Package ‘tuber’

June 26, 2019

Title Client for the YouTube API
Version 0.9.8
Maintainer Gaurav Sood <gsood07@gmail.com>
Description Get comments posted on YouTube videos, information on how many times a video has been liked, search for videos with particular content, and much more. You can also scrape captions from a few videos. To learn more about the YouTube API, see <https://developers.google.com/youtube/v3/>.

URL http://github.com/soodoku/tuber
BugReports http://github.com/soodoku/tuber/issues
Depends R (>= 3.4.0)
License MIT + file LICENSE
LazyData true
Imports httr, plyr, dplyr, purrr, jsonlite, utils
VignetteBuilder knitr
Suggests knitr (>= 1.11), testthat, rmarkdown, xml2, lintr
RoxygenNote 6.0.1.9000
Encoding UTF-8
NeedsCompilation no
Author Gaurav Sood [aut, cre], Kate Lyons [ctb], John Muschelli [ctb]
Repository CRAN
Date/Publication 2019-06-26 09:50:16 UTC

R topics documented:

- delete_captions ................................................................. 3
- delete_channel_sections .................................................. 3
- delete_comments ............................................................. 4
<table>
<thead>
<tr>
<th>Topics Documented</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete_playlists</td>
<td>5</td>
</tr>
<tr>
<td>delete_playlist_items</td>
<td>5</td>
</tr>
<tr>
<td>delete_videos</td>
<td>6</td>
</tr>
<tr>
<td>get_all_channel_video_stats</td>
<td>7</td>
</tr>
<tr>
<td>get_all_comments</td>
<td>8</td>
</tr>
<tr>
<td>get_captions</td>
<td>8</td>
</tr>
<tr>
<td>get_channel_stats</td>
<td>9</td>
</tr>
<tr>
<td>get_comments</td>
<td>10</td>
</tr>
<tr>
<td>get_comment_threads</td>
<td>11</td>
</tr>
<tr>
<td>get_playlists</td>
<td>12</td>
</tr>
<tr>
<td>get_playlist_items</td>
<td>13</td>
</tr>
<tr>
<td>get_related_videos</td>
<td>14</td>
</tr>
<tr>
<td>get_stats</td>
<td>15</td>
</tr>
<tr>
<td>get_subscriptions</td>
<td>16</td>
</tr>
<tr>
<td>get_video_details</td>
<td>17</td>
</tr>
<tr>
<td>list_abuse_report_reasons</td>
<td>18</td>
</tr>
<tr>
<td>list_captions</td>
<td>19</td>
</tr>
<tr>
<td>list_caption_tracks</td>
<td>20</td>
</tr>
<tr>
<td>list_channel_activities</td>
<td>21</td>
</tr>
<tr>
<td>list_channel_resources</td>
<td>22</td>
</tr>
<tr>
<td>list_channel_sections</td>
<td>23</td>
</tr>
<tr>
<td>list_channel_videos</td>
<td>24</td>
</tr>
<tr>
<td>list_guidecats</td>
<td>25</td>
</tr>
<tr>
<td>list_langs</td>
<td>26</td>
</tr>
<tr>
<td>list_my_videos</td>
<td>26</td>
</tr>
<tr>
<td>list_regions</td>
<td>27</td>
</tr>
<tr>
<td>list_videocats</td>
<td>28</td>
</tr>
<tr>
<td>list_videos</td>
<td>29</td>
</tr>
<tr>
<td>tuber</td>
<td>30</td>
</tr>
<tr>
<td>tuber_check</td>
<td>30</td>
</tr>
<tr>
<td>tuber_DELETE</td>
<td>30</td>
</tr>
<tr>
<td>tuber_GET</td>
<td>31</td>
</tr>
<tr>
<td>tuber_POST</td>
<td>31</td>
</tr>
<tr>
<td>upload_caption</td>
<td>32</td>
</tr>
<tr>
<td>upload_video</td>
<td>33</td>
</tr>
<tr>
<td>yt_check_token</td>
<td>34</td>
</tr>
<tr>
<td>yt_oauth</td>
<td>34</td>
</tr>
<tr>
<td>yt_search</td>
<td>35</td>
</tr>
<tr>
<td>yt_topic_search</td>
<td>37</td>
</tr>
</tbody>
</table>

Index 38
delete_captions  
*Delete a Particular Caption Track*

**Description**
Delete a Particular Caption Track

**Usage**
delete_captions(id = NULL, ...)

**Arguments**
id  String. Required. id of the caption track that is being retrieved
... Additional arguments passed to tuber_DELETE.

**References**
https://developers.google.com/youtube/v3/docs/captions/delete

**Examples**

```bash
## Not run:
# Set API token via yt_oauth() first
delete_captions(id = "y3ELxCMe3ISISZ6izkWv5Vv7Pu8pA")
## End(Not run)
```

delete_channel_sections  
*Delete Channel Sections*

**Description**
Delete a Channel Section

**Usage**
delete_channel_sections(id = NULL, ...)

**Arguments**
id  Required. ID of the channel section.
... Additional arguments passed to tuber_DELETE.
delete_comments

References

https://developers.google.com/youtube/v3/docs/channelSections/delete

Examples

## Not run:

# Set API token via yt_oauth() first
delete_channel_sections(c(channel_id = "UCRw8Bi2wMLmfgAgWm903cA"))

## End(Not run)

-----------------------
delete_comments Delete a Particular Comment-----------------------

Description

Delete a Particular Comment

Usage

delete_comments(id = NULL, ...)

Arguments

id String. Required. id of the comment being retrieved
...

Additional arguments passed to tuber_DELETE.

References

https://developers.google.com/youtube/v3/docs/comments/delete

Examples

## Not run:

# Set API token via yt_oauth() first
delete_comments(id = "y3E1xEME3ISz6izkWVT5GvxjPu8pA")

## End(Not run)
**delete_playlists**  
*Delete a Playlist*

**Description**
Delete a Playlist

**Usage**
```
delete_playlists(id = NULL, ...)
```

**Arguments**
- **id**: String. Required. id of the playlist that is to be deleted
- **...**: Additional arguments passed to `tuber_DELETE`.

**References**
https://developers.google.com/youtube/v3/docs/playlists/delete

**Examples**
```
## Not run:

# Set API token via yt_oauth() first

delete_playlists(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

---

**delete_playlist_items**  
*Delete a Playlist Item*

**Description**
Delete a Playlist Item

**Usage**
```
delete_playlist_items(id = NULL, ...)
```

**Arguments**
- **id**: String. Required. id of the playlist item that is to be deleted
- **...**: Additional arguments passed to `tuber_DELETE`. 
delete_videos

References

https://developers.google.com/youtube/v3/docs/playlistItems/delete

Examples

```r
## Not run:

# Set API token via yt_oauth() first

delete_playlist_items(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

delte_videos  

Delete a Video

Description

Delete a Video

Usage

```r
delete_videos(id = NULL, ...)
```

Arguments

- **id**: String. Required. id of the video that is to be deleted
- **...**: Additional arguments passed to `tuber_DELETE`.

References

https://developers.google.com/youtube/v3/docs/playlistItems/delete

Examples

```r
## Not run:

# Set API token via yt_oauth() first

delete_videos(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```
**get_all_channel_video_stats**

*Get statistics on all the videos in a Channel*

**Description**

Get statistics on all the videos in a Channel

**Usage**

```python
get_all_channel_video_stats(channel_id = NULL, mine = FALSE, ...)
```

**Arguments**

- **channel_id**
  Character. Id of the channel
- **mine**
  Boolean. TRUE if you want to fetch stats of your own channel. Default is FALSE.
- **...**
  Additional arguments passed to `tuber_GET`.

**Value**

Nested named list with top element names: kind, etag, id, snippet (list of details of the channel including title). If the channel_id is mistyped or there is no information, an empty list is returned.

**References**

https://developers.google.com/youtube/v3/docs/channels/list

**Examples**

```r
## Not run:

# Set API token via `yt_oauth()` first

get_all_channel_video_stats(channel_id="UCxOhDvtaoXDAB336AolWs3A")
get_all_channel_video_stats(channel_id="UCMtFA184ehT5Y5E9Xo")  # Incorrect channel ID

## End(Not run)
```
get_all_comments  
*Get all the comments for a video including replies*

**Description**

Get all the comments for a video including replies

**Usage**

```r
get_all_comments(video_id = NULL, ...)
```

**Arguments**

- `video_id`: string; Required. video_id: video ID.
- `...`: Additional arguments passed to `tuber_GET`.

**Value**

a data.frame with the following columns: `authorDisplayName`, `authorProfileImageUrl`, `authorChannelUrl`, `authorChannelId`, `value`, `videoId`, `textDisplay`, `canRate`, `viewerrating`, `likecount`, `publishedAt`, `updatedAt`, `id`, `moderationstatus`, `parentid`

**References**

https://developers.google.com/youtube/v3/docs/commentThreads/list

**Examples**

```r
## Not run:

# Set API token via yt_oauth() first

get_all_comments(video_id = "a-UQz7fqR3w")

## End(Not run)
```

get_captions  
*Get Particular Caption Track*

**Description**

For getting captions from the v3 API, you must specify the id resource. Check `list_caption_tracks` for more information.

**Usage**

```r
get_captions(id = NULL, lang = "en", format = "sbv", ...)
```
get_channel_stats

Arguments

id     String. Required. id of the caption track that is being retrieved
lang   Optional. Default is en.
format Optional. Default is sbv.
...    Additional arguments passed to tuber_GET.

Value

String.

References

https://developers.google.com/youtube/v3/docs/captions/download

Examples

## Not run:

```r
# Set API token via yt_oauth() first

getCaptions(id = "y3E1XcEME3lSIZ6izkWVT5GvxjPu8pA")

## End(Not run)
```

get_channel_stats  Get statistics of a Channel

Description

Get statistics of a Channel

Usage

```r
get_channel_stats(channel_id = NULL, mine = NULL, ...)
list_my_channel(...)
```

Arguments

channel_id    Character. Id of the channel
mine          Boolean. TRUE if you want to fetch stats of your own channel. Default is NULL.
...           Additional arguments passed to tuber_GET.
get_comments

Value

nested named list with top element names: kind, etag, id, snippet (list of details of the channel including title, statistics, etc.).

If the channel_id is mistyped or there is no information, an empty list is returned.

References

https://developers.google.com/youtube/v3/docs/channels/list

Examples

```r
## Not run:

# Set API token via yt_oauth() first

get_channel_stats(channel_id="UCmtFAi84ehTSYSE9XoHefig")
get_channel_stats(channel_id="UCmtFAi84ehTSYSE9Xo") # Incorrect channel ID

## End(Not run)
```

get_comments

Get Comments

Description

Get Comments

Usage

```r
get_comments(filter = NULL, part = "snippet", max_results = 100,
              text_format = "html", page_token = NULL, simplify = TRUE, ...)
```

Arguments

- **filter**
  - string: Required. named vector of length 1 potential names of the entry in the vector: comment_id, comment ID, parent_id, parent ID.
- **part**
  - Comment resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., "id, snippet", "id", etc. Default: snippet.
- **max_results**
  - Maximum number of items that should be returned. Integer. Optional. Can be between 20 and 100. Default is 100.
- **text_format**
  - Data Type: Character. Default is "html". Only takes "html" or "plaintext". Optional.
- **page_token**
  - Specific page in the result set that should be returned. Optional.
- **simplify**
  - Data Type: Boolean. Default is TRUE. If TRUE, the function returns a data frame. Else a list with all the information returned.
- **...**
  - Additional arguments passed to tuber_GET.
**get_comment_threads**

**Value**

Nested named list. The entry `items` is a list of comments along with meta information. Within each of the `items` is an item `snippet` which has an item `topLevelComment$snippet$textDisplay` that contains the actual comment.

When `filter` is `comment_id`, and `simplify` is `TRUE`, and there is a correct comment id, it returns a data.frame with the following cols: `id`, `authorDisplayName`, `authorProfileImageUrl`, `authorChannelUrl`, `value`, `textDisplay`, `canRate`, `viewerrating`, `likeCount`, `publishedAt`, `updatedAt`

**References**

https://developers.google.com/youtube/v3/docs/comments/list

**Examples**

```r
## Not run:

# Set API token via `yt_oauth()` first

get_comments(filter = c(comment_id = "z13dh13j5rr0wbnzq04cifrhtuywp14hsdk"))
get_comments(filter = c(parent_id = "z13ds5xjyq3zzptx84chlkbx2yh3ezxtc0k"))
get_comments(filter = c(comment_id = "z13dh13j5rr0wbnzq04cifrhtuywp14hsdk, z13dh13j5rr0wbnzq04cifrhtuywp14hsdk"))

## End(Not run)
```

---

**get_comment_threads**  
*Get Comments Threads*

**Description**

Get Comments Threads

**Usage**

```r
get_comment_threads(filter = NULL, part = "snippet", text_format = "html", simplify = TRUE, max_results = 100, page_token = NULL, ...)
```

**Arguments**

- **filter**
  - string: Required. named vector of length 1 potential names of the entry in the vector: `video_id`: video ID. `channel_id`: channel ID. `thread_id`: comma-separated list of comment thread IDs `threads_related_to_channel`: channel ID.

- **part**
  - Comment resource requested. Required. Comma separated list of one or more of the following: `id`, `snippet`. e.g., `"id", "snippet", "id"`, etc. Default: `snippet`.  

get_playlists

Description

Get Playlists

Usage

get_playlists(filter = NULL, part = "snippet", max_results = 50,
hl = NULL, page_token = NULL, simplify = TRUE, ...)
Arguments

filter  string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: ID of the channel playlist_id: YouTube playlist ID.

part  Required. One of the following: contentDetails, id, localizations, player, snippet, status. Default: contentDetails.

max_results  Maximum number of items that should be returned. Integer. Optional. Can be between 0 and 50. Default is 50.

hl  Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.

page_token  specific page in the result set that should be returned, optional

simplify  Data Type: Boolean. Default is TRUE. If TRUE and if part requested is contentDetails, the function returns a data.frame. Else a list with all the information returned.

Additional arguments passed to tuber_GET.

Value

playlists When simplify is TRUE, a data.frame with 4 columns is returned: kind, etag, id, contentDetails.itemCount

References

https://developers.google.com/youtube/v3/docs/playlists/list

Examples

## Not run:

# Set API token via yt_oauth() first

get_playlists(filter=c(channel_id="UCMtFAi84ehTSYSE9XoHefig"))
get_playlists(filter=c(channel_id="UCMtFAi84ehTSYSE9X"))  # incorrect Channel ID

## End(Not run)

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}

get_playlist_items  Get Playlist Items

Description

Get Playlist Items

Usage

get_playlist_items(filter = NULL, part = "contentDetails",
max_results = 50, video_id = NULL, page_token = NULL, simplify = TRUE,
...}
get_related_videos

Arguments

- **filter**: string; Required. named vector of length 1 potential names of the entry in the vector: *item_id*: comma-separated list of one or more unique playlist item IDs. *playlist_id*: YouTube playlist ID.
- **part**: Required. Comma separated string including one or more of the following: *contentDetails*, *id*, *snippet*, *status*. Default: *contentDetails*.
- **max_results**: Maximum number of items that should be returned. Integer. Optional. Default is 50. If over 50, all the results are returned.
- **video_id**: Optional. request should return only the playlist items that contain the specified video.
- **page_token**: specific page in the result set that should be returned, optional
- **simplify**: returns a data.frame rather than a list.
- **...**: Additional arguments passed to `tuber_GET`.

Value

playlist items

References

https://developers.google.com/youtube/v3/docs/playlists/list

Examples

```r
## Not run:

# Set API token via `yt_oauth()` first

generate_playlist_items(filter = c(playlist_id = "PLrEnWoR732-CN09YykVof21xdI3MLOZda"))
generate_playlist_items(filter = c(playlist_id = "PL0f01XVeVw9QM03GoESky4yDgQFk25sXN"),
                        max_results = 51)

## End(Not run)
```

get_related_videos  

Get Related Videos

Description

Takes a video id and returns related videos

Usage

```r
get_related_videos(video_id = NULL, max_results = 50, safe_search = "none", ...)
```
get_stats

Arguments

- **video_id**: Character. Required. No default.
- **max_results**: Maximum number of items that should be returned. Integer. Optional. Can be between 0 and 50. Default is 50.
- **safe_search**: Character. Optional. Takes one of three values: 'moderate', 'none' (default) or 'strict'
- **...**: Additional arguments passed to `tuber_GET`.

Value

*data.frame* with 16 columns: `video_id`, `rel_video_id`, `publishedAt`, `channelId`, `title`, `description`, `thumbnails_default_url`, `thumbnails_default_width`, `thumbnails_high_url`, `thumbnails_high_width`, `thumbnails_high_height`, `channelTitle`, `liveBroadcastContent`

References

[https://developers.google.com/youtube/v3/docs/search/list](https://developers.google.com/youtube/v3/docs/search/list)

Examples

```r
## Not run:

# Set API token via `yt_oauth()` first

get_related_videos(video_id = "yJXTXN4xI8")

## End(Not run)
```

---

**get_stats**

Get statistics of a Video

Description

Get statistics of a Video

Usage

```r
get_stats(video_id = NULL, ...)
```

Arguments

- **video_id**: Character. Id of the video. Required.
- **...**: Additional arguments passed to `tuber_GET`.

Value

*list* with 6 elements: `id`, `viewCount`, `likeCount`, `dislikeCount`, `favoriteCount`, `commentCount`
get_subscriptions

References

https://developers.google.com/youtube/v3/docs/videos/list#parameters

Examples

```r
## Not run:

# Set API token via yt_oauth() first

get_stats(video_id="N708P-A45D0")

## End(Not run)
```

---

get_subscriptions  Get Subscriptions

Description

Get Subscriptions

Usage

```r
get_subscriptions(filter = NULL, part = "contentDetails",
max_results = 50, for_channel_id = NULL, order = NULL,
page_token = NULL, ...)
```

Arguments

- **filter**: string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: ID of the channel. Required. No default. subscription_id: YouTube subscription ID
- **part**: Part of the resource requested. Required. Character. A comma separated list of one or more of the following: contentDetails, id, snippet, subscriberSnippet. e.g. "id, snippet", "id", etc. Default: contentDetails.
- **max_results**: Maximum number of items that should be returned. Integer. Optional. Can be between 0 and 50. Default is 50.
- **for_channel_id**: Optional. String. A comma-separated list of channel IDs. Limits response to subscriptions matching those channels.
- **order**: method that will be used to sort resources in the API response. Takes one of the following: alphabetical, relevance, unread
- **page_token**: Specific page in the result set that should be returned. Optional. String.
- **...**: Additional arguments passed to `tuber_GET`.

Value

named list of subscriptions
get_video_details

References

https://developers.google.com/youtube/v3/docs/subscriptions/list

Examples

## Not run:

```r
# Set API token via yt_oauth() first
get_subscriptions(filter = c(channel_id = "UChTJY5kY3hYazJZ0-euHg"))
```

## End(Not run)

---

**get_video_details**  
*Get Details of a Video or Videos*

Description

Get details such as when the video was published, the title, description, thumbnails, category etc.

Usage

```r
get_video_details(video_id = NULL, part = "snippet", ...)
```

Arguments

- `video_id`: Comma separated list of IDs of the videos for which details are requested. Required.
- `part`: Comma-separated list of video resource properties requested. Options include: contentDetails, fileDetails, id, liveStreamingDetails, localizations, player, processingDetails, recordingDetails, snippet, statistics, status, suggestions, topicDetails.
- `...`: Additional arguments passed to `tuber_GET`.

Value

list. If part is snippet, the list will have the following elements: id (video id that was passed), publishedAt, channelId, title, description, thumbnails, channelTitle, categoryId, liveBroadcastContent...

References

https://developers.google.com/youtube/v3/docs/videos/list
list_abuse_report_reasons

List reasons that can be used to report abusive videos

Description

List reasons that can be used to report abusive videos

Usage

list_abuse_report_reasons(part = "id, snippet", hl = "en-US", ...)

Arguments

part Caption resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., "id, snippet", "id", etc. Default: snippet.

hl Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.

... Additional arguments passed to tuber_GET.

Value

If no results, empty data.frame returned If part requested = "id, snippet" or "snippet", data.frame with 4 columns: etag, id, label, secReasons If part requested = "id", data.frame with 2 columns: etag, id

References

Examples

## Not run:

```
# Set API token via yt_oauth() first

list_abuse_report_reasons()
list_abuse_report_reasons(part="id")
list_abuse_report_reasons(part="snippet")

## End(Not run)
```

---

**list_captions**  
*Upload Video to Youtube*

Description

Upload Video to Youtube

Usage

```
list_captions(video_id, query = NULL, ...)
```

Arguments

- **video_id**: ID of the YouTube video
- **query**: Fields for ‘query’ in ‘GET’
- **...**: Additional arguments to send to `tuber.GET` and therefore `GET`

Value

A list of the response object from the `GET` and content about captions

Examples

```
## Not run:

video_id <- "M7F1vfx5J10"
list_captions(video_id)

## End(Not run)
```
list_caption_tracks  List Captions of a Video

Description
List Captions of a Video

Usage
list_caption_tracks(part = "snippet", video_id = NULL, lang = "en", id = NULL, simplify = TRUE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>part</td>
<td>Caption resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., &quot;id, snippet&quot;, &quot;id&quot; Default: snippet.</td>
</tr>
<tr>
<td>video_id</td>
<td>ID of the video whose captions are requested. Required. No default.</td>
</tr>
<tr>
<td>lang</td>
<td>Language of the caption; required; default is English (&quot;en&quot;)</td>
</tr>
<tr>
<td>id</td>
<td>comma-separated list of IDs that identify the caption resources that should be retrieved; optional; string</td>
</tr>
<tr>
<td>simplify</td>
<td>Boolean. Default is TRUE. When TRUE, and part is snippet, a data.frame is returned</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to tuber_GET.</td>
</tr>
</tbody>
</table>

Value
list of caption tracks. When simplify is TRUE, a data.frame is returned with following columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>videoId</td>
<td>ID of the video whose captions are requested</td>
</tr>
<tr>
<td>lastUpdated</td>
<td>Last updated date of the caption</td>
</tr>
<tr>
<td>trackKind</td>
<td>Kind of the caption tracks</td>
</tr>
<tr>
<td>language</td>
<td>Language of the caption</td>
</tr>
<tr>
<td>name</td>
<td>Name of the caption</td>
</tr>
<tr>
<td>audioTrackType</td>
<td>Type of the audio track</td>
</tr>
<tr>
<td>isCC</td>
<td>Is caption closed captions?</td>
</tr>
<tr>
<td>isLarge</td>
<td>Is caption large?</td>
</tr>
<tr>
<td>isEasyReader</td>
<td>Is caption easy reader?</td>
</tr>
<tr>
<td>isDraft</td>
<td>Is caption draft?</td>
</tr>
<tr>
<td>isAutosynced</td>
<td>Is caption autosynced?</td>
</tr>
<tr>
<td>status</td>
<td>Status of the caption</td>
</tr>
<tr>
<td>id (caption id)</td>
<td>ID of the caption</td>
</tr>
</tbody>
</table>

References
https://developers.google.com/youtube/v3/docs/captions/list

Examples
```r
## Not run:

# Set API token via yt_oauth() first

list_caption_tracks(video_id = "yJXTXN4xrI8")

## End(Not run)
```
list_channel_activities

List Channel Activity

Description

Returns a list of channel events that match the request criteria.

Usage

list_channel_activities(filter = NULL, part = "snippet", max_results = 50, page_token = NULL, published_after = NULL, published_before = NULL, region_code = NULL, simplify = TRUE, ...)

Arguments

filter string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: ID of the channel. Required. No default.

part specify which part do you want. It can only be one of the three: contentDetails, id, snippet. Default is snippet.

max_results Maximum number of items that should be returned. Integer. Optional. Can be between 0 and 50. Default is 50.

page_token specific page in the result set that should be returned, optional

published_after Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"

published_before Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"

region_code ISO 3166-1 alpha-2 country code, optional, see also list_regions

simplify Data Type: Boolean. Default is TRUE. If TRUE and if part requested is contentDetails, the function returns a data.frame. Else a list with all the information returned.

... Additional arguments passed to tuber_GET.

Value

named list If simplify is TRUE, a data.frame is returned with 18 columns: publishedAt, channelId, title, description, ...

References

https://developers.google.com/youtube/v3/docs/activities/list
Examples

## Not run:

```r
# Set API token via yt_oauth() first

list_channel_activities(filter = c(channel_id = "UCRW8bIz2wMLmfgAgWm903ca"))
list_channel_activities(filter = c(channel_id = "UCRW8bIz2wMLmfgAgWm903ca", regionCode="US"))
list_channel_activities(filter = c(channel_id = "UCMtFAi84ehTSYSE9xHefig"),
  published_before = "2016-02-10T00:00:00Z",
  published_after = "2016-01-01T00:00:00Z")
```

## End(Not run)

---

list_channel_resources

Returns List of Requested Channel Resources

Description

Returns List of Requested Channel Resources

Usage

```r
list_channel_resources(filter = NULL, part = "contentDetails",
  max_results = 50, page_token = NULL, hl = "en-US", ...)
```

Arguments

- `filter` string: Required. named vector of length 1 potential names of the entry in the vector: `category_id`: YouTube guide category that returns channels associated with that category.

- `part` string: Required. One of the following: `auditDetails`, `brandingSettings`, `contentDetails`, `contentOwnerDetails`, `id`, `inVideoPromotion`, `localizations`, `snippet`, `statistics`, `status`, `topicDetails`. Default is `contentDetails`.

- `max_results` Maximum number of items that should be returned. Integer. Optional. Can be between 0 and 50. Default is 50.

- `page_token` specific page in the result set that should be returned, optional

- `hl` Language used for text values. Optional. Default is `en-US`. For other allowed language codes, see `list_langs`.

Value

```
list
```
**list_channel_sections**

**References**

https://developers.google.com/youtube/v3/docs/channels/list

**Examples**

```r
## Not run:

# Set API token via yt_oauth() first

list_channel_resources(filter = c(channel_id = "UCT5Cxl4LS3wXkjJXNyuj4TA"))
list_channel_resources(filter = c(username = "latenight"), part = "id, contentDetails")
list_channel_resources(filter = c(username = "latenight"), part = "id, contentDetails", max_results = 10)

## End(Not run)
```

---

**list_channel_sections  List Channel Sections**

**Description**

Returns list of channel sections that channel id belongs to.

**Usage**

```r
list_channel_sections(filter = NULL, part = "snippet", hl = NULL, ...)```

**Arguments**

- **filter** string: Required. named vector of length 1 potential names of the entry in the vector: channel_id: Channel ID id: Section ID
- **part** specify which part do you want. It can only be one of the following: contentDetails, id, localization. Default is snippet.
- **hl** language that will be used for text values, optional, default is en-US. See also list_langs
- **...** Additional arguments passed to tuber_GET.

**Value**

captions for the video from one of the first track

**References**

https://developers.google.com/youtube/v3/docs/activities/list
list_channel_videos

Examples

## Not run:

# Set API token via yt_oauth() first

list_channel_sections(c(channel_id = "UCRw8bIzZwMLmfgAgWm903cA"))

## End(Not run)

---

list_channel_videos  Returns List of Requested Channel Videos

Description

Iterate through the max_results number of playlists in channel and get the videos for each of the playlists.

Usage

list_channel_videos(channel_id = NULL, max_results = 50,
page_token = NULL, hl = "en-US", ...)

Arguments

- channel_id  String. ID of the channel. Required.
- max_results  Maximum number of videos returned. Integer. Default is 50. If the number is over 50, all the videos will be returned.
- page_token  Specific page in the result set that should be returned. Optional.
- hl  Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs
- ...  Additional arguments passed to tuber_GET.

Value

list of data.frame with each list corresponding to a different playlist

References

https://developers.google.com/youtube/v3/docs/channels/list
list_guidecats

Examples

```r
## Not run:
# Set API token via yt_oauth() first
list_channel_videos(channel_id = "UCXQKEdF0FxH6-u3K8SHg")
list_channel_videos(channel_id = "UCXQKEdF0FxH6-u3K8SHg", max_results = 10)
## End(Not run)
```

---

**list_guidecats**

*Get list of categories that can be associated with YouTube channels*

**Description**

Get list of categories that can be associated with YouTube channels

**Usage**

```r
list_guidecats(filter = NULL, hl = NULL, ...)
```

**Arguments**

- `filter` [string]: Required. named vector of length 1 potential names of the entry in the vector: `region_code`: Character. Required. Has to be a ISO 3166-1 alpha-2 code (see [https://www.iso.org/obp/ui/#search](https://www.iso.org/obp/ui/#search))
- `hl` [language code]: Language used for text values. Optional. Default is en-US. For other allowed language codes, see `list_langs`.
- `...` Additional arguments passed to `tuber_get`.

**Value**

*data.frame* with 5 columns: `region_code`, `channelId`, `title`, `etag`, `id`

**References**

[https://developers.google.com/youtube/v3/docs/guideCategories/list](https://developers.google.com/youtube/v3/docs/guideCategories/list)

**Examples**

```r
## Not run:
# Set API token via yt_oauth() first
list_guidecats(c(region_code = "JP"))
## End(Not run)
```
**list_langs**  
*List Languages That YouTube Currently Supports*

**Description**
List Languages That YouTube Currently Supports

**Usage**
list_langs(hl = NULL, ...)

**Arguments**
- **hl**: Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
- **...**: Additional arguments passed to tuber_GET.

**Value**
data.frame with 3 columns: hl (two letter abbreviation), name (of the language), etag

**References**
https://developers.google.com/youtube/v3/docs/i18nLanguages/list

**Examples**
```r
## Not run:

# Set API token via yt_oauth() first
list_langs()

## End(Not run)
```

**list_my_videos**  
*List My videos*

**Description**
List My videos

**Usage**
list_my_videos(...)
**list_regions**

**Arguments**

...  additional arguments to pass to `list_channel_videos`

**Value**

data.frame with each list corresponding to a different playlist

**Examples**

```r
## Not run:
list_my_videos()

## End(Not run)
```

---

**list_regions**

**List Content Regions That YouTube Currently Supports**

**Description**

List Content Regions That YouTube Currently Supports

**Usage**

`list_regions(hl = NULL, ...)`

**Arguments**

- `hl` Language used for text values. Optional. Default is en-US. For other allowed language codes, see `list_langs`.
- `...` Additional arguments passed to `tuber_GET`.

**Value**

data.frame with 3 columns: `gl` (two letter abbreviation), `name` (of the region), `etag`

**References**

[https://developers.google.com/youtube/v3/docs/i18nRegions/list](https://developers.google.com/youtube/v3/docs/i18nRegions/list)

**Examples**

```r
## Not run:

# Set API token via `yt_oauth()` first
list_regions()

## End(Not run)
```
list_videocats  

List of Categories That Can be Associated with Videos

Description

List of Categories That Can be Associated with Videos

Usage

list_videocats(filter = NULL, ...)

Arguments

- **filter**: string; Required. named vector of length 1 potential names of the entry in the vector: region_code: Character. Required. Has to be a ISO 3166-1 alpha-2 code (see https://www.iso.org/obp/ui/#search) category_id: video category ID
- ... Additional arguments passed to tuber_GET.

Value

data.frame with 6 columns: region_code, channelId, title, assignable, etag, id

References

https://developers.google.com/youtube/v3/docs/videoCategories/list

Examples

```r
## Not run:

# Set API token via yt_oauth() first

list_videocats(c(region_code = "JP"))
list_videocats() # Will throw an error asking for a valid filter with valid region_code

## End(Not run)
```
list_videos  List (Most Popular) Videos

Description
List (Most Popular) Videos

Usage

list_videos(part = "contentDetails", max_results = 50, page_token = NULL,
             hl = NULL, region_code = NULL, video_category_id = NULL, ...)

Arguments

- **part**: Required. Comma separated string including one or more of the following:
  contentDetails, fileDetails, id, liveStreamingDetails, localizations, player, processingDetails,
  recordingDetails, snippet, statistics, status, suggestions, topicDetails.
  Default: contentDetails.

- **max_results**: Maximum number of items that should be returned. Integer. Optional. Can be
  between 0 and 50. Default is 50.

- **page_token**: specific page in the result set that should be returned, optional

- **hl**: Language used for text values. Optional. Default is en-US. For other allowed
  language codes, see list_langs.

- **region_code**: Character. Required. Has to be a ISO 3166-1 alpha-2 code (see

- **video_category_id**: the video category for which the chart should be retrieved. See also list_videocats.

- **...**: Additional arguments passed to tuber_GET.

Value
data.frame with 5 columns: channelId, title, assignable, etag, id

References
https://developers.google.com/youtube/v3/docs/search/list

Examples

```R
## Not run:

# Set API token via yt_oauth() first
list_videos()

## End(Not run)
```
### tuber

**tuber** provides access to the YouTube API V3.

### tuber_check

**Request Response Verification**

#### Description

Request Response Verification

#### Usage

```
tuber_check(req)
```

#### Arguments

- **req**: request

#### Value

- in case of failure, a message

### tuber_DELETE

**DELETE**

#### Description

DELETE

#### Usage

```
tuber_DELETE(path, query, ...)
```

#### Arguments

- **path**: path to specific API request URL
- **query**: query list
- **...**: Additional arguments passed to GET.

#### Value

- list
**tuber_GET**

### Description

**GET**

### Usage

```
tuber_GET(path, query, ...)
```

### Arguments

- **path**
  - path to specific API request URL
- **query**
  - query list
- **...**
  - Additional arguments passed to **GET**.

### Value

```
list
```

---

**tuber_POST**

### Description

**POST**

### Usage

```
tuber_POST(path, query, body = "", ...)
```

### Arguments

- **path**
  - path to specific API request URL
- **query**
  - query list
- **body**
  - passing image through body
- **...**
  - Additional arguments passed to **GET**.

### Value

```
list
```
upload_caption

Upload Video Caption to Youtube

Description

Upload Video Caption to Youtube

Usage

upload_caption(file, video_id, language = "en-US", caption_name, is_draft = FALSE, query = NULL, open_url = FALSE, ...)

Arguments

file Filename of the caption, probably `.srt`
video_id YouTube Video ID. Try list_my_videos for examples.
language character string of ‘BCP47’ language type. See http://www.rfc-editor.org/rfc/bcp/bcp47.txt for language specification
caption_name character vector of the name for the caption.
is_draft logical indicating whether the caption track is a draft.
query Fields for ‘query’ in ‘POST’
open_url Should the video be opened using browseURL
... Additional arguments to send to POST

Value

A list of the response object from the POST, content, and the URL of the video

Note

See https://developers.google.com/youtube/v3/docs/captions#resource for full specification

Examples

## Not run:
xx = list_my_videos()
video_id = xx$contentDetails.videoId[1]
video_id = as.character(video_id)
language = "en-US"

## End(Not run)
Description

Upload Video to Youtube

Usage

upload_video(file, snippet = NULL, status = list(privacyStatus = "public"), query = NULL, open_url = FALSE, ...)

Arguments

file Filename of the video locally
snippet Additional fields for the video, including 'description' and 'title'. See [https://developers.google.com/youtube/v3/docs/videos#resource](https://developers.google.com/youtube/v3/docs/videos#resource) for other fields. Coerced to a JSON object
status Additional fields to be put into the status input. options for 'status' are 'license' (which should hold: 'creativeCommon', or 'youtube'), 'privacyStatus', 'publicStatsViewable', 'publishAt'.
query Fields for 'query' in 'POST'
open_url Should the video be opened using browseURL
... Additional arguments to send to tuber_POST and therefore POST

Value

A list of the response object from the POST, content, and the URL of the uploaded

Note

The information for 'status' and 'snippet' are at [https://developers.google.com/youtube/v3/docs/videos#resource](https://developers.google.com/youtube/v3/docs/videos#resource) but the subset of these fields to pass in are located at: [https://developers.google.com/youtube/v3/docs/videos/insert](https://developers.google.com/youtube/v3/docs/videos/insert) The 'part" parameter serves two purposes in this operation. It identifies the properties that the write operation will set, this will be automatically detected by the names of 'body'. See [https://developers.google.com/youtube/v3/docs/videos/insert#usage](https://developers.google.com/youtube/v3/docs/videos/insert#usage)

Examples

```r
snippet = list(
    title = "Test Video",
    description = "This is just a random test.",
    tags = c("r language", "r programming", "data analysis")
)
status = list(privacyStatus = "private")
```
yt_check_token  
*Check if authentication token is in options*

**Description**

Check if authentication token is in options

**Usage**

```
yt_check_token()
```

---

yt_oauth  
*Set up Authorization*

**Description**

The function looks for `.httr_oauth` in the working directory. If it doesn’t find it, it expects an application ID and a secret. If you want to remove the existing `.httr_oauth`, set `remove_old_oauth` to TRUE. By default, it is set to FALSE. The function launches a browser to allow you to authorize the application

**Usage**

```
yt_oauth(app_id = NULL, app_secret = NULL, scope = "ssl",
         token = ".httr_oauth", ...)
```

**Arguments**

- `app_id`  
  client id; required; no default
- `app_secret`  
  client secret; required; no default
- `scope`  
  Character. `ssl`, `basic`, `own_account_readonly`, `upload_and_manage_own_videos`, `partner`, and `partner_audit`. Required. `ssl` and `basic` are basically interchangeable. Default is `ssl`.
- `token`  
  path to file containing the token. If a path is given, the function will first try to read from it. Default is `.httr_oauth` in the local directory. So if there is such a file, the function will first try to read from it.
- `...`  
  Additional arguments passed to `oauth2.0_token`

**Value**

sets the `google_token` option and also saves `.httr_oauth` in the working directory (find out the working directory via `getwd()`)
yt_search

References

https://developers.google.com/youtube/v3/docs/
https://developers.google.com/youtube/v3/docs/auth/client-side-web-apps for different scopes

Examples

## Not run:
yt_oauth("998136489867-5t3tq1g7hbvoj46dreqd6k5kd35ctjn.apps.googleusercontent.com",
"Mb0t6cQhhFkwETXkur-L9rN")

## End(Not run)

---

**yt_search**

**Search YouTube**

**Description**

Search for videos, channels and playlists. (By default, the function searches for videos.)

**Usage**

```r
ty_search(term = NULL, max_results = 50, channel_id = NULL,
channel_type = NULL, type = "video", event_type = NULL,
location = NULL, location_radius = NULL, published_after = NULL,
published_before = NULL, video_definition = "any",
video_caption = "any", video_license = "any", video_syndicated = "any",
video_type = "any", simplify = TRUE, get_all = TRUE,
page_token = NULL, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>term</td>
<td>Character. Search term; required; no default</td>
</tr>
<tr>
<td>max_results</td>
<td>Maximum number of items that should be returned. Integer. Optional. Can be</td>
</tr>
<tr>
<td></td>
<td>between 0 and 50. Default is 50. Search results are constrained to a maximum</td>
</tr>
<tr>
<td></td>
<td>of 500 videos if type is video and we have a value of channel_id.</td>
</tr>
<tr>
<td>channel_id</td>
<td>Character. Only return search results from this channel; Optional.</td>
</tr>
<tr>
<td>channel_type</td>
<td>Character. Optional. Takes one of two values: 'any', 'show'. Default is</td>
</tr>
<tr>
<td></td>
<td>'any'</td>
</tr>
<tr>
<td>type</td>
<td>Character. Optional. Takes one of three values: 'video', 'channel', 'playlist'.</td>
</tr>
<tr>
<td></td>
<td>Default is 'video'.</td>
</tr>
<tr>
<td>event_type</td>
<td>Character. Optional. Takes one of three values: 'completed', 'live', 'upcoming'</td>
</tr>
<tr>
<td>location</td>
<td>Character. Optional. Latitude and Longitude within parentheses, e.g. &quot;(37.42307,-122.08427)&quot;</td>
</tr>
</tbody>
</table>
location_radius
Character. Optional. e.g. "1500m", "5km", "10000ft", "0.75mi"

published_after
Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"

published_before
Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"

video_definition
Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'high', 'standard'

video_caption
Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'closedCaption', 'none'. Type must be set to video.

video_license
Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'creativeCommon' (return videos with Creative Commons license), 'youtube' (return videos with standard YouTube license).

video_syndicated
Character. Optional. Takes one of two values: 'any' (return all videos; Default), 'true' (return only syndicated videos)

video_type
Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'episode' (return episode of shows), 'movie' (return movies)

simplify
Boolean. Return a data.frame if TRUE. Default is TRUE. If TRUE, it returns a list that carries additional information.

get_all
get all results, iterating through all the results pages. Default is TRUE. Result is a data.frame. Optional.

page_token
specific page in the result set that should be returned, optional

... Additional arguments passed to tuber_GET.

Value
data.frame with 16 elements: video_id, publishedAt, channelId, title, description, thumbnails.default.url, ...

References
https://developers.google.com/youtube/v3/docs/search/list

Examples

## Not run:

# Set API token via yt_oauth() first

yt_search(term = "Barack Obama")
yt_search(term = "Barack Obama", published_after = "2016-10-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-09-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-03-01T00:00:00Z", published_after = "2016-02-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-02-10T00:00:00Z", published_after = "2016-02-01T00:00:00Z")
Search YouTube by Topic

### Description

Search YouTube by Topic It uses the Freebase list of topics

### Usage

yt_topic_search(topic = NULL, ...)

### Arguments

- topic: topic being searched for; required; no default
- ...: Additional arguments passed to tuber_GET.

### Value

a list

### Examples

```r
## Not run:

# Set API token via yt_oauth() first
yt_topic_search(topic = "Barack Obama")

## End(Not run)
```
browseURL, 32, 33

delete_captions, 3
delete_channel_sections, 3
delete_comments, 4
delete_playlist_items, 5
delete_playlists, 5
delete_videos, 6

GET, 19, 30, 31
get_all_channel_video_stats, 7
get_all_comments, 8
get_captions, 8
get_channel_stats, 9
get_comment_threads, 11
get_comments, 10
get_playlist_items, 13
get_playlists, 12
get_related_videos, 14
get_stats, 15
get_subscriptions, 16
get_video_details, 17

list_abuse_report_reasons, 18
list_caption_tracks, 8, 20
list_captions, 19
list_channel_activities, 21
list_channel_resources, 22
list_channel_sections, 23
list_channel_videos, 24, 27
list_guidecats, 25
list_langs, 13, 18, 22–26, 26, 27, 29
list_my_channel (get_channel_stats), 9
list_my_videos, 26, 32
list_regions, 21, 27
list_videocats, 28, 29
list_videos, 29

oauth2.0_token, 34

POST, 32, 33

tuber, 30
tuber-package (tuber), 30
tuber_check, 30
tuber_DELETE, 3–6, 30
tuber_GET, 7–10, 12–29, 31, 36, 37
tuber_POST, 31, 33

upload_caption, 32
upload_video, 33

yt_check_token, 34
yt_oauth, 34
yt_search, 35
yt_topic_search, 37