Package ‘rytstat’

December 7, 2021

Title Work with 'YouTube API'
Version 0.2.1
Description Provide function for get data from 'YouTube Data API'
<https://developers.google.com/youtube/v3/docs/>, 'YouTube Analytics API'
<https://developers.google.com/youtube/analytics/reference/> and
'YouTube Reporting API' <https://developers.google.com/youtube/reporting/v1/reports>.
Imports cli, dplyr, gargle, snakecase, stringr, tidyr, pbapply, httr,
withr, rlang
URL https://selesnow.github.io/rytstat/docs/
BugReports https://github.com/selesnow/rytstat/issues
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.1.2
NeedsCompilation no
Author Alexey Seleznev [aut, cre] (<https://orcid.org/0000-0003-0410-7385>),
Netpeak [cph]
Maintainer Alexey Seleznev <selesnow@gmail.com>
Repository CRAN
Date/Publication 2021-12-07 21:50:02 UTC

R topics documented:

  rytstat-package ................................................................. 2
  ryt_auth ................................................................. 2
  ryt_auth_configure ...................................................... 4
  ryt_deauth ................................................................. 6
  ryt_get_analytics ......................................................... 7
  ryt_get_channels ........................................................ 8
  ryt_get_channel_activities .............................................. 9
  ryt_get_comments .......................................................... 10
Description


Author(s)

Maintainer: Alexey Seleznev <selesnow@gmail.com> (ORCID)

Other contributors:

- Netpeak [copyright holder]

See Also

Useful links:

- https://selesnow.github.io/rytstat/docs/
- Report bugs at https://github.com/selesnow/rytstat/issues
**Description**

Authorize rytstat to view and manage your YouTube Account. 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 rytstat permission to operate on your behalf with YouTube. 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
tyte_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)
```

**Arguments**

- `email` Optional. Allows user to target a specific Google identity.
- `path` Path to JSON file with identifying the service account
- `cache` Specifies the OAuth token cache.
- `use_oob` Whether to prefer "out of band" authentication.
- `token` A token with class `Token2.0` or an object of

**Details**

Most users, most of the time, do not need to call `ryt_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 rytstat 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.
ryt_auth_configure

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

Value

Token2.0

See Also

Other auth functions: `ryt_auth_configure()`, `ryt_deauth()`

Examples

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

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

## force a menu where you can choose from existing tokens or
## choose to get a new one
ryt_auth(email = NA)

## -----------------------
## use own OAuth client app
ryt_auth_configure(
  path = "path/to/your/oauth_client.json"
)

ryt_auth(email = "yourname@example.com")

## End(Not run)
```

Description

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

- OAuth app, which is used when obtaining a user token.
- API key. If rytstat is de-authorized via `ryt_deauth()`, all requests are sent with an API key in lieu of a 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. `ryt_oauth_app()` and `ryt_api_key()` retrieve the currently configured OAuth app and API key, respectively.
ryt_auth_configure

Usage

ryt_auth_configure(app, path, api_key)

ryt_auth_cache_path()

ryt_open_auth_cache_folder()

ryt_api_key()

Arguments

app OAuth app, in the sense of http::oauth_app().

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).

api_key API key.

Value

• ryt_auth_configure(): An object of R6 class gargle::AuthState, invisibly.
• ryt_oauth_app(): the current user-configured http::oauth_app().
• ryt_api_key(): the current user-configured API key.

See Also

Other auth functions: ryt_auth(), ryt_deauth()

Examples

## Not run:
# see and store the current user-configured OAuth app (probably `NULL`)    
(original_app <- ryt_oauth_app())

# see and store the current user-configured API key (probably `NULL`)  
(original_api_key <- ryt_api_key())

if (require(httr)) {
  # bring your own app via client id (aka key) and secret
  google_app <- http::oauth_app(
    "my-awesome-google-api-wrapping-package",
    key = "YOUR_CLIENT_ID_GOES_HEREx",
    secret = "YOUR_SECRET.GOES_HERE"
  )
  google_key <- "YOUR_API_KEY"
  ryt_auth_configure(app = google_app, api_key = google_key)

  # confirm the changes
}
ryt_oauth_app()
ryt_api_key()

# bring your own app via JSON downloaded from Google Developers Console
# this file has the same structure as the JSON from Google
ryt_auth_configure(path = app_path)

# confirm the changes
ryt_oauth_app()
}

# restore original auth config
gs4_auth_configure(app = original_app, api_key = original_api_key)

## End(Not run)

---

### ryt_deauth

**Suspend authorization**

**Description**

Put rytstat into a de-authorized state. Instead of sending a token, rytstat will send an API key. This can be used to access public resources for which no Google sign-in is required. This is handy for using rytstat in a non-interactive setting to make requests that do not require a token. It will prevent the attempt to obtain a token interactively in the browser. The user can configure their own API key via `ryt_auth_configure()` and retrieve that key via `ryt_api_key()`. In the absence of a user-configured key, a built-in default key is used.

**Usage**

```r
ryt_deauth()
```

**Value**

only suspend authorization

**See Also**

Other auth functions: `ryt_auth_configure()`, `ryt_auth()`
**ryt_get_analytics**  
*Get statistics from 'YouTube Analytics API'*

### Description

The YouTube Analytics API enables you to generate custom reports containing YouTube Analytics data. The API supports reports for channels and for content owners.

### Usage

```r
github-url
docs
downloads

table
```

```r
ryt_get_analytics(  
  start_date = Sys.Date() - 14,  
  end_date = Sys.Date(),  
  metrics = c("views", "estimatedMinutesWatched", "averageViewDuration",  
             "averageViewPercentage", "subscribersGained"),  
  dimensions = "day",  
  filters = NULL  
)
```

### Arguments

- **start_date**: The start date for fetching YouTube Analytics data. The value should be in YYYY-MM-DD format.
- **end_date**: The end date for fetching YouTube Analytics data. The value should be in YYYY-MM-DD format.
- **metrics**: Character vector of YouTube Analytics metrics, such as views or likes, dislikes. See the documentation for channel reports or a list of the reports that you can retrieve and the metrics available in each report. The Metrics document contains definitions for all of the metrics.
- **dimensions**: Character vector of YouTube Analytics dimensions, such as video or ageGroup, gender. The Dimensions document contains definitions for all of the dimensions.
- **filters**: Character vector of filters that should be applied when retrieving YouTube Analytics data. The documentation for channel reports reports identifies the dimensions that can be used to filter each report, and the Dimensions document defines those dimensions.

### Value

A tibble with analytics data

### Examples

```r
## Not run:  
# auth  
ryt_auth()
```
# get list of your videos
videos <- ryt_get_videos()

# function for loading video stat
get_videos_stat <- function(video_id) {
  data <- ryt_get_analytics(
    metrics = c('views', 'likes', 'dislikes', 'comments', 'shares'),
    filters = stringr::str_glue('video=={video_id}')
  )
  if ( nrow(data) > 0 ) {
    data <- mutate(data, video_id = video_id)
  }
}

# load video stat
video_stat <- purrr::map_df(videos$id_video_id, get_videos_stat)

# join stat with video metadata
video_stat <- left_join(video_stat, videos, by = c("video_id" = "id_video_id")) %>%
  select(video_id, title, day, views, likes, dislikes, comments, shares)

## End(Not run)

---

**ryt_get_channels**

*Get channel info from 'YouTube API'*

**Description**

Get channel info from 'YouTube API'

**Usage**

```r
tyet_get_channels(
  fields = c("contentDetails", "id", "snippet", "statistics", "status", "topicDetails")
)
```

**Arguments**

- `fields` Fields of channel metadata, see API documentation.
**ryt_get_channel_activities**

**Value**

a tibble with channel metadata

**Examples**

```r
## Not run:
channels <- ryt_get_channels()

## End(Not run)
```

---

**ryt_get_channel_activities**

*Returns a list of channel activity events*

**Description**

Returns a list of channel activity events

**Usage**

`ryt_get_channel_activities(fields = c("contentDetails", "id", "snippet"))`

**Arguments**

- **fields** Fields of channel metadata, see [API documentation](#).

**Value**

a tibble with channel activities

**Examples**

```r
## Not run:
channel_activities <- ryt_get_channel_activities()

## End(Not run)
```
ryt_get_comments  

Returns a list of comment threads of video or channel

Description

Returns a list of comment threads of video or channel

Usage

ryt_get_comments(
  video_id = NULL,
  channel_id = NULL,
  text_format = c("plainText", "html")
)

Arguments

video_id  
YouTUbe Video ID

channel_id  
YouTube Channel ID

text_format  
Set this parameter’s value to html or plainText to instruct the API to return the comments left by users in html formatted or in plain text. The default value is plainText

Value

tibble with comments

See Also

Reporting API Documentation.

Examples

## Not run:
# all comments
comments <- ryt_get_comments()

# videos comments
video_comments <- ryt_get_comments(video_id = 'fW7gGS^G78')

## End(Not run)
**ryt_get_playlists**

*Get playlist from 'YouTube API'*

**Description**

Get playlist from 'YouTube API'

**Usage**

```r
ryt_get_playlists(
  fields = c("contentDetails", "id", "localizations", "player", "snippet", "status")
)
```

**Arguments**

- `fields`: Fields of playlist metadata, see [API documentation](#).

**Value**

tibble with playlist metadata

---

**ryt_get_playlist_items**

*Get playlist items data on 'YouTube'*

**Description**

Get playlist items data on 'YouTube'

**Usage**

```r
ryt_get_playlist_items(
  playlist_id,
  fields = c("contentDetails", "id", "snippet", "status"),
  cl = NULL
)
```

**Arguments**

- `playlist_id`: Playlist ID, see `ryt_get_playlists`.
- `fields`: Fields of video metadata, see [API documentation](#).
- `cl`: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
**Value**

A tibble with playlist items details

---

**ryt_get_report_types**  Returns a list of report types that the channel or content owner can retrieve. Each item in the list contains an id property, which identifies the report’s ID, and you need this value to schedule a reporting job.

---

**Description**

By creating a reporting job, you are instructing YouTube to generate that report on a daily basis. The report is available within 24 hours of the time that the job is created.

Each resource in the response contains an id property, which specifies the ID that YouTube uses to uniquely identify the job. You need that ID to retrieve the list of reports that have been generated for the job or to delete the job.

**Usage**

```r
ryt_get_report_types()

ryt_create_job(
  report_type = c("channel_annotations_a1", "channel_basic_a2", "channel_cards_a1",
  "channel_combined_a2", "channel_demographics_a1", "channel_device_os_a2",
  "channel_end_screens_a1", "channel_playback_location_a2", "channel_province_a2",
  "channel_sharing_service_a1", "channel_playback_location_a2", "channel_province_a2",
  "channel_sharing_service_a1", "channel_subtitles_a2", "channel_traffic_source_a2",
  "playlist_basic_a1", "playlist_device_os_a1", "playlist_playback_location_a1",
  "playlist_province_a1", "playlist_traffic_source_a1")
)

ryt_get_job_list()

ryt_get_report_list(
  job_id,
  created_after = NULL,
  start_time_at_or_after = NULL,
  start_time_before = NULL
)

ryt_get_report(download_url)

ryt_get_report_metadata(job_id, report_id)

ryt_delete_job(job_id)
```
Arguments

- **report_type** - The type of report that the job creates. The property value corresponds to the id of a reportType as retrieved from the `ryt_get_report_types` function.
- **job_id** - The ID that YouTube uses to uniquely identify the job that is being deleted. Use `ryt_get_job_list`.
- **created_after** - If specified, this parameter indicates that the API response should only contain reports created after the specified date and time, including new reports with backfilled data. Note that the value pertains to the time that the report is created and not the dates associated with the returned data. The value is a timestamp in RFC3339 UTC "Zulu" format, accurate to microseconds. Example: "2015-10-02T15:01:23.045678Z".
- **start_time_at_or_after** - This parameter indicates that the API response should only contain reports if the earliest data in the report is on or after the specified date. Whereas the createdAfter parameter value pertains to the time the report was created, this date pertains to the data in the report. The value is a timestamp in RFC3339 UTC "Zulu" format, accurate to microseconds. Example: "2015-10-02T15:01:23.045678Z".
- **start_time_before** - This parameter indicates that the API response should only contain reports if the earliest data in the report is before the specified date. Whereas the createdAfter parameter value pertains to the time the report was created, this date pertains to the data in the report. The value is a timestamp in RFC3339 UTC "Zulu" format, accurate to microseconds. Example: "2015-10-02T15:01:23.045678Z".
- **download_url** - download URL, you can get it by `ryt_get_report_list` or `ryt_get_report_metadata`.
- **report_id** - The ID that YouTube uses to uniquely identify the report that is being retrieved. Use `ryt_get_report_list`.

Value

- `ryt_get_report_types`: tibble with report types
- `ryt_reports_create_job`: No return value, called for side effects
- `ryt_get_job_list`: tibble with jobs metadata
- `ryt_get_report_list`: tibble with reports metadata
- `ryt_get_report`: tibble with report data
- `ryt_get_report_metadata`: list with report metadata
- `ryt_reports_delete_job`: No return value, called for side effects

See Also

- Reporting API Documentation: Method reportTypes.list.
- Reporting API Documentation: Method jobs.create
- Reporting API Documentation: Method jobs.list
- Reporting API Documentation: Method jobs.reports.list
- Reporting API Documentation: Data Model
**ryt_get_videos**

Get list of your videos from 'YouTube'

### Description
Get list of your videos from 'YouTube'

### Usage
```r
ryt_get_videos()
```

### Value
- tibble with video list
ryt_get_video_details  Get detail data of your videos on 'YouTube'

Description
Get detail data of your videos on 'YouTube'

Usage
ryt_get_video_details(
  video_id,
  fields = c("contentDetails", "fileDetails", "id", "liveStreamingDetails", "localizations", "player", "processingDetails", "recordingDetails", "snippet", "statistics", "status", "suggestions", "topicDetails"),
  cl = NULL
)

Arguments
video_id Video ID, see ryt_get_videos.
fields Fields of video metadata, see API documentation.
cl A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value
tibble with video details

Examples
## Not run:
# get all videos
videos <- ryt_get_videos()

# get videos metadata
videos_details <- ryt_get_video_details(
  video_id = videos$id_video_id
)

## End(Not run)
ryt_has_token  
*Is there a token on hand?*

**Description**

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

**Usage**

```r
ryt_has_token()
```

**Value**

Logical.

**See Also**

Other low-level API functions: `ryt_token()`

---

`ryt_token`  
*Produce configured token*

**Description**

For internal use or for those programming around the YouTube 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, `ryt_auth()` is what they need. If there is no current token, `ryt_auth()` is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via `ryt_deauth()`, `ryt_token()` returns NULL.

**Usage**

```r
ryt_token()
```

**Value**

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

**See Also**

Other low-level API functions: `ryt_has_token()`
ryt_user

Get info on current user

Description
Reveals the email address of the user associated with the current token. If no token has been loaded yet, this function does not initiate auth.

Usage
ryt_user()

Value
An email address or, if no token has been loaded, NULL.

See Also
gargle::token_userinfo(), gargle::token_email(), gargle::token_tokeninfo()
Index

* auth functions
  ryt_auth, 2
  ryt_auth_configure, 4
  ryt_deauth, 6
* low-level API functions
  ryt_has_token, 16
  ryt_token, 16
* reporting api functions
  ryt_get_report_types, 12

gargle::AuthState, 5

gargle::gargle_options, 4
gargle::token_email, 17

gargle::token_fetch, 3, 4

gargle::token_tokeninfo, 17

gargle::token_userinfo, 17

httr, 16

tstr::config, 16

httr::oauth_app, 5

jsonlite::fromJSON, 5

makeCluster, 11, 15

ryt_api_key (ryt_auth_configure), 4
ryt_api_key, 4, 6
ryt_auth, 2, 5, 6
ryt_auth(), 4, 16
ryt_auth_cache_path
  (ryt_auth_configure, 4
  ryt_auth_configure, 4, 4, 6
  ryt_auth_configure(), 4, 6

ryt_create_job (ryt_get_report_types), 12
ryt_deauth, 4, 5, 6
ryt_deauth(), 4, 16
ryt_delete_job (ryt_get_report_types),
  12
ryt_get_analytics, 7
ryt_get_channel_activities, 9

ryt_get_channels, 8
ryt_get_comments, 10
ryt_get_job_list, 13
ryt_get_job_list
  (ryt_get_report_types), 12
ryt_get_playlist_items, 11
ryt_get_playlists, 11, 11
ryt_get_report (ryt_get_report_types),
  12
ryt_get_report_list, 13
ryt_get_report_list
  (ryt_get_report_types), 12
ryt_get_report_metadata, 13
ryt_get_report_metadata
  (ryt_get_report_types), 12
ryt_get_report_types, 12, 13
ryt_get_video_details, 15
ryt_get_videos, 14, 15
ryt_has_token, 16, 16
ryt_oauth_app (ryt_auth_configure), 4
ryt_open_auth_cache_folder
  (ryt_auth_configure, 4
ryt_token, 16, 16
ryt_user, 17
rytstat (rytstat-package), 2
rytstat-package, 2

Token2.0, 3, 4