**: retrieve a specific page of results
- **user_id**: gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/list

See Also

Other draft: `gm_delete_draft()`, `gm_draft()`, `gm_send_draft()`

Examples

```r
## Not run:
my_drafts = gm_drafts()

first_10_drafts = gm_drafts(10)

## End(Not run)
```
gm_has_token  

**Description**

Reports whether gmailr has stored a token, ready for use in downstream requests.

**Usage**

```
gm_has_token()
```

**Value**

Logical.

**Examples**

```
gm_has_token()
```

---

gm_history  

**Description**

Retrieves the history results in chronological order

**Usage**

```
gm_history(
  start_history_id = NULL,
  num_results = NULL,
  label_id = NULL,
  page_token = NULL,
  user_id = "me"
)
```

**Arguments**

- `start_history_id`: the point to start the history. The historyId can be obtained from a message, thread or previous list response.
- `num_results`: the number of results to return, max per page is 100
- `label_id`: filter history only for this label
- `page_token`: retrieve a specific page of results
- `user_id`: gmail user_id to access, special value of ‘me’ indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.history/list](https://developers.google.com/gmail/api/reference/rest/v1/users.history/list)
## gm_id

Get the id of a gmailr object

### Description
Get the id of a gmailr object

### Usage

```r
gm_id(x, ...)  
```

### Arguments

- `x`: the object from which to retrieve the id
- `...`: other parameters passed to methods
- `what`: the type of id to return

### Examples

```r
## Not run:
gm_id(my_message)  
gm_id(my_draft)  
## End(Not run)
```

## gm_import_message

Import a message into the gmail mailbox from a mime message

### Description
Import a message into the gmail mailbox from a mime message

### Usage

```r
gm_import_message(  
  mail,  
  label_ids,  
  type = c("multipart", "media", "resumable"),  
  internal_date_source = c("dateHeader", "recievedTime"),  
  user_id = "me"  
)
```
Arguments

- **mail**: mime mail message created by mime
- **label_ids**: optional label ids to apply to the message
- **type**: the type of upload to perform
- **internal_date_source**: whether to date the object based on the date of the message or when it was received by gmail.
- **user_id**: gmail user_id to access, special value of `me` indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages/import

See Also

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_insert_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_trash_message()`, `gm_untrash_message()`

Examples

```r
## Not run:
gm_insert_message(gm_mime(From="you@me.com", To="any@one.com", Subject="hello", "how are you doing?
```

```r
## End(Not run)
```

---

**gm_insert_message**

Insert a message into the gmail mailbox from a mime message

Description

Insert a message into the gmail mailbox from a mime message

Usage

```r
gm_insert_message(  
  mail,  
  label_ids,  
  type = c("multipart", "media", "resumable"),  
  internal_date_source = c("dateHeader", "recievedTime"),  
  user_id = "me"  
)
```
**gm_label**

**Arguments**

- **mail**: mime mail message created by mime
- **label_ids**: optional label ids to apply to the message
- **type**: the type of upload to perform
- **internal_date_source**: whether to date the object based on the date of the message or when it was received by gmail.
- **user_id**: gmail user_id to access, special value of `me` indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.messages/insert](https://developers.google.com/gmail/api/reference/rest/v1/users.messages/insert)

**See Also**

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_send_message()`, `gm_trash_message()`, `gm_untrash_message()`

**Examples**

```r
## Not run:
gm_insert_message(gm_mime(From="you@me.com", To="any@one.com", Subject="hello", "how are you doing?"))
## End(Not run)
```

---

**gm_label**

*Get a specific label*

**Description**

Get a specific label by id and user_id.

**Usage**

```r
gm_label(id, user_id = "me")
```

**Arguments**

- **id**: label id to retrieve
- **user_id**: gmail user_id to access, special value of `me` indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.labels/get](https://developers.google.com/gmail/api/reference/rest/v1/users.labels/get)

**See Also**

Other label: `gm_create_label()`, `gm_delete_label()`, `gm_labels()`, `gm_update_label()`
**gm_labels**

*Get a list of all labels*

**Description**

Get a list of all labels for a user.

**Usage**

```r
gm_labels(user_id = "me")
```

**Arguments**

- `user_id`
  - Gmail user_id to access, special value of `me` indicates the authenticated user.

**References**

[https://developers.google.com/gmail/api/reference/rest/v1/users.labels/list](https://developers.google.com/gmail/api/reference/rest/v1/users.labels/list)

**See Also**

Other label: `gm_create_label()`, `gm_delete_label()`, `gm_label()`, `gm_update_label()`

**Examples**

```r
## Not run:
my_labels = gm_labels()
## End(Not run)
```

---

**gm_last_response**

*Response from the last query*

**Description**

Response from the last query

**Usage**

```r
gm_last_response()
```
**gm_message**

### Description

Function to retrieve a given message by id

### Usage

```r
gm_message(
  id,
  user_id = "me",
  format = c("full", "metadata", "minimal", "raw")
)
```

### Arguments

- **id**: message id to access
- **user_id**: gmail user_id to access, special value of `me` indicates the authenticated user.
- **format**: format of the message returned

### References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages

### See Also

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_messages()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_trash_message()`, `gm_untrash_message()`

### Examples

```r
## Not run:
my_message = gm_message(12345)
## End(Not run)
```

---

**gm_messages**

### Description

Get a list of messages possibly matching a given query string.
Usage

```r
gm_messages(
    search = NULL,
    num_results = NULL,
    label_ids = NULL,
    include_spam_trash = NULL,
    page_token = NULL,
    user_id = "me"
)
```

Arguments

- `search`: query to use, same format as gmail search box.
- `num_results`: the number of results to return.
- `label_ids`: restrict search to given labels
- `include_spam_trash`: boolean whether to include the spam and trash folders in the search
- `page_token`: retrieve a specific page of results
- `user_id`: gmail user_id to access, special value of `"me"` indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages/list

See Also

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_trash_message()`

Examples

```r
## Not run:
#Search for R, return 10 results using label 1 including spam and trash folders
my_messages = gm_messages("R", 10, "label_1", TRUE)

## End(Not run)
```

---

`gm_mime` Create a mime formatted message object

Description

These functions create a MIME message. They can be created atomically using `gm_mime()` or iteratively using the various accessors.
Usage

gm_mime(..., attr = NULL, body = NULL, parts = list())

## S3 method for class 'mime'
gm_to(x, val, ...)

## S3 method for class 'mime'
gm_from(x, val, ...)

## S3 method for class 'mime'
gm_cc(x, val, ...)

## S3 method for class 'mime'
gm_bcc(x, val, ...)

## S3 method for class 'mime'
gm_subject(x, val, ...)

gm_text_body(
  mime,
  body,
  content_type = "text/plain",
  charset = "utf-8",
  encoding = "quoted-printable",
  format = "flowed",
  ...
)

gm_html_body(
  mime,
  body,
  content_type = "text/html",
  charset = "utf-8",
  encoding = "base64",
  ...
)

gm_attach_part(mime, part, id = NULL, ...)

gm_attach_file(mime, filename, type = NULL, id = NULL, ...)

Arguments

... additional parameters to put in the attr field
attr attributes to pass to the message
body Message body.
parts mime parts to pass to the message
x the object whose fields you are setting
val the value to set, can be a vector, in which case the values will be joined by ",".
content_type  The content type to use for the body.
charset  The character set to use for the body.
encoding  The transfer encoding to use for the body.
format  The mime format to use for the body.
part  Message part to attach
id  The content ID of the attachment
filename  name of file to attach
type  mime type of the attached file

Examples

# using the field functions
msg = gm_mime() %>%
gm_from("james.f.hester@gmail.com") %>%
gm_to("asdf@asdf.com") %>%
gm_text_body("Test Message")

# alternatively you can set the fields using gm_mime(), however you have
# to use properly formatted MIME names
msg = gm_mime(From="james.f.hester@gmail.com",
               To="asdf@asdf.com") %>%
               gm_html_body("<b>Test</b> Message")

Description

Function to modify the labels on a given message by id. Note you need to use the label ID as arguments to this function, not the label name.

Usage

gm_modify_message(id, add_labels = NULL, remove_labels = NULL, user_id = "me")

Arguments

id  message id to access
add_labels  label IDs to add to the specified message
remove_labels  label IDs to remove from the specified message
user_id  gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages/modify

See Also

Other message: gm_attachment(), gm_delete_message(), gm_import_message(), gm_insert_message(), gm_messages(), gm_message(), gm_save_attachments(), gm_save_attachment(), gm_send_message(), gm_trash_message(), gm_untrash_message()
gm_modify_thread

Modify the labels on a thread

Description

Function to modify the labels on a given thread by id.

Usage

```r
gm_modify_thread(id, add_labels = character(0), remove_labels = character(0), user_id = "me")
```

Arguments

- `id`: thread id to access
- `add_labels`: labels to add to the specified thread
- `remove_labels`: labels to remove from the specified thread
- `user_id`: gmail user_id to access, special value of 'me' indicates the authenticated user.

References

- [https://developers.google.com/gmail/api/reference/rest/v1/users.threads/modify](https://developers.google.com/gmail/api/reference/rest/v1/users.threads/modify)

See Also

Other thread: `gm_delete_thread()`, `gm_threads()`, `gm_thread()`, `gm_trash_thread()`, `gm_untrash_thread()`

Examples

```r
## Not run: modify_thread(12345, add_labels='label_1') modify_thread(12345, remove_labels='label_1') #add and remove at the same time modify_thread(12345, add_labels='label_2', remove_labels='label_1')
## End(Not run)
```
gm_profile  Get info on current gmail profile

Description
Reveals information about the profile associated with the current token.

Usage
gm_profile(user_id = "me", verbose = TRUE)

Arguments
user_id  gmail user_id to access, special value of 'me' indicates the authenticated user.
verbose  Logical, indicating whether to print informative messages (default TRUE).

Value
A list of class gmail_profile.

See Also
Wraps the getProfile endpoint:
  • https://developers.google.com/gmail/api/reference/rest/v1/users/getProfile

Examples
## Not run:
gm_profile()

## more info is returned than is printed
prof <- gm_profile()
prof[["historyId"]]

## End(Not run)

gm_save_attachment  Save the attachment to a file

Description
This is a low level function that only works on attachments retrieved with gm_attachment(). To save an attachment directly from a message see gm_save_attachments(), which is a higher level interface more suitable for most uses.

Usage
gm_save_attachment(x, filename)
gm_save_attachments

Arguments

x attachment to save
filename location to save to

See Also

Other message: gm_attachment(), gm_delete_message(), gm_import_message(), gm_insert_message(),
gm_messages(), gm_message(), gm_modify_message(), gm_save_attachments(), gm_send_message(),
gm_trash_message(), gm_untrash_message()

Examples

## Not run:
my_attachment = attachment('a32e324b', '12345')
# save attachment to a file
save_attachment(my_attachment, 'photo.jpg')

## End(Not run)

gm_save_attachments Save attachments to a message

Description

Function to retrieve and save all of the attachments to a message by id of the message.

Usage

gm_save_attachments(x, attachment_id = NULL, path = ".", user_id = "me")

Arguments

x message with attachment
attachment_id id of the attachment to save, if none specified saves all attachments
path where to save the attachments
user_id gmail user_id to access, special value of ’me’ indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages.attachments/get

See Also

Other message: gm_attachment(), gm_delete_message(), gm_import_message(), gm_insert_message(),
gm_messages(), gm_message(), gm_modify_message(), gm_save_attachment(), gm_send_message(),
gm_trash_message(), gm_untrash_message()
Examples

## Not run:
# save all attachments
save_attachments(my_message)
# save a specific attachment
save_attachments(my_message, 'a32e324b')

## End(Not run)

### gm_scopes

#### Authorize gmailr

**Description**

Authorize gmailr to view and manage your Gmail projects. This function is a wrapper around `gargle::token_fetch()`.

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant gmailr permission to operate on your behalf with Google Gmail. By default, with your permission, these user credentials are cached in a folder below your home directory, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

If you are interacting with R within a browser (applies to RStudio Server, RStudio Workbench, and RStudio Cloud), you need a variant of this flow, known as out-of-band auth ("oob"). If this does not happen automatically, you can request it yourself with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

#### Usage

```r
gm_scopes()

gm_auth(
  email = gm_default_email(),
  path = NULL,
  scopes = "full",
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)
```

**Arguments**

- `email` Optional. Allows user to target a specific Google identity. If specified, this is used for token lookup, i.e. to determine if a suitable token is already available in the cache. If no such token is found, `email` is used to pre-select the targetted Google identity in the OAuth chooser. Note, however, that the email associated with a token when it’s cached is always determined from the token itself, never from this argument. Use NA or FALSE to match nothing and force the OAuth dance in the browser. Use TRUE to allow email auto-discovery, if exactly one matching token is found in the cache. Specify just the domain with a glob pattern, e.g. "*@example.com", to create code that "just works" for both alice@example.com and bob@example.com. Defaults to the option named "gargle_oauth_email", retrieved by `gargle_oauth_email()`.
path

JSON identifying the service account, in one of the forms supported for the `txt` argument of `jsonlite::fromJSON()` (typically, a file path or JSON string).

scopes

One or more Gmail API scope to use, one of 'labels', 'send', 'readonly', 'compose', 'insert', 'modify', 'metadata', 'settings_basic', 'settings_sharing' or 'full' (default: 'full'). See https://developers.google.com/gmail/api/auth/scopes for details on the permissions for each scope. and `gm_scopes()` to return a vector of the available scopes.

cache

Specifies the OAuth token cache. Defaults to the option named "gargle_oauth_cache", retrieved via `gargle_oauth_cache()`.

use_oob

Whether to prefer "out of band" authentication. Defaults to the option named "gargle_oob_default", retrieved via `gargle_oob_default()`.

token

A token with class `Token2.0` or an object of `httr`'s class `request`, i.e. a token that has been prepared with `httr::config()` and has a `Token2.0` in the `auth_token` component.

details

Most users, most of the time, do not need to call `gm_auth()` explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice. However, when necessary, this function allows the user to explicitly:

- Declare which Google identity to use, via an email address. If there are multiple cached tokens, this can clarify which one to use. It can also force Gmail to switch from one identity to another. If there's no cached token for the email, this triggers a return to the browser to choose the identity and give consent. You can specify just the domain by using a glob pattern. This means that a script containing `email = "*@example.com"` can be run without further tweaks on the machine of either `alice@example.com` or `bob@example.com`.
- Use a service account token or workload identity federation.
- Bring their own `Token2.0`.
- Specify non-default behavior re: token caching and out-of-bound authentication.
- Customize scopes.

For details on the many ways to find a token, see `gargle::token_fetch()`. For deeper control over auth, use `gm_auth_configure()` to bring your own OAuth app or API key. Read more about gargle options, see `gargle::gargle_options`.

see also

Other auth functions: `gm_auth_configure()`, `gm_deauth()`

examples

```r
## Not run:
## load/refresh existing credentials, if available
## otherwise, go to browser for authentication and authorization
gm_auth()

## force use of a token associated with a specific email
gm_auth(email = "jim@example.com")

## force a menu where you can choose from existing tokens or
## choose to get a new one
```
gm_send_draft

Send a draft

description

Send a draft to the recipients in the To, CC, and Bcc headers.

Usage

gm_send_draft(draft, user_id = "me")

Arguments

draft: the draft to send

user_id: gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.drafts/send

See Also

Other draft: gm_delete_draft(), gm_drafts(), gm_draft()

Examples

## Not run:
draft <- gm_create_draft(gm_mime(From="you@me.com", To="any@one.com",
Subject="hello", "how are you doing?"))
gm_send_draft(draft)

## End(Not run)
Send a message from a mime message

Usage

```r
gm_send_message(
  mail,
  type = c("multipart", "media", "resumable"),
  thread_id = NULL,
  user_id = "me"
)
```

Arguments

- **mail**: mime mail message created by mime
- **type**: the type of upload to perform
- **thread_id**: the id of the thread to send from.
- **user_id**: gmail user_id to access, special value of 'me' indicates the authenticated user.

References

[https://developers.google.com/gmail/api/reference/rest/v1/users.messages/send](https://developers.google.com/gmail/api/reference/rest/v1/users.messages/send)

See Also

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_trash_message()`, `gm_untrash_message()`

Examples

```r
## Not run:
gm_send_message(gm_mime(from="you@me.com", to="any@one.com",
                         subject="hello", "how are you doing?"))
## End(Not run)
```
gm_threads

**gm_thread**  
*Get a single thread*

**Description**
Function to retrieve a given thread by id

**Usage**

gm_thread(id, user_id = "me")

**Arguments**
id  
thread id to access
user_id  
gmail user_id to access, special value of ’me’ indicates the authenticated user.

**References**

https://developers.google.com/gmail/api/reference/rest/v1/users.threads

**See Also**

Other thread: gm_delete_thread(), gm_modify_thread(), gm_threads(), gm_trash_thread(), gm_untrash_thread()

**Examples**

```r
## Not run:
my_thread = gm_thread(12345)
## End(Not run)
```

---

gm_threads

**Get a list of threads**

**Description**
Get a list of threads possibly matching a given query string.

**Usage**

```r
gm_threads(
  search = NULL,
  num_results = NULL,
  page_token = NULL,
  label_ids = NULL,
  include_spam_trash = NULL,
  user_id = "me"
)
```

Arguments

- search: query to use, same format as Gmail search box.
- num_results: the number of results to return.
- page_token: retrieve a specific page of results.
- label_ids: restrict search to given labels.
- include_spam_trash: boolean whether to include the spam and trash folders in the search.
- user_id: Gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.threads/list

See Also

Other thread: gm_delete_thread(), gm_modify_thread(), gm_thread(), gm_trash_thread(), gm_untrash_thread()

Examples

```r
## Not run:
my_threads = gm_threads()
first_10_threads = gm_threads(10)
## End(Not run)
```

---

**gm_to**

*Methods to get values from message or drafts*

Description

Methods to get values from message or drafts.

Usage

```r
gm_to(x, ...)
gm_from(x, ...)
gm_cc(x, ...)
gm_bcc(x, ...)
gm_date(x, ...)
gm_subject(x, ...)
```

Arguments

- `x`: the object from which to get or set the field.
- `...`: other parameters passed to methods.
gm_token

Produce configured token

Description
For internal use or for those programming around the Gmail API. Returns a token pre-processed with `httr::config()`. Most users do not need to handle tokens "by hand" or, even if they need some control, `gm_auth()` is what they need. If there is no current token, `gm_auth()` is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via `gm_deauth()`, `gm_token()` returns NULL.

Usage

```r
gm_token()
```

Value
A request object (an S3 class provided by `httr`).

Examples

```r
## Not run:
gm_token()
## End(Not run)
```

gm_trash_message

Send a single message to the trash

Description
Function to trash a given message by id. This can be undone by `gm_untrash_message()`.

Usage

```r
gm_trash_message(id, user_id = "me")
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>message id to access</td>
</tr>
<tr>
<td>user_id</td>
<td>gmail user_id to access, special value of 'me' indicates the authenticated user.</td>
</tr>
</tbody>
</table>

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages/trash

See Also
Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_untrash_message()`
### gm_trash_thread

Send a single thread to the trash

#### Description

Function to trash a given thread by id. This can be undone by gm_untrash_thread().

#### Usage

```r
gm_trash_thread(id, user_id = "me")
```

#### Arguments

- `id` thread id to access
- `user_id` gmail user_id to access, special value of 'me' indicates the authenticated user.

#### References

[https://developers.google.com/gmail/api/reference/rest/v1/users.threads/trash](https://developers.google.com/gmail/api/reference/rest/v1/users.threads/trash)

#### See Also

Other thread: gm_delete_thread(), gm_modify_thread(), gm_threads(), gm_thread(), gm_untrash_thread()

#### Examples

```r
## Not run:
trash_thread(12345)
## End(Not run)
```

---

### gm_untrash_message

Remove a single message from the trash

#### Description

Function to trash a given message by id. This can be undone by gm_untrash_message().

#### Usage

```r
gm_untrash_message(id, user_id = "me")
```
gm_untrash_thread

Arguments

**id**
message id to access

**user_id**
gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.messages/trash

See Also

Other message: `gm_attachment()`, `gm_delete_message()`, `gm_import_message()`, `gm_insert_message()`, `gm_messages()`, `gm_message()`, `gm_modify_message()`, `gm_save_attachments()`, `gm_save_attachment()`, `gm_send_message()`, `gm_trash_message()`

Examples

```r
## Not run:
gm_untrash_message('12345')
## End(Not run)
```

---

**gm_untrash_thread**

Remove a single thread from the trash.

Description

Function to untrash a given thread by id. This can reverse the results of a previous `trash_thread()`.

Usage

```r
gm_untrash_thread(id, user_id = "me")
```

Arguments

**id**
thread id to access

**user_id**
gmail user_id to access, special value of 'me' indicates the authenticated user.

References

https://developers.google.com/gmail/api/reference/rest/v1/users.threads/untrash

See Also

Other thread: `gm_delete_thread()`, `gm_modify_thread()`, `gm_threads()`, `gm_thread()`, `gm_trash_thread()`

Examples

```r
## Not run:
untrash_thread(12345)
## End(Not run)
```
gm_update_label

Update a existing label.

Description
Get a specific label by id and user_id. update_label_patch is identical to update_label but the latter uses HTTP PATCH to allow partial update.

Usage
```python
gm_update_label(id, label, user_id = "me")
gm_update_label_patch(id, label, user_id = "me")
```

Arguments
- `id`: label id to update
- `label`: the label fields to update
- `user_id`: gmail user_id to access, special value of `me` indicates the authenticated user.

References
- https://developers.google.com/gmail/api/reference/rest/v1/users.labels/update
- https://developers.google.com/gmail/api/reference/rest/v1/users.labels/patch

See Also
- Other label: gm_create_label(), gm_delete_label(), gm_labels(), gm_label()
- Other label: gm_create_label(), gm_delete_label(), gm_labels(), gm_label()

quoted_printable_encode

Encode text using quoted printable

Description
Does no do any line wrapping of the output to 76 characters Implementation derived from the perl MIME::QuotedPrint

Usage
```python
quoted_printable_encode(data)
```

Arguments
- `data`: data to encode

References
- http://search.cpan.org/~gaas/MIME-Base64-3.14/QuotedPrint.pm
Index

* auth functions
  gm_auth_configure, 4
gm_deauth, 7
gm_scopes, 24

* draft
  gm_delete_draft, 8
gm_draft, 10
gm_drafts, 11
gm_send_draft, 26

* label
  gm_create_label, 7
gm_delete_label, 9
gm_label, 15
gm_labels, 16
gm_update_label, 33

* low-level API functions
  gm_token, 30

* message
  gm_attachment, 3
gm_delete_message, 9
gm_import_message, 13
gm_insert_message, 14
gm_message, 17
gm_messages, 17
gm_modify_message, 20
gm_save_attachment, 22
gm_save_attachments, 23
gm_send_message, 27
gm_trash_message, 30
gm_untrash_message, 31

* mime
  gm_mime, 18

* thread
  gm_delete_thread, 10
gm_modify_thread, 21
gm_thread, 28
gm_threads, 28
gm_untrash_thread, 31

as.character.mime, 2
gargle::AuthState, 5
gargle::gargle_options, 25
gargle::token_fetch(), 7, 24, 25
gargle_oauth_cache(), 25
gargle_oauth_email(), 24
gargle_oob_default(), 25
gm_attach_file (gm_mime), 18
gm_attach_part (gm_mime), 18
gm_attachment, 3, 9, 14, 15, 17, 18, 20, 23,
  27, 30, 32
gm_attachment(), 22
gm_attentions, 4
gm_auth (gm_scopes), 24
gm_auth(), 4, 7, 30
gm_auth_configure, 4, 8, 25
gm_auth_configure(), 25
gm_bcc (gm_to), 29
gm_bcc_mime (gm_mime), 18
gm_body, 6
gm_cc (gm_to), 29
gm_cc_mime (gm_mime), 18
gm_create_draft, 6
gm_create_label, 7, 9, 15, 16, 33
gm_date (gm_to), 29
gm_deauth, 5, 7, 25
gm_deauth(), 30
gm_delete_draft, 8, 11, 26
gm_delete_label, 7, 9, 15, 16, 33
gm_delete_message, 3, 9, 14, 15, 17, 18, 20,
  23, 27, 30, 32
gm_delete_thread, 10, 21, 28, 29, 31, 32
gm_draft, 8, 10, 11, 26
gm_drafts, 8, 11, 11, 26
gm_from (gm_to), 29
gm_from_mime (gm_mime), 18
gm_has_token, 12
gm_history, 12
gm_html_body (gm_mime), 18
gm_id, 13
gm_import_message, 3, 9, 13, 15, 17, 18, 20,
  23, 27, 30, 32
gm_insert_message, 3, 9, 14, 14, 17, 18, 20,
  23, 27, 30, 32
gm_label, 7, 9, 15, 16, 33
gm_labels, 7, 9, 15, 16, 33