ytmusicapi

Release 1.6.1.dev5+g1660d37

sigma67

Apr 15, 2024

CONTENTS

1 Features **3**

2 Requirements **5**

3 Setup **7**

4 Usage **9**

5 Contents **11**
- 5.1 Setup 11
- 5.2 Usage 13
- 5.3 Reference 13
- 5.4 FAQ 50

Index **57**

The purpose of this library is to automate interactions with YouTube Music, such as retrieving your library content, managing playlists and uploading songs. To achieve this, it emulates web requests that would occur if you performed the same actions in your web browser.

This project is not supported nor endorsed by Google

CHAPTER

ONE

FEATURES

Browsing:

- search (including all filters) and suggestions
- get artist information and releases (songs, videos, albums, singles, related artists)
- get user information (videos, playlists)
- get albums
- get song metadata
- get watch playlists (next songs when you press play/radio/shuffle in YouTube Music)
- get song lyrics

Exploring music:

- get moods and genres playlists
- get latest charts (globally and per country)

Library management:

- get library contents: playlists, songs, artists, albums and subscriptions, podcasts, channels
- add/remove library content: rate songs, albums and playlists, subscribe/unsubscribe artists
- get and modify play history

Playlists:

- create and delete playlists
- modify playlists: edit metadata, add/move/remove tracks
- get playlist contents
- get playlist suggestions

Podcasts:

- get podcasts
- get episodes
- get channels
- get episodes playlists

Uploads:

- upload songs and remove them again
- list uploaded songs, artists and albums

Localization:

- all regions are supported (see locations FAQ
- 16 languages are supported (see languages FAQ

If you find something missing or broken, check the FAQ or feel free to create an issue.

CHAPTER

TWO

REQUIREMENTS

- Python 3.8 or higher - https://www.python.org

CHAPTER

THREE

SETUP

See the Documentation for detailed instructions

CHAPTER

FOUR

USAGE

```
from ytmusicapi import YTMusic

yt = YTMusic('oauth.json')
playlistId = yt.create_playlist('test', 'test description')
search_results = yt.search('Oasis Wonderwall')
yt.add_playlist_items(playlistId, [search_results[0]['videoId']])
```

The tests are also a great source of usage examples.

To **get started**, read the *setup instructions*.

For a **complete documentation** of available functions, see the *Reference*.

CHAPTER

FIVE

CONTENTS

5.1 Setup

To install, run:

```
pip install ytmusicapi
```

Further setup is only needed if you want to access account data using authenticated requests.

The simplest way of authentication is to use *OAuth authentication*.

However, these OAuth credentials do not work for uploads. If you need to upload music, instead follow the instructions at *Browser authentication*.

5.1.1 OAuth authentication

After you have installed `ytmusicapi`, simply run

```
ytmusicapi oauth
```

and follow the instructions. This will create a file `oauth.json` in the current directory.

You can pass this file to `YTMusic` as explained in *Usage*.

This OAuth flow uses the Google API flow for TV devices.

5.1.2 Browser authentication

This method of authentication emulates your browser session by reusing its request headers. Follow the instructions to have your browser's YouTube Music session request headers parsed to a `ytmusicapi` configuration file.

Copy authentication headers

To run authenticated requests, set it up by first copying your request headers from an authenticated POST request in your browser. To do so, follow these steps:

- Open a new tab
- Open the developer tools (Ctrl-Shift-I) and select the "Network" tab
- Go to https://music.youtube.com and ensure you are logged in

- Find an authenticated POST request. The simplest way is to filter by `/browse` using the search bar of the developer tools. If you don't see the request, try scrolling down a bit or clicking on the library button in the top bar.
- Verify that the request looks like this: **Status** 200, **Method** POST, **Domain** music.youtube.com, **File** `browse?...`
- Copy the request headers (right click > copy > copy request headers)
- Verify that the request looks like this: **Status** 200, **Name** `browse?...`
- Click on the Name of any matching request. In the "Headers" tab, scroll to the section "Request headers" and copy everything starting from "accept: */*" to the end of the section

Using the headers in your project

To set up your project, open a console and call

```
ytmusicapi browser
```

Follow the instructions and paste the request headers to the terminal input.

If you don't want terminal interaction in your project, you can pass the request headers with the `headers_raw` parameter:

```
import ytmusicapi
ytmusicapi.setup(filepath="browser.json", headers_raw="<headers copied above>")
```

The function returns a JSON string with the credentials needed for *Usage*. Alternatively, if you passed the filepath parameter as described above, a file called `browser.json` will be created in the current directory, which you can pass to `YTMusic()` for authentication.

These credentials remain valid as long as your YTMusic browser session is valid (about 2 years unless you log out).

- MacOS terminal application can only accept 1024 characters pasted to std input. To paste in terminal, a small utility called pbpaste must be used.
- **In terminal just prefix the command used to run the script you created above with**
  `pbpaste |`
- This will pipe the contents of the clipboard into the script just as if you had pasted it from the edit menu.

Manual file creation

Alternatively, you can create your own file `browser.json` and paste the cookie:

```
{
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101␣
↪Firefox/72.0",
    "Accept": "*/*",
    "Accept-Language": "en-US,en;q=0.5",
    "Content-Type": "application/json",
    "X-Goog-AuthUser": "0",
    "x-origin": "https://music.youtube.com",
    "Cookie" : "PASTE_COOKIE"
}
```

5.2 Usage

5.2.1 Unauthenticated

Unauthenticated requests for retrieving playlist content or searching:

```
from ytmusicapi import YTMusic

ytmusic = YTMusic()
```

If an endpoint requires authentication you will receive an error: `Please provide authentication before using this function`

5.2.2 Authenticated

For authenticated requests you need to set up your credentials first: *Setup*

After you have created the authentication JSON, you can instantiate the class:

```
from ytmusicapi import YTMusic
ytmusic = YTMusic("oauth.json")
```

With the `ytmusic` instance you can now perform authenticated requests:

```
playlistId = ytmusic.create_playlist("test", "test description")
search_results = ytmusic.search("Oasis Wonderwall")
ytmusic.add_playlist_items(playlistId, [search_results[0]['videoId']])
```

Brand accounts

To send requests as a brand account, there is no need to change authentication credentials. Simply provide the ID of the brand account when instantiating YTMusic. You can get the ID from https://myaccount.google.com/ after selecting your brand account (https://myaccount.google.com/b/21_digit_number).

Example:

```
from ytmusicapi import YTMusic
ytmusic = YTMusic("oauth.json", "101234161234936123473")
```

5.3 Reference

Reference for the YTMusic class.

class ytmusicapi.**YTMusic**(*auth: str | Dict | None = None*, *user: str | None = None*, *requests_session=True*, *proxies: Dict[str, str] | None = None*, *language: str = 'en'*, *location: str = ''*, *oauth_credentials: OAuthCredentials | None = None*)

Allows automated interactions with YouTube Music by emulating the YouTube web client's requests. Permits both authenticated and non-authenticated requests. Authentication header data must be provided on initialization.

YTMusic.__init__(*auth: str | Dict | None = None*, *user: str | None = None*, *requests_session=True*, *proxies: Dict[str, str] | None = None*, *language: str = 'en'*, *location: str = ''*, *oauth_credentials: OAuthCredentials | None = None*)

Create a new instance to interact with YouTube Music.

Parameters

- **auth** – Optional. Provide a string, path to file, or oauth token dict. Authentication credentials are needed to manage your library. See *setup()* for how to fill in the correct credentials. Default: A default header is used without authentication.
- **user** – Optional. Specify a user ID string to use in requests. This is needed if you want to send requests on behalf of a brand account. Otherwise the default account is used. You can retrieve the user ID by going to https://myaccount.google.com/brandaccounts and selecting your brand account. The user ID will be in the URL: https://myaccount.google.com/b/user_id/
- **requests_session** – A Requests session object or a truthy value to create one. Default sessions have a request timeout of 30s, which produces a requests.exceptions.ReadTimeout. The timeout can be changed by passing your own Session object:

```
s = requests.Session()
s.request = functools.partial(s.request, timeout=3)
ytm = YTMusic(requests_session=s)
```

  A falsy value disables sessions. It is generally a good idea to keep sessions enabled for performance reasons (connection pooling).
- **proxies** – Optional. Proxy configuration in requests format.
- **language** – Optional. Can be used to change the language of returned data. English will be used by default. Available languages can be checked in the ytmusicapi/locales directory.
- **location** – Optional. Can be used to change the location of the user. No location will be set by default. This means it is determined by the server. Available languages can be checked in the FAQ.
- **oauth_credentials** – Optional. Used to specify a different oauth client to be used for authentication flow.

5.3.1 Setup

See also the *Setup* page

ytmusicapi.setup(*filepath: str | None = None*, *headers_raw: str | None = None*) → str

Requests browser headers from the user via command line and returns a string that can be passed to YTMusic()

Parameters

- **filepath** – Optional filepath to store headers to.
- **headers_raw** – Optional request headers copied from browser. Otherwise requested from terminal

Returns

configuration headers string

ytmusicapi.setup_oauth(*filepath: str | None = None*, *session: Session | None = None*, *proxies: dict | None = None*, *open_browser: bool = False*, *client_id: str | None = None*, *client_secret: str | None = None*) → RefreshingToken

Starts oauth flow from the terminal and returns a string that can be passed to YTMusic()

Parameters

- **session** – Session to use for authentication
- **proxies** – Proxies to use for authentication
- **filepath** – Optional filepath to store headers to.
- **open_browser** – If True, open the default browser with the setup link
- **client_id** – Optional. Used to specify the client_id oauth should use for authentication flow. If provided, client_secret MUST also be passed or both will be ignored.
- **client_secret** – Optional. Same as client_id but for the oauth client secret.

Returns

configuration headers string

5.3.2 Search

YTMusic.**search**(*query: str*, *filter: str | None = None*, *scope: str | None = None*, *limit: int = 20*, *ignore_spelling: bool = False*) → List[Dict]

Search YouTube music Returns results within the provided category.

Parameters

- **query** – Query string, i.e. 'Oasis Wonderwall'
- **filter** – Filter for item types. Allowed values: `songs`, `videos`, `albums`, `artists`, `playlists`, `community_playlists`, `featured_playlists`, `uploads`. Default: Default search, including all types of items.
- **scope** – Search scope. Allowed values: `library`, `uploads`. Default: Search the public YouTube Music catalogue. Changing scope from the default will reduce the number of settable filters. Setting a filter that is not permitted will throw an exception. For uploads, no filter can be set. For library, community_playlists and featured_playlists filter cannot be set.
- **limit** – Number of search results to return Default: 20
- **ignore_spelling** – Whether to ignore YTM spelling suggestions. If True, the exact search term will be searched for, and will not be corrected. This does not have any effect when the filter is set to `uploads`. Default: False, will use YTM's default behavior of autocorrecting the search.

Returns

List of results depending on filter. resultType specifies the type of item (important for default search). albums, artists and playlists additionally contain a browseId, corresponding to albumId, channelId and playlistId (browseId=``VL``+playlistId)

Example list for default search with one result per resultType for brevity. Normally there are 3 results per resultType and an additional `thumbnails` key:

```
[
  {
    "category": "Top result",
    "resultType": "video",
    "videoId": "vU05Eksc_iM",
```

(continues on next page)

(continued from previous page)


```
    "title": "Wonderwall",
    "artists": [
      {
        "name": "Oasis",
        "id": "UCmMUZbaYdNH0bEd1PAlAqsA"
      }
    ],
    "views": "1.4M",
    "videoType": "MUSIC_VIDEO_TYPE_OMV",
    "duration": "4:38",
    "duration_seconds": 278
  },
  {
    "category": "Songs",
    "resultType": "song",
    "videoId": "ZrOKjDZOtkA",
    "title": "Wonderwall",
    "artists": [
      {
        "name": "Oasis",
        "id": "UCmMUZbaYdNH0bEd1PAlAqsA"
      }
    ],
    "album": {
      "name": "(What's The Story) Morning Glory? (Remastered)",
      "id": "MPREb_9nqEki4ZDpp"
    },
    "duration": "4:19",
    "duration_seconds": 259
    "isExplicit": false,
    "feedbackTokens": {
      "add": null,
      "remove": null
    }
  },
  {
    "category": "Albums",
    "resultType": "album",
    "browseId": "MPREb_9nqEki4ZDpp",
    "title": "(What's The Story) Morning Glory? (Remastered)",
    "type": "Album",
    "artist": "Oasis",
    "year": "1995",
    "isExplicit": false
  },
  {
    "category": "Community playlists",
    "resultType": "playlist",
    "browseId": "VLPLK1PkWQlWtnNfovRdGWpKffO1Wdi2kvDx",
    "title": "Wonderwall - Oasis",
    "author": "Tate Henderson",
    "itemCount": "174"
```


(continues on next page)

(continued from previous page)

```
  },
  {
    "category": "Videos",
    "resultType": "video",
    "videoId": "bx1Bh8ZvH84",
    "title": "Wonderwall",
    "artists": [
      {
        "name": "Oasis",
        "id": "UCmMUZbaYdNH0bEd1PAlAqsA"
      }
    ],
    "views": "386M",
    "duration": "4:38",
    "duration_seconds": 278
  },
  {
    "category": "Artists",
    "resultType": "artist",
    "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
    "artist": "Oasis",
    "shuffleId": "RDAOkjHYJjL1a3xspEyVkhHAsg",
    "radioId": "RDEMkjHYJjL1a3xspEyVkhHAsg"
  },
  {
    "category": "Profiles",
    "resultType": "profile",
    "title": "Taylor Swift Time",
    "name": "@TaylorSwiftTime",
    "browseId": "UCSCRK7XlVQ6fBdEl00kX6pQ",
    "thumbnails": ...
  }
]
```

YTMusic.**get_search_suggestions**(*query: str*, *detailed_runs=False*) → List[str] | List[Dict]

Get Search Suggestions

Parameters

- **query** – Query string, i.e. 'faded'
- **detailed_runs** – Whether to return detailed runs of each suggestion. If True, it returns the query that the user typed and the remaining suggestion along with the complete text (like many search services usually bold the text typed by the user). Default: False, returns the list of search suggestions in plain text.

Returns

List of search suggestion results depending on `detailed_runs` param.

Example response when `query` is 'fade' and `detailed_runs` is set to `False`:

```
[
  "faded",
  "faded alan walker lyrics",
```

(continues on next page)

(continued from previous page)

```
  "faded alan walker",
  "faded remix",
  "faded song",
  "faded lyrics",
  "faded instrumental"
]
```

Example response when `detailed_runs` is set to `True`:

```
[
  {
    "text": "faded",
    "runs": [
      {
        "text": "fade",
        "bold": true
      },
      {
        "text": "d"
      }
    ]
  },
  {
    "text": "faded alan walker lyrics",
    "runs": [
      {
        "text": "fade",
        "bold": true
      },
      {
        "text": "d alan walker lyrics"
      }
    ]
  },
  {
    "text": "faded alan walker",
    "runs": [
      {
        "text": "fade",
        "bold": true
      },
      {
        "text": "d alan walker"
      }
    ]
  },
  ...
]
```

5.3.3 Browsing

YTMusic.**get_home**(*limit=3*) → List[Dict]

Get the home page. The home page is structured as titled rows, returning 3 rows of music suggestions at a time. Content varies and may contain artist, album, song or playlist suggestions, sometimes mixed within the same row

Parameters

limit – Number of rows to return

Returns

List of dictionaries keyed with 'title' text and 'contents' list

Example list:

```
[
    {
        "title": "Your morning music",
        "contents": [
            { //album result
                "title": "Sentiment",
                "year": "Said The Sky",
                "browseId": "MPREb_QtqXtd2xZMR",
                "thumbnails": [...]
            },
            { //playlist result
                "title": "r/EDM top submissions 01/28/2022",
                "playlistId": "PLz7-xrYmULdSLRZGk-6GKUtaBZcgQNwel",
                "thumbnails": [...],
                "description": "redditEDM • 161 songs",
                "count": "161",
                "author": [
                    {
                        "name": "redditEDM",
                        "id": "UCaTrZ9tPiIGHrkCe5bxOGwA"
                    }
                ]
            }
        ]
    },
    {
        "title": "Your favorites",
        "contents": [
            { //artist result
                "title": "Chill Satellite",
                "browseId": "UCrPLFBWdOroD57bkqPbZJog",
                "subscribers": "374",
                "thumbnails": [...]
            }
            { //album result
                "title": "Dragon",
                "year": "Two Steps From Hell",
                "browseId": "MPREb_M9aDqLRbSeg",
                "thumbnails": [...]
```

(continues on next page)

(continued from previous page)

```
            }
        ]
    },
    {
        "title": "Quick picks",
        "contents": [
            { //song quick pick
                "title": "Gravity",
                "videoId": "EludZd6lfts",
                "artists": [{
                        "name": "yetep",
                        "id": "UCSW0r7dClqCoCvQeqXiZBlg"
                    }],
                "thumbnails": [...],
                "album": {
                    "name": "Gravity",
                    "id": "MPREb_D6bICFcuuRY"
                }
            },
            { //video quick pick
                "title": "Gryffin & Illenium (feat. Daya) - Feel Good (L3V3LS Remix)
↪",
                "videoId": "bR5l0hJDnX8",
                "artists": [
                    {
                        "name": "L3V3LS",
                        "id": "UCCVNihbOdkOWw_-ajIYhAbQ"
                    }
                ],
                "thumbnails": [...],
                "views": "10M"
            }
        ]
    }
]
```

YTMusic.**get_artist**(*channelId: str*) → Dict

Get information about an artist and their top releases (songs, albums, singles, videos, and related artists). The top lists contain pointers for getting the full list of releases.

Possible content types for get_artist are:

- songs
- albums
- singles
- shows
- videos
- episodes
- podcasts
- related

Each of these content keys in the response contains `results` and possibly `browseId` and `params`.

- For songs/videos, pass the browseId to *get_playlist()*.
- For albums/singles/shows, pass browseId and params to *get_artist_albums()*.

Parameters
channelId – channel id of the artist

Returns
Dictionary with requested information.

Warning: The returned channelId is not the same as the one passed to the function. It should be used only with *subscribe_artists()*.

Example:


```
{
    "description": "Oasis were ...",
    "views": "1838795605",
    "name": "Oasis",
    "channelId": "UCUDVBtnOQi4c7E8jebpjc9Q",
    "subscribers": "2.3M",
    "subscribed": false,
    "thumbnails": [...],
    "songs": {
        "browseId": "VLPLMpM3Z0118S42R1npOhcjoakLIv1aqnS1",
        "results": [
            {
                "videoId": "ZrOKjDZOtkA",
                "title": "Wonderwall (Remastered)",
                "thumbnails": [...],
                "artist": "Oasis",
                "album": "(What's The Story) Morning Glory? (Remastered)"
            }
        ]
    },
    "albums": {
        "results": [
            {
                "title": "Familiar To Millions",
                "thumbnails": [...],
                "year": "2018",
                "browseId": "MPREb_AYetWMZunqA"
            }
        ],
        "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
        "params": "6gPTAUNwc0JDbndLYlFBQV..."
    },
    "singles": {
        "results": [
            {
                "title": "Stand By Me (Mustique Demo)",
```


(continues on next page)

(continued from previous page)

```
                "thumbnails": [...],
                "year": "2016",
                "browseId": "MPREb_7MPKLhibN5G"
            }
        ],
        "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
        "params": "6gPTAUNwc0JDbndLYlFBQV..."
    },
    "videos": {
        "results": [
            {
                "title": "Wonderwall",
                "thumbnails": [...],
                "views": "358M",
                "videoId": "bx1Bh8ZvH84",
                "playlistId": "PLMpM3Z0118S5xuNckw1HUcj1D021AnMEB"
            }
        ],
        "browseId": "VLPLMpM3Z0118S5xuNckw1HUcj1D021AnMEB"
    },
    "related": {
        "results": [
            {
                "browseId": "UCt2KxZpY5D__kapeQ8cauQw",
                "subscribers": "450K",
                "title": "The Verve"
            },
            {
                "browseId": "UCwK2Grm574W1u-sBzLikldQ",
                "subscribers": "341K",
                "title": "Liam Gallagher"
            },
            ...
        ]
    }
}
```

YTMusic.**get_artist_albums**(*channelId: str*, *params: str*, *limit: int | None = 100*, *order: str | None = None*) → List[Dict]

Get the full list of an artist's albums, singles or shows

Parameters

- **channelId** – browseId of the artist as returned by *get_artist()*
- **params** – params obtained by *get_artist()*
- **limit** – Number of albums to return. *None* retrieves them all. Default: 100
- **order** – Order of albums to return. Allowed values: 'Recency', 'Popularity', 'Alphabetical order'. Default: Default order.

Returns

List of albums in the format of *get_library_albums()*, except artists key is missing.

YTMusic.**get_album**(*browseId: str*) → Dict

Get information and tracks of an album

Parameters

browseId – browseId of the album, for example returned by *search()*

Returns

Dictionary with album and track metadata.

The result is in the following format:


```
{
  "title": "Revival",
  "type": "Album",
  "thumbnails": [],
  "description": "Revival is the...",
  "artists": [
    {
      "name": "Eminem",
      "id": "UCedvOgsKFzcK3hA5taf3KoQ"
    }
  ],
  "year": "2017",
  "trackCount": 19,
  "duration": "1 hour, 17 minutes",
  "audioPlaylistId": "OLAK5uy_nMr9h2VlS-2PULNz3M3XVXQj_P3C2bqaY",
  "tracks": [
    {
      "videoId": "iKLU7z_xdYQ",
      "title": "Walk On Water (feat. Beyoncé)",
      "artists": [
        {
          "name": "Eminem",
          "id": "UCedvOgsKFzcK3hA5taf3KoQ"
        }
      ],
      "album": "Revival",
      "likeStatus": "INDIFFERENT",
      "thumbnails": null,
      "isAvailable": true,
      "isExplicit": true,
      "duration": "5:03",
      "duration_seconds": 303,
      "trackNumber": 0,
      "feedbackTokens": {
        "add": "AB9zfpK...",
        "remove": "AB9zfpK..."
      }
    }
  ],
  "other_versions": [
    {
      "title": "Revival",
      "year": "Eminem",
      "browseId": "MPREb_fefKFOTEZSp",
```


(continues on next page)

(continued from previous page)

```
      "thumbnails": [...],
      "isExplicit": false
    },
  ],
  "duration_seconds": 4657
}
```

YTMusic.**get_album_browse_id**(*audioPlaylistId: str*) → str | None

Get an album's browseId based on its audioPlaylistId

Parameters
audioPlaylistId – id of the audio playlist (starting with *OLAK5uy_*)

Returns
browseId (starting with *MPREb_*)

YTMusic.**get_user**(*channelId: str*) → Dict

Retrieve a user's page. A user may own videos or playlists.

Parameters
channelId – channelId of the user

Returns
Dictionary with information about a user.

Example:

```
{
  "name": "4Tune - No Copyright Music",
  "videos": {
    "browseId": "UC44hbeRoCZVVMVg5z0FfIww",
    "results": [
      {
        "title": "Epic Music Soundtracks 2019",
        "videoId": "bJonJjgS2mM",
        "playlistId": "RDAMVMbJonJjgS2mM",
        "thumbnails": [
          {
            "url": "https://i.ytimg.com/vi/bJon...",
            "width": 800,
            "height": 450
          }
        ],
        "views": "19K"
      }
    ]
  },
  "playlists": {
    "browseId": "UC44hbeRoCZVVMVg5z0FfIww",
    "results": [
      {
        "title": " Machinimasound | Playlist",
        "playlistId": "PLRm766YvPiO9ZqkBuEzSTt6Bk4eWIr3gB",
        "thumbnails": [
          {
```

(continues on next page)

(continued from previous page)


```
            "url": "https://i.ytimg.com/vi/...",
            "width": 400,
            "height": 225
          }
        ]
      }
    ],
    "params": "6gO3AUNvWU..."
  }
}
```


YTMusic.**get_user_playlists**(*channelId: str*, *params: str*) → List[Dict]

Retrieve a list of playlists for a given user. Call this function again with the returned `params` to get the full list.

Parameters

- **channelId** – channelId of the user.
- **params** – params obtained by *get_artist()*

Returns

List of user playlists in the format of *get_library_playlists()*

YTMusic.**get_song**(*videoId: str*, *signatureTimestamp: int | None = None*) → Dict

Returns metadata and streaming information about a song or video.

Parameters

- **videoId** – Video id
- **signatureTimestamp** – Provide the current YouTube signatureTimestamp. If not provided a default value will be used, which might result in invalid streaming URLs

Returns

Dictionary with song metadata.

Example:


```
{
    "playabilityStatus": {
        "status": "OK",
        "playableInEmbed": true,
        "audioOnlyPlayability": {
            "audioOnlyPlayabilityRenderer": {
                "trackingParams": "CAEQx2kiEwiuv9X5i5H1AhWBvlUKHRoZAHk=",
                "audioOnlyAvailability": "FEATURE_AVAILABILITY_ALLOWED"
            }
        },
        "miniplayer": {
            "miniplayerRenderer": {
                "playbackMode": "PLAYBACK_MODE_ALLOW"
            }
        },
        "contextParams": "Q0FBU0FnZ0M="
    },
    "streamingData": {
        "expiresInSeconds": "21540",
```


(continues on next page)

(continued from previous page)


```
        "adaptiveFormats": [
            {
                "itag": 140,
                "url": "https://rr1---sn-h0jelnez.c.youtube.com/videoplayback?
↪expire=1641080272...",
                "mimeType": "audio/mp4; codecs="mp4a.40.2"",
                "bitrate": 131007,
                "initRange": {
                    "start": "0",
                    "end": "667"
                },
                "indexRange": {
                    "start": "668",
                    "end": "999"
                },
                "lastModified": "1620321966927796",
                "contentLength": "3967382",
                "quality": "tiny",
                "projectionType": "RECTANGULAR",
                "averageBitrate": 129547,
                "highReplication": true,
                "audioQuality": "AUDIO_QUALITY_MEDIUM",
                "approxDurationMs": "245000",
                "audioSampleRate": "44100",
                "audioChannels": 2,
                "loudnessDb": -1.3000002
            }
        ]
    },
    "playbackTracking": {
        "videostatsPlaybackUrl": {
          "baseUrl": "https://s.youtube.com/api/stats/playback?cl=491307275&
↪docid=AjXQiKP5kMs&ei=Nl2HY-6MH5WE8gPjnYnoDg&fexp=1714242%2C9405963%2C23804281
↪%2C23858057%2C23880830%2C23880833%2C23882685%2C23918597%2C23934970%2C23946420
↪%2C23966208%2C23983296%2C23998056%2C24001373%2C24002022%2C24002025%2C24004644
↪%2C24007246%2C24034168%2C24036947%2C24077241%2C24080738%2C24120820%2C24135310
↪%2C24135692%2C24140247%2C24161116%2C24162919%2C24164186%2C24169501%2C24175560
↪%2C24181174%2C24187043%2C24187377%2C24187854%2C24191629%2C24197450%2C24199724
↪%2C24200839%2C24209349%2C24211178%2C24217535%2C24219713%2C24224266%2C24241378
↪%2C24248091%2C24248956%2C24255543%2C24255545%2C24262346%2C24263796%2C24265426
↪%2C24267564%2C24268142%2C24279196%2C24280220%2C24283426%2C24283493%2C24287327
↪%2C24288045%2C24290971%2C24292955%2C24293803%2C24299747%2C24390674%2C24391018
↪%2C24391537%2C24391709%2C24392268%2C24392363%2C24392401%2C24401557%2C24402891
↪%2C24403794%2C24406605%2C24407200%2C24407665%2C24407914%2C24408220%2C24411766
↪%2C24413105%2C24413820%2C24414162%2C24415866%2C24416354%2C24420756%2C24421162
↪%2C24425861%2C24428962%2C24590921%2C39322504%2C39322574%2C39322694%2C39322707&
↪ns=yt&plid=AAXusD4TIOMjS5N4&el=detailpage&len=246&of=Jx1iRksbq-rB9N1KSijZLQ&
↪osid=MWU2NzBjYTI%3AAOeUNAagU8UyWDUJIki5raGHy29-60-yTA&uga=29&
↪vm=CAEQABgEOjJBUEV3RWxUNmYzMXNMMC1MYVpCVnRZTmZWMWw1OWVZX2ZOcUtCSkphQ245VFZwOXdTQWJbQVBta0tETEpWNX
↪",
          "headers": [
            {
```


(continues on next page)

(continued from previous page)


```
            "headerType": "USER_AUTH"
          },
          {
            "headerType": "VISITOR_ID"
          },
          {
            "headerType": "PLUS_PAGE_ID"
          }
        ]
      },
      "videostatsDelayplayUrl": {(as above)},
      "videostatsWatchtimeUrl": {(as above)},
      "ptrackingUrl": {(as above)},
      "qoeUrl": {(as above)},
      "atrUrl": {(as above)},
      "videostatsScheduledFlushWalltimeSeconds": [
        10,
        20,
        30
      ],
      "videostatsDefaultFlushIntervalSeconds": 40
  },
  "videoDetails": {
      "videoId": "AjXQiKP5kMs",
      "title": "Sparks",
      "lengthSeconds": "245",
      "channelId": "UCvCk2zFqkCYzpnSgWfx0qOg",
      "isOwnerViewing": false,
      "isCrawlable": false,
      "thumbnail": {
          "thumbnails": []
      },
      "allowRatings": true,
      "viewCount": "12",
      "author": "Thomas Bergersen",
      "isPrivate": true,
      "isUnpluggedCorpus": false,
      "musicVideoType": "MUSIC_VIDEO_TYPE_PRIVATELY_OWNED_TRACK",
      "isLiveContent": false
  },
  "microformat": {
      "microformatDataRenderer": {
          "urlCanonical": "https://music.youtube.com/watch?v=AjXQiKP5kMs",
          "title": "Sparks - YouTube Music",
          "description": "Uploaded to YouTube via YouTube Music Sparks",
          "thumbnail": {
              "thumbnails": [
                  {
                      "url": "https://i.ytimg.com/vi/AjXQiKP5kMs/hqdefault.jpg",
                      "width": 480,
                      "height": 360
                  }
```


(continues on next page)

(continued from previous page)


```
                ]
            },
            "siteName": "YouTube Music",
            "appName": "YouTube Music",
            "androidPackage": "com.google.android.apps.youtube.music",
            "iosAppStoreId": "1017492454",
            "iosAppArguments": "https://music.youtube.com/watch?v=AjXQiKP5kMs",
            "ogType": "video.other",
            "urlApplinksIos": "vnd.youtube.music://music.youtube.com/watch?
↪v=AjXQiKP5kMs&feature=applinks",
            "urlApplinksAndroid": "vnd.youtube.music://music.youtube.com/watch?
↪v=AjXQiKP5kMs&feature=applinks",
            "urlTwitterIos": "vnd.youtube.music://music.youtube.com/watch?
↪v=AjXQiKP5kMs&feature=twitter-deep-link",
            "urlTwitterAndroid": "vnd.youtube.music://music.youtube.com/watch?
↪v=AjXQiKP5kMs&feature=twitter-deep-link",
            "twitterCardType": "player",
            "twitterSiteHandle": "@YouTubeMusic",
            "schemaDotOrgType": "http://schema.org/VideoObject",
            "noindex": true,
            "unlisted": true,
            "paid": false,
            "familySafe": true,
            "pageOwnerDetails": {
                "name": "Music Library Uploads",
                "externalChannelId": "UCvCk2zFqkCYzpnSgWfx0qOg",
                "youtubeProfileUrl": "http://www.youtube.com/channel/
↪UCvCk2zFqkCYzpnSgWfx0qOg"
            },
            "videoDetails": {
                "externalVideoId": "AjXQiKP5kMs",
                "durationSeconds": "246",
                "durationIso8601": "PT4M6S"
            },
            "linkAlternates": [
                {
                    "hrefUrl": "android-app://com.google.android.youtube/http/
↪youtube.com/watch?v=AjXQiKP5kMs"
                },
                {
                    "hrefUrl": "ios-app://544007664/http/youtube.com/watch?
↪v=AjXQiKP5kMs"
                },
                {
                    "hrefUrl": "https://www.youtube.com/oembed?format=json&url=https
↪%3A%2F%2Fmusic.youtube.com%2Fwatch%3Fv%3DAjXQiKP5kMs",
                    "title": "Sparks",
                    "alternateType": "application/json+oembed"
                },
                {
                    "hrefUrl": "https://www.youtube.com/oembed?format=xml&url=https
↪%3A%2F%2Fmusic.youtube.com%2Fwatch%3Fv%3DAjXQiKP5kMs",
```


(continues on next page)

(continued from previous page)

```
                    "title": "Sparks",
                    "alternateType": "text/xml+oembed"
                }
            ],
            "viewCount": "12",
            "publishDate": "1969-12-31",
            "category": "Music",
            "uploadDate": "1969-12-31"
        }
    }
}
```

YTMusic.**get_song_related**(*browseId: str*)

Gets related content for a song. Equivalent to the content shown in the "Related" tab of the watch panel.

Parameters

browseId – The *related* key in the *get_watch_playlist* response.

Example:

```
[
  {
    "title": "You might also like",
    "contents": [
      {
        "title": "High And Dry",
        "videoId": "7fv84nPfTH0",
        "artists": [{
            "name": "Radiohead",
            "id": "UCr_iyUANcn9OX_yy9piYoLw"
          }],
        "thumbnails": [
          {
            "url": "https://lh3.googleusercontent.com/
↪TWWT47cHLv3yAugk4h9eOzQ46FHmXc_g-KmBVy2d4sbg_F-Gv6xrPglztRVzp8D_l-yzOnvh-
↪QToM8s=w60-h60-l90-rj",
            "width": 60,
            "height": 60
          }
        ],
        "isExplicit": false,
        "album": {
          "name": "The Bends",
          "id": "MPREb_xsmDKhqhQrG"
        }
      }
    ]
  },
  {
    "title": "Recommended playlists",
    "contents": [
      {
        "title": "'90s Alternative Rock Hits",
```

(continues on next page)

(continued from previous page)


```
        "playlistId": "RDCLAK5uy_m_h-nx7OCFaq9AlyXv78lG0AuloqW_NUA",
        "thumbnails": [...],
        "description": "Playlist • YouTube Music"
      }
    ]
  },
  {
    "title": "Similar artists",
    "contents": [
      {
        "title": "Noel Gallagher",
        "browseId": "UCu7yYcX_wIZgG9azR3PqrxA",
        "subscribers": "302K",
        "thumbnails": [...]
      }
    ]
  },
  {
    "title": "Oasis",
    "contents": [
      {
        "title": "Shakermaker",
        "year": "2014",
        "browseId": "MPREb_WNGQWp5czjD",
        "thumbnails": [...]
      }
    ]
  },
  {
    "title": "About the artist",
    "contents": "Oasis were a rock band consisting of Liam Gallagher, Paul ...␣
↪(full description shortened for documentation)"
  }
]
```


YTMusic.**get_lyrics**(*browseId: str*) → Dict

Returns lyrics of a song or video.

Parameters

browseId – Lyrics browse id obtained from *get_watch_playlist*

Returns

Dictionary with song lyrics.

Example:


```
{
    "lyrics": "Today is gonna be the day\nThat they're gonna throw it back to you\n
↪",
    "source": "Source: LyricFind"
}
```


YTMusic.**get_tasteprofile**() → Dict

Fetches suggested artists from taste profile (music.youtube.com/tasteprofile). Tasteprofile allows users to pick

artists to update their recommendations. Only returns a list of suggested artists, not the actual list of selected entries

Returns
Dictionary with artist and their selection & impression value

Example:

```
{
    "Drake": {
        "selectionValue": "tastebuilder_selection=/m/05mt_q"
        "impressionValue": "tastebuilder_impression=/m/05mt_q"
    }
}
```

YTMusic.**set_tasteprofile**(*artists: List[str]*, *taste_profile: Dict | None = None*) → None

Favorites artists to see more recommendations from the artist. Use *get_tasteprofile()* to see which artists are available to be recommended

Parameters

- **artists** – A List with names of artists, must be contained in the tasteprofile
- **taste_profile** – tasteprofile result from *get_tasteprofile()*. Pass this if you call *get_tasteprofile()* anyway to save an extra request.

Returns
None if successful

5.3.4 Explore

YTMusic.**get_mood_categories**() → Dict

Fetch "Moods & Genres" categories from YouTube Music.

Returns
Dictionary of sections and categories.

Example:

```
{
    'For you': [
        {
            'params': 'ggMPOg1uX1ZwN0pHT2NBT1Fk',
            'title': '1980s'
        },
        {
            'params': 'ggMPOg1uXzZQbDB5eThLRTQ3',
            'title': 'Feel Good'
        },
        ...
    ],
    'Genres': [
        {
            'params': 'ggMPOg1uXzVLbmZnaWI4STNs',
            'title': 'Dance & Electronic'
        },
```

(continues on next page)

(continued from previous page)

```
        {
            'params': 'ggMPOg1uX3NjZllsNGVEMkZo',
            'title': 'Decades'
        },
        ...
    ],
    'Moods & moments': [
        {
            'params': 'ggMPOg1uXzVuc0dnZlhpV3Ba',
            'title': 'Chill'
        },
        {
            'params': 'ggMPOg1uX2ozUHlwbWM3ajNq',
            'title': 'Commute'
        },
        ...
    ],
}
```

YTMusic.**get_mood_playlists**(*params: str*) → List[Dict]

Retrieve a list of playlists for a given "Moods & Genres" category.

Parameters

params – params obtained by *get_mood_categories()*

Returns

List of playlists in the format of *get_library_playlists()*

YTMusic.**get_charts**(*country: str = 'ZZ'*) → Dict

Get latest charts data from YouTube Music: Top songs, top videos, top artists and top trending videos. Global charts have no Trending section, US charts have an extra Genres section with some Genre charts.

Parameters

country – ISO 3166-1 Alpha-2 country code. Default: ZZ = Global

Returns

Dictionary containing chart songs (only if authenticated with premium account), chart videos, chart artists and trending videos.

Example:

```
{
    "countries": {
        "selected": {
            "text": "United States"
        },
        "options": ["DE",
            "ZZ",
            "ZW"]
    },
    "songs": {
        "playlist": "VLPL4fGSI1pDJn6O1LS0XSdF3RyO0Rq_LDeI",
        "items": [
            {
                "title": "Outside (Better Days)",
```

(continues on next page)

(continued from previous page)


```
                "videoId": "oT79YlRtXDg",
                "artists": [
                    {
                        "name": "MO3",
                        "id": "UCdFt4Cvhr7Okaxo6hZg5K8g"
                    },
                    {
                        "name": "OG Bobby Billions",
                        "id": "UCLusb4T2tW3gOpJS1fJ-A9g"
                    }
                ],
                "thumbnails": [...],
                "isExplicit": true,
                "album": {
                    "name": "Outside (Better Days)",
                    "id": "MPREb_fX4Yv8frUNv"
                },
                "rank": "1",
                "trend": "up"
            }
        ]
    },
    "videos": {
        "playlist": "VLPL4fGSI1pDJn69On1f-8NAvX_CYlx7QyZc",
        "items": [
            {
                "title": "EVERY CHANCE I GET (Official Music Video) (feat. Lil Baby␣
↪& Lil Durk)",
                "videoId": "BTivsHlVcGU",
                "playlistId": "PL4fGSI1pDJn69On1f-8NAvX_CYlx7QyZc",
                "thumbnails": [],
                "views": "46M"
            }
        ]
    },
    "artists": {
        "playlist": null,
        "items": [
            {
                "title": "YoungBoy Never Broke Again",
                "browseId": "UCR28YDxjDE3ogQROaNdnRbQ",
                "subscribers": "9.62M",
                "thumbnails": [],
                "rank": "1",
                "trend": "neutral"
            }
        ]
    },
    "genres": [
        {
            "title": "Top 50 Pop Music Videos United States",
            "playlistId": "PL4fGSI1pDJn77aK7sAW2AT0oOzo5inWY8",
```


(continues on next page)

(continued from previous page)

```
            "thumbnails": []
        }
    ],
    "trending": {
        "playlist": "VLPLrEnWoR732-DtKgaDdnPkezM_nDidBU9H",
        "items": [
            {
                "title": "Permission to Dance",
                "videoId": "CuklIb9d3fI",
                "playlistId": "PLrEnWoR732-DtKgaDdnPkezM_nDidBU9H",
                "artists": [
                    {
                        "name": "BTS",
                        "id": "UC9vrvNSL3xcWGSkV86REBSg"
                    }
                ],
                "thumbnails": [],
                "views": "108M"
            }
        ]
    }
}
```

5.3.5 Watch

YTMusic.**get_watch_playlist**(*videoId: str | None = None*, *playlistId: str | None = None*, *limit=25*, *radio: bool = False*, *shuffle: bool = False*) → Dict[str, List[Dict] | str | None]

Get a watch list of tracks. This watch playlist appears when you press play on a track in YouTube Music.

Please note that the *INDIFFERENT* likeStatus of tracks returned by this endpoint may be either *INDIFFERENT* or *DISLIKE*, due to ambiguous data returned by YouTube Music.

Parameters

- **videoId** – videoId of the played video
- **playlistId** – playlistId of the played playlist or album
- **limit** – minimum number of watch playlist items to return
- **radio** – get a radio playlist (changes each time)
- **shuffle** – shuffle the input playlist. only works when the playlistId parameter is set at the same time. does not work if radio=True

Returns

List of watch playlist items. The counterpart key is optional and only appears if a song has a corresponding video counterpart (UI song/video switcher).

Example:

```
{
    "tracks": [
        {
          "videoId": "9mWr4c_ig54",
```

(continues on next page)

(continued from previous page)


```
          "title": "Foolish Of Me (feat. Jonathan Mendelsohn)",
          "length": "3:07",
          "thumbnail": [
            {
              "url": "https://lh3.googleusercontent.com/
↪ulK2YaLtOW0PzcN7ufltG6e4ae3WZ9Bvg8CCwhe6LOccu1lCKxJy2r5AsYrsHeMBSLrGJCNpJqXgwczk=w60-
↪h60-l90-rj",
              "width": 60,
              "height": 60
            }...
          ],
          "feedbackTokens": {
            "add": "AB9zfpIGg9XN4u2iJ...",
            "remove": "AB9zfpJdzWLcdZtC..."
          },
          "likeStatus": "INDIFFERENT",
          "videoType": "MUSIC_VIDEO_TYPE_ATV",
          "artists": [
            {
              "name": "Seven Lions",
              "id": "UCYd2yzYRx7b9FYnBSlbnknA"
            },
            {
              "name": "Jason Ross",
              "id": "UCVCD9Iwnqn2ipN9JIF6B-nA"
            },
            {
              "name": "Crystal Skies",
              "id": "UCTJZESxeZ0J_M7JXyFUVmvA"
            }
          ],
          "album": {
            "name": "Foolish Of Me",
            "id": "MPREb_C8aRK1qmsDJ"
          },
          "year": "2020",
          "counterpart": {
            "videoId": "E0S4W34zFMA",
            "title": "Foolish Of Me [ABGT404] (feat. Jonathan Mendelsohn)",
            "length": "3:07",
            "thumbnail": [...],
            "feedbackTokens": null,
            "likeStatus": "LIKE",
            "artists": [
              {
                "name": "Jason Ross",
                "id": null
              },
              {
                "name": "Seven Lions",
                "id": null
              },
```


(continues on next page)

(continued from previous page)

```
                {
                  "name": "Crystal Skies",
                  "id": null
                }
              ],
              "views": "6.6K"
            }
          },...
    ],
    "playlistId": "RDAMVM4y33h81phKU",
    "lyrics": "MPLYt_HNNclO0Ddoc-17"
}
```

5.3.6 Library

YTMusic.**get_library_playlists**(*limit: int | None = 25*) → List[Dict]

Retrieves the playlists in the user's library.

Parameters

limit – Number of playlists to retrieve. *None* retrieves them all.

Returns

List of owned playlists.

Each item is in the following format:

```
{
    'playlistId': 'PLQwVIlKxHM6rz0fDJVv_0UlXGEWf-bFys',
    'title': 'Playlist title',
    'thumbnails: [...],
    'count': 5
}
```

YTMusic.**get_library_songs**(*limit: int = 25*, *validate_responses: bool = False*, *order: str | None = None*) → List[Dict]

Gets the songs in the user's library (liked videos are not included). To get liked songs and videos, use *get_liked_songs()*

Parameters

- **limit** – Number of songs to retrieve
- **validate_responses** – Flag indicating if responses from YTM should be validated and retried in case when some songs are missing. Default: False
- **order** – Order of songs to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of songs. Same format as *get_playlist()*

YTMusic.**get_library_albums**(*limit: int = 25*, *order: str | None = None*) → List[Dict]

Gets the albums in the user's library.

Parameters

- **limit** – Number of albums to return
- **order** – Order of albums to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns
List of albums.

Each item is in the following format:

```
{
  "browseId": "MPREb_G8AiyN7RvFg",
  "playlistId": "OLAK5uy_lKgoGvlrWhX0EIPavQUXxyPed8Cj38AWc",
  "title": "Beautiful",
  "type": "Album",
  "thumbnails": [...],
  "artists": [{
    "name": "Project 46",
    "id": "UCXFv36m62USAN5rnVct9B4g"
  }],
  "year": "2015"
}
```

YTMusic.**get_library_artists**(*limit: int = 25*, *order: str | None = None*) → List[Dict]

Gets the artists of the songs in the user's library.

Parameters

- **limit** – Number of artists to return
- **order** – Order of artists to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns
List of artists.

Each item is in the following format:

```
{
  "browseId": "UCxEqaQWosMHaTih-tgzDqug",
  "artist": "WildVibes",
  "subscribers": "2.91K",
  "thumbnails": [...]
}
```

YTMusic.**get_library_subscriptions**(*limit: int = 25*, *order: str | None = None*) → List[Dict]

Gets the artists the user has subscribed to.

Parameters

- **limit** – Number of artists to return
- **order** – Order of artists to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns
List of artists. Same format as *get_library_artists()*

YTMusic.**get_library_podcasts**(*limit: int = 25*, *order: str | None = None*) → List[Dict]

Get podcasts the user has added to the library

Parameters

- **limit** – Number of podcasts to return
- **order** – Order of podcasts to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of podcasts. New Episodes playlist is the first podcast returned, but only if subscribed to relevant podcasts.

Example:

```
[
    {
        "title": "New Episodes",
        "channel":
        {
            "id": null,
            "name": "Auto playlist"
        },
        "browseId": "VLRDPN",
        "podcastId": "RDPN",
        "thumbnails": [...]
    },
    {
        "title": "5 Minuten Harry Podcast",
        "channel":
        {
            "id": "UCDIDXF4WM1qQzerrxeEfSdA",
            "name": "coldmirror"
        },
        "browseId": "MPSPPLDvBqWb1UAGeEt9n6vFH_zdGw65Obf3sH",
        "podcastId": "PLDvBqWb1UAGeEt9n6vFH_zdGw65Obf3sH",
        "thumbnails": [...]
    }
]
```

YTMusic.**get_library_channels**(*limit: int = 25*, *order: str | None = None*) → List[Dict]

Get channels the user has added to the library

Parameters

- **limit** – Number of channels to return
- **order** – Order of channels to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of channels.

Example:

```
[
    {
        "browseId": "UCRFF8xw5dg9mL4r5ryFOtKw",
        "artist": "Jumpers Jump",
        "subscribers": "1.54M",
```

(continues on next page)

(continued from previous page)

```
        "thumbnails": [...]
    },
    {
        "browseId": "UCQ3f2_sO3NJyDkuCxCNSOVA",
        "artist": "BROWN BAG",
        "subscribers": "74.2K",
        "thumbnails": [...]
    }
]
```

YTMusic.**get_liked_songs**(*limit: int = 100*) → Dict

Gets playlist items for the 'Liked Songs' playlist

Parameters
limit – How many items to return. Default: 100

Returns
List of playlistItem dictionaries. See *get_playlist()*

YTMusic.**get_saved_episodes**(*limit: int = 100*) → Dict

Gets playlist items for the 'Liked Songs' playlist

Parameters
limit – How many items to return. Default: 100

Returns
List of playlistItem dictionaries. See *get_playlist()*

YTMusic.**get_history**() → List[Dict]

Gets your play history in reverse chronological order

Returns
List of playlistItems, see *get_playlist()* The additional property `played` indicates when the playlistItem was played The additional property `feedbackToken` can be used to remove items with *remove_history_items()*

YTMusic.**add_history_item**(*song*)

Add an item to the account's history using the playbackTracking URI obtained from *get_song()*.

Parameters
song – Dictionary as returned by *get_song()*

Returns
Full response. response.status_code is 204 if successful

YTMusic.**remove_history_items**(*feedbackTokens: List[str]*) → Dict

Remove an item from the account's history. This method does currently not work with brand accounts

Parameters
feedbackTokens – Token to identify the item to remove, obtained from *get_history()*

Returns
Full response

YTMusic.**rate_song**(*videoId: str*, *rating: str = 'INDIFFERENT'*) → Dict | None

Rates a song ("thumbs up"/"thumbs down" interactions on YouTube Music)

Parameters

- **videoId** – Video id
- **rating** – One of 'LIKE', 'DISLIKE', 'INDIFFERENT'

'INDIFFERENT' removes the previous rating and assigns no rating

Returns
Full response

YTMusic.**edit_song_library_status**(*feedbackTokens: List[str] | None = None*) → Dict

Adds or removes a song from your library depending on the token provided.

Parameters
feedbackTokens – List of feedbackTokens obtained from authenticated requests to endpoints that return songs (i.e. *get_album()*)

Returns
Full response

YTMusic.**rate_playlist**(*playlistId: str*, *rating: str = 'INDIFFERENT'*) → Dict

Rates a playlist/album ("Add to library"/"Remove from library" interactions on YouTube Music) You can also dislike a playlist/album, which has an effect on your recommendations

Parameters

- **playlistId** – Playlist id
- **rating** – One of 'LIKE', 'DISLIKE', 'INDIFFERENT'

'INDIFFERENT' removes the playlist/album from the library

Returns
Full response

YTMusic.**subscribe_artists**(*channelIds: List[str]*) → Dict

Subscribe to artists. Adds the artists to your library

Parameters
channelIds – Artist channel ids

Returns
Full response

YTMusic.**unsubscribe_artists**(*channelIds: List[str]*) → Dict

Unsubscribe from artists. Removes the artists from your library

Parameters
channelIds – Artist channel ids

Returns
Full response

YTMusic.**get_account_info**() → Dict

Gets information about the currently authenticated user's account.

Returns
Dictionary with user's account name, channel handle, and URL of their account photo.

Example:

```
{
    "accountName": "Sample User",
    "channelHandle": "@SampleUser
    "accountPhotoUrl": "https://yt3.ggpht.com/sample-user-photo"
}
```

5.3.7 Playlists

YTMusic.**get_playlist**(*playlistId: str*, *limit: int | None = 100*, *related: bool = False*, *suggestions_limit: int = 0*) → Dict

Returns a list of playlist items

Parameters

- **playlistId** – Playlist id
- **limit** – How many songs to return. *None* retrieves them all. Default: 100
- **related** – Whether to fetch 10 related playlists or not. Default: False
- **suggestions_limit** – How many suggestions to return. The result is a list of suggested playlist items (videos) contained in a "suggestions" key. 7 items are retrieved in each internal request. Default: 0

Returns

Dictionary with information about the playlist. The key `tracks` contains a List of playlistItem dictionaries

The result is in the following format:

```
{
  "id": "PLQwVIlKxHM6qv-o99iX9R85og7IzF9YS_",
  "privacy": "PUBLIC",
  "title": "New EDM This Week 03/13/2020",
  "thumbnails": [...]
  "description": "Weekly r/EDM new release roundup. Created with github.com/sigma67/
↪spotifyplaylist_to_gmusic",
  "author": "sigmatics",
  "year": "2020",
  "duration": "6+ hours",
  "duration_seconds": 52651,
  "trackCount": 237,
  "suggestions": [
      {
        "videoId": "HLCsfOykA94",
        "title": "Mambo (GATTÜSO Remix)",
        "artists": [{
            "name": "Nikki Vianna",
            "id": "UCMW5eSIO1moVlIBLQzq4PnQ"
          }],
        "album": {
          "name": "Mambo (GATTÜSO Remix)",
          "id": "MPREb_jLeQJsd7U9w"
```

(continues on next page)

(continued from previous page)


```
        },
        "likeStatus": "LIKE",
        "thumbnails": [...],
        "isAvailable": true,
        "isExplicit": false,
        "duration": "3:32",
        "duration_seconds": 212,
        "setVideoId": "to_be_updated_by_client"
      }
  ],
  "related": [
      {
        "title": "Presenting MYRNE",
        "playlistId": "RDCLAK5uy_mbdO3_xdD4NtU1rWI0OmvRSRZ8NH4uJCM",
        "thumbnails": [...],
        "description": "Playlist • YouTube Music"
      }
  ],
  "tracks": [
    {
      "videoId": "bjGppZKiuFE",
      "title": "Lost",
      "artists": [
        {
          "name": "Guest Who",
          "id": "UCkgCRdnnqWnUeIH7EIc3dBg"
        },
        {
          "name": "Kate Wild",
          "id": "UCwR2l3JfJbvB6aq0RnnJfWg"
        }
      ],
      "album": {
        "name": "Lost",
        "id": "MPREb_PxmzvDuqOnC"
      },
      "duration": "2:58",
      "likeStatus": "INDIFFERENT",
      "thumbnails": [...],
      "isAvailable": True,
      "isExplicit": False,
      "videoType": "MUSIC_VIDEO_TYPE_OMV",
      "feedbackTokens": {
        "add": "AB9zfpJxtvrU...",
        "remove": "AB9zfpKTyZ..."
    }
  ]
}
```


The setVideoId is the unique id of this playlist item and needed for moving/removing playlist items

YTMusic.**create_playlist**(*title: str*, *description: str*, *privacy_status: str = 'PRIVATE'*, *video_ids: List | None = None*, *source_playlist: str | None = None*) → str | Dict

Creates a new empty playlist and returns its id.

Parameters

- **title** – Playlist title
- **description** – Playlist description
- **privacy_status** – Playlists can be 'PUBLIC', 'PRIVATE', or 'UNLISTED'. Default: 'PRIVATE'
- **video_ids** – IDs of songs to create the playlist with
- **source_playlist** – Another playlist whose songs should be added to the new playlist

Returns

ID of the YouTube playlist or full response if there was an error

YTMusic.**edit_playlist**(*playlistId: str*, *title: str | None = None*, *description: str | None = None*, *privacyStatus: str | None = None*, *moveItem: Tuple[str, str] | None = None*, *addPlaylistId: str | None = None*, *addToTop: bool | None = None*) → str | Dict

Edit title, description or privacyStatus of a playlist. You may also move an item within a playlist or append another playlist to this playlist.

Parameters

- **playlistId** – Playlist id
- **title** – Optional. New title for the playlist
- **description** – Optional. New description for the playlist
- **privacyStatus** – Optional. New privacy status for the playlist
- **moveItem** – Optional. Move one item before another. Items are specified by setVideoId, see *get_playlist()*
- **addPlaylistId** – Optional. Id of another playlist to add to this playlist
- **addToTop** – Optional. Change the state of this playlist to add items to the top of the playlist (if True) or the bottom of the playlist (if False - this is also the default of a new playlist).

Returns

Status String or full response

YTMusic.**delete_playlist**(*playlistId: str*) → str | Dict

Delete a playlist.

Parameters

playlistId – Playlist id

Returns

Status String or full response

YTMusic.**add_playlist_items**(*playlistId: str*, *videoIds: List[str] | None = None*, *source_playlist: str | None = None*, *duplicates: bool = False*) → str | Dict

Add songs to an existing playlist

Parameters

- **playlistId** – Playlist id
- **videoIds** – List of Video ids
- **source_playlist** – Playlist id of a playlist to add to the current playlist (no duplicate check)

- **duplicates** – If True, duplicates will be added. If False, an error will be returned if there are duplicates (no items are added to the playlist)

Returns
Status String and a dict containing the new setVideoId for each videoId or full response

YTMusic.**remove_playlist_items**(*playlistId: str*, *videos: List[Dict]*) → str | Dict

Remove songs from an existing playlist

Parameters

- **playlistId** – Playlist id
- **videos** – List of PlaylistItems, see *get_playlist()*. Must contain videoId and setVideoId

Returns
Status String or full response

5.3.8 Podcasts

YTMusic.**get_channel**(*channelId: str*) → Dict

Get information about a podcast channel (episodes, podcasts). For episodes, a maximum of 10 episodes are returned, the full list of episodes can be retrieved via *get_channel_episodes()*

Parameters
channelId – channel id

Returns
Dict containing channel info

Example:

```
{
    "title": 'Stanford Graduate School of Business',
    "thumbnails": [...]
    "episodes":
    {
        "browseId": "UCGwuxdEeCf0TIA2RbPOj-8g",
        "results":
        [
            {
                "index": 0,
                "title": "The Brain Gain: The Impact of Immigration on American␣
↪Innovation with Rebecca Diamond",
                "description": "Immigrants' contributions to America ...",
                "duration": "24 min",
                "videoId": "TS3Ovvk3VAA",
                "browseId": "MPEDTS3Ovvk3VAA",
                "videoType": "MUSIC_VIDEO_TYPE_PODCAST_EPISODE",
                "date": "Mar 6, 2024",
                "thumbnails": [...]
            },
        ],
        "params": "6gPiAUdxWUJXcFlCQ3BN..."
    },
    "podcasts":
```

(continues on next page)

(continued from previous page)


```
    {
        "browseId": null,
        "results":
        [
            {
                "title": "Stanford GSB Podcasts",
                "channel":
                {
                    "id": "UCGwuxdEeCf0TIA2RbPOj-8g",
                    "name": "Stanford Graduate School of Business"
                },
                "browseId": "MPSPPLxq_lXOUlvQDUNyoBYLkN8aVt5yAwEtG9",
                "podcastId": "PLxq_lXOUlvQDUNyoBYLkN8aVt5yAwEtG9",
                "thumbnails": [...]
            }
       ]
    }
}
```


YTMusic.**get_channel_episodes**(*channelId: str*, *params: str*) → List[Dict]

Get all channel episodes. This endpoint is currently unlimited

Parameters

- **channelId** – channelId of the user
- **params** – params obtained by *get_channel()*

Returns

List of channel episodes in the format of *get_channel()* "episodes" key

YTMusic.**get_podcast**(*playlistId: str*, *limit: int | None = 100*) → Dict

Returns podcast metadata and episodes

Note: To add a podcast to your library, you need to call *rate_playlist* on it

Parameters

- **playlistId** – Playlist id
- **limit** – How many songs to return. *None* retrieves them all. Default: 100

Returns

Dict with podcast information

Example:


```
{
    "author":
    {
        "name": "Stanford Graduate School of Business",
        "id": "UCGwuxdEeCf0TIA2RbPOj-8g"
    },
    "title": "Think Fast, Talk Smart: The Podcast",
```


(continues on next page)

(continued from previous page)


```
    "description": "Join Matt Abrahams, a lecturer of...",
    "saved": false,
    "episodes":
    [
        {
            "index": 0,
            "title": "132. Lean Into Failure: How to Make Mistakes That Work |␣
↪Think Fast, Talk Smart: Communication...",
            "description": "Effective and productive teams and...",
            "duration": "25 min",
            "videoId": "xAEGaW2my7E",
            "browseId": "MPEDxAEGaW2my7E",
            "videoType": "MUSIC_VIDEO_TYPE_PODCAST_EPISODE",
            "date": "Mar 5, 2024",
            "thumbnails": [...]
        }
    ]
}
```


YTMusic.**get_episode**(*videoId: str*) → Dict

Retrieve episode data for a single episode

Note: To save an episode, you need to call *add_playlist_items* to add it to the *SE* (saved episodes) playlist.

Parameters

videoId – browseId (MPED..) or videoId for a single episode

Returns

Dict containing information about the episode

The description elements are based on a custom dataclass, not shown in the example below The description text items also contain "n" to indicate newlines, removed below due to RST issues

Example:


```
{
    "author":
    {
        "name": "Stanford GSB Podcasts",
        "id": "MPSPPLxq_lXOUlvQDUNyoBYLkN8aVt5yAwEtG9"
    },
    "title": "124. Making Meetings Me...",
    "date": "Jan 16, 2024",
    "duration": "25 min",
    "saved": false,
    "playlistId": "MPSPPLxq_lXOUlvQDUNyoBYLkN8aVt5yAwEtG9",
    "description":
    [
        {
            "text": "Delve into why people hate meetings, ... Karin Reed ("
        },
```


(continues on next page)

(continued from previous page)


```
        {
            "text": "https://speakerdynamics.com/team/",
            "url": "https://speakerdynamics.com/team/"
        },
        {
            "text": ")Chapters:("
        },
        {
            "text": "00:00",
            "seconds": 0
        },
        {
            "text": ") Introduction Host Matt Abrahams...("
        },
        {
            "text": "01:30",
            "seconds": 90
        },
    ]
}
```


YTMusic.**get_episodes_playlist**(*playlist_id: str = 'RDPN'*) → Dict

Get all episodes in an episodes playlist. Currently the only known playlist is the "New Episodes" auto-generated playlist

Parameters

playlist_id – Playlist ID, defaults to "RDPN", the id of the New Episodes playlist

Returns

Dictionary in the format of *get_podcast()*

5.3.9 Uploads

YTMusic.**get_library_upload_songs**(*limit: int | None = 25*, *order: str | None = None*) → List[Dict]

Returns a list of uploaded songs

Parameters

- **limit** – How many songs to return. *None* retrieves them all. Default: 25
- **order** – Order of songs to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of uploaded songs.

Each item is in the following format:


```
{
  "entityId": "t_po_CICr2crg7OWpchDpjPjrBA",
  "videoId": "Uise6RPKoek",
  "artists": [{
    'name': 'Coldplay',
    'id': 'FEmusic_library_privately_owned_artist_detaila_po_
```


(continues on next page)

(continued from previous page)

```
↪CICr2crg7OWpchIIY29sZHBsYXk',
  }],
  "title": "A Sky Full Of Stars",
  "album": "Ghost Stories",
  "likeStatus": "LIKE",
  "thumbnails": [...]
}
```

YTMusic.**get_library_upload_artists**(*limit: int | None = 25*, *order: str | None = None*) → List[Dict]

Gets the artists of uploaded songs in the user's library.

Parameters

- **limit** – Number of artists to return. *None* retrieves them all. Default: 25
- **order** – Order of artists to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of artists as returned by *get_library_artists()*

YTMusic.**get_library_upload_albums**(*limit: int | None = 25*, *order: str | None = None*) → List[Dict]

Gets the albums of uploaded songs in the user's library.

Parameters

- **limit** – Number of albums to return. *None* retrives them all. Default: 25
- **order** – Order of albums to return. Allowed values: 'a_to_z', 'z_to_a', 'recently_added'. Default: Default order.

Returns

List of albums as returned by *get_library_albums()*

YTMusic.**get_library_upload_artist**(*browseId: str*, *limit: int = 25*) → List[Dict]

Returns a list of uploaded tracks for the artist.

Parameters

- **browseId** – Browse id of the upload artist, i.e. from *get_library_upload_songs()*
- **limit** – Number of songs to return (increments of 25).

Returns

List of uploaded songs.

Example List:

```
[
  {
    "entityId": "t_po_CICr2crg7OWpchDKwoakAQ",
    "videoId": "Dtffhy8WJgw",
    "title": "Hold Me (Original Mix)",
    "artists": [
      {
        "name": "Jakko",
        "id": "FEmusic_library_privately_owned_artist_detaila_po_
↪CICr2crg7OWpchIFamFra28"
      }
```

(continues on next page)

(continued from previous page)

```
    ],
    "album": null,
    "likeStatus": "LIKE",
    "thumbnails": [...]
  }
]
```

YTMusic.**get_library_upload_album**(*browseId: str*) → Dict

Get information and tracks of an album associated with uploaded tracks

Parameters

browseId – Browse id of the upload album, i.e. from i.e. from *get_library_upload_songs()*

Returns

Dictionary with title, description, artist and tracks.

Example album:

```
{
  "title": "18 Months",
  "type": "Album",
  "thumbnails": [...],
  "trackCount": 7,
  "duration": "24 minutes",
  "audioPlaylistId": "MLPRb_po_55chars",
  "tracks": [
    {
      "entityId": "t_po_22chars",
      "videoId": "FVo-UZoPygI",
      "title": "Feel So Close",
      "duration": "4:15",
      "duration_seconds": 255,
      "artists": None,
      "album": {
        "name": "18 Months",
        "id": "FEmusic_library_privately_owned_release_detailb_po_55chars"
      },
      "likeStatus": "INDIFFERENT",
      "thumbnails": None
    },
```

YTMusic.**upload_song**(*filepath: str*) → str | Response

Uploads a song to YouTube Music

Parameters

filepath – Path to the music file (mp3, m4a, wma, flac or ogg)

Returns

Status String or full response

YTMusic.**delete_upload_entity**(*entityId: str*) → str | Dict

Deletes a previously uploaded song or album

Parameters

entityId – The entity id of the uploaded song or album, e.g. retrieved from

get_library_upload_songs()

Returns
Status String or error

5.4 FAQ

Frequently asked questions for `ytmusicapi`. Contributions are welcome, please submit a PR.

5.4.1 Setup

My library results are empty even though I set up my cookie correctly.

Please make sure that you don't have multiple Google accounts. `ytmusicapi` might be returning results from a different account which is currently empty. You can set your account using `X-Goog-AuthUser` in your headers file (numeric index) or by providing the id of a brand account with `ytmusic = YTMusic(headers, "1234..")`. For more details see the *Reference*.

5.4.2 Usage

How do I add content to my library?

- **songs**: edit_song_library_status . Liking a song using rate_song does *not* add it to your library, only to your liked songs playlist.
- **albums, playlists**: rate_playlist
- **artists**: subscribe_artists . This will add the artist to your Subscriptions tab. The Artists tab is determined by the songs/albums you have added to your library.
- **podcasts**: rate_playlist
- **episodes**: add_playlist_items("SE", episode_id)

How can I get the radio playlist for a song, video, playlist or album?

- **songs, videos**: `RDAMVM` + `videoId`
- **playlists, albums**: `RDAMPL` + `playlistId`

See also What is a browseId below.

How can I get the shuffle playlist for a playlist or album?

Use get_watch_playlist_shuffle with the `playlistId` or `audioPlaylistId` (albums).

How can I get all my public playlists in a single request?

Call get_user_playlists with your own `channelId`.

Can I download songs?

You can use youtube-dl for this purpose.

How do I package ytmusicapi with `pyinstaller`?

To package ytmusicapi correctly, you need to add the locales files to your executable.

You can use `--add-data path-to-ytmusicapi/locales` or `--collect-all ytmusicapi` to accomplish this.

5.4.3 YouTube Music API Internals

Is there a difference between songs and videos?

Yes. Videos are regular videos from YouTube, which can be uploaded by any user. Songs are actual songs uploaded by artists.

You can also add songs to your library, while you can't add videos.

Is there a rate limit?

There most certainly is, although you shouldn't run into it during normal usage. See related issues:

- Creating playlists
- Uploads

What is a browseId?

A `browseId` is an internal, globally unique identifier used by YouTube Music for browsable content.

In most cases you can compose it yourself:

Prefix	Main	Content
VM	playlistId	Playlist
RDAMVM	playlistId	Video-based Radio
RDAMPL	videoId	Playlist-based Radio
MPLA	channelId	Artist
MPRE	custom	Album
MPSP	playlistId	Podcast
MPED	videoId	Episode

Which videoTypes exist and what do they mean?

`videoType` is prefixed with `MUSIC_VIDEO_TYPE_`, i.e. `MUSIC_VIDEO_TYPE_OMV`. Currently the following variants of `videoType` are known:

- `OMV`: Original Music Video - uploaded by original artist with actual video content
- `UGC`: User Generated Content - uploaded by regular YouTube user
- `ATV`: High quality song uploaded by original artist with cover image
- `OFFICIAL_SOURCE_MUSIC`: Official video content, but not for a single track

Why is ytmusicapi returning more results than requested with the limit parameter?

YouTube Music always returns increments of a specific pagination value, usually between 20 and 100 items at a time. This is the case if a ytmusicapi method supports the `limit` parameter. The default value of the `limit` parameter indicates the server-side pagination increment. ytmusicapi will keep fetching continuations from the server until it has reached at least the `limit` parameter, and return all of these results.

Which values can I use for languages?

The *language* parameter determines the language of the returned results. `ytmusicapi` only supports a subset of the languages supported by YouTube Music, as translations need to be done manually. Contributions are welcome, see here for instructions.

For the list of values you can use for the `language` parameter, see below:

Language	Value
Arabic	ar
German	de
English (default)	en
Spanish	es
French	fr
Hindi	hi
Italian	it
Japanese	ja
Korean	ko
Dutch	nl
Portuguese	pt
Russian	ru
Turkish	tr
Urdu	ur
Chinese (Mainland)	zh_CN
Chinese (Taiwan)	zh_TW

Which values can I use for locations?

Pick a value from the list below for your desired location and pass it using the `location` parameter.

Location	Value
Algeria	DZ
Argentina	AR
Australia	AU
Austria	AT
Azerbaijan	AZ
Bahrain	BH
Bangladesh	BD
Belarus	BY
Belgium	BE
Bolivia	BO
Bosnia and Herzegovina	BA
Brazil	BR
Bulgaria	BG
Cambodia	KH
Canada	CA
Chile	CL
Colombia	CO
Costa Rica	CR
Croatia	HR
Cyprus	CY
Czechia	CZ
Denmark	DK
Dominican Republic	DO
Ecuador	EC
Egypt	EG
El Salvador	SV
Estonia	EE
Finland	FI
France	FR
Georgia	GE
Germany	DE
Ghana	GH
Greece	GR
Guatemala	GT
Honduras	HN
Hong Kong	HK
Hungary	HU
Iceland	IS
India	IN
Indonesia	ID
Iraq	IQ
Ireland	IE
Israel	IL
Italy	IT
Jamaica	JM
Japan	JP
Jordan	JO

continues on next page

Table 1 – continued from previous page

Kazakhstan	KZ
Kenya	KE
Kuwait	KW
Laos	LA
Latvia	LV
Lebanon	LB
Libya	LY
Liechtenstein	LI
Lithuania	LT
Luxembourg	LU
Malaysia	MY
Malta	MT
Mexico	MX
Montenegro	ME
Morocco	MA
Nepal	NP
Netherlands	NL
New Zealand	NZ
Nicaragua	NI
Nigeria	NG
North Macedonia	MK
Norway	NO
Oman	OM
Pakistan	PK
Panama	PA
Papua New Guinea	PG
Paraguay	PY
Peru	PE
Philippines	PH
Poland	PL
Portugal	PT
Puerto Rico	PR
Qatar	QA
Romania	RO
Russia	RU
Saudi Arabia	SA
Senegal	SN
Serbia	RS
Singapore	SG
Slovakia	SK
Slovenia	SI
South Africa	ZA
South Korea	KR
Spain	ES
Sri Lanka	LK
Sweden	SE
Switzerland	CH
Taiwan	TW
Tanzania	TZ
Thailand	TH
Tunisia	TN
Turkey	TR

continues on next page

Table 1 – continued from previous page

Uganda	UG
Ukraine	UA
United Arab Emirates	AE
United Kingdom	GB
United States	US
Uruguay	UY
Venezuela	VE
Vietnam	VN
Yemen	YE
Zimbabwe	ZW

INDEX

Symbols

__init__() (*ytmusicapi.YTMusic method*), 13

A

add_history_item() (*ytmusicapi.YTMusic method*), 39
add_playlist_items() (*ytmusicapi.YTMusic method*), 43

C

create_playlist() (*ytmusicapi.YTMusic method*), 42

D

delete_playlist() (*ytmusicapi.YTMusic method*), 43
delete_upload_entity() (*ytmusicapi.YTMusic method*), 49

E

edit_playlist() (*ytmusicapi.YTMusic method*), 43
edit_song_library_status() (*ytmusicapi.YTMusic method*), 40

G

get_account_info() (*ytmusicapi.YTMusic method*), 40
get_album() (*ytmusicapi.YTMusic method*), 22
get_album_browse_id() (*ytmusicapi.YTMusic method*), 24
get_artist() (*ytmusicapi.YTMusic method*), 20
get_artist_albums() (*ytmusicapi.YTMusic method*), 22
get_channel() (*ytmusicapi.YTMusic method*), 44
get_channel_episodes() (*ytmusicapi.YTMusic method*), 45
get_charts() (*ytmusicapi.YTMusic method*), 32
get_episode() (*ytmusicapi.YTMusic method*), 46
get_episodes_playlist() (*ytmusicapi.YTMusic method*), 47
get_history() (*ytmusicapi.YTMusic method*), 39
get_home() (*ytmusicapi.YTMusic method*), 19
get_library_albums() (*ytmusicapi.YTMusic method*), 36
get_library_artists() (*ytmusicapi.YTMusic method*), 37
get_library_channels() (*ytmusicapi.YTMusic method*), 38
get_library_playlists() (*ytmusicapi.YTMusic method*), 36
get_library_podcasts() (*ytmusicapi.YTMusic method*), 37
get_library_songs() (*ytmusicapi.YTMusic method*), 36
get_library_subscriptions() (*ytmusicapi.YTMusic method*), 37
get_library_upload_album() (*ytmusicapi.YTMusic method*), 49
get_library_upload_albums() (*ytmusicapi.YTMusic method*), 48
get_library_upload_artist() (*ytmusicapi.YTMusic method*), 48
get_library_upload_artists() (*ytmusicapi.YTMusic method*), 48
get_library_upload_songs() (*ytmusicapi.YTMusic method*), 47
get_liked_songs() (*ytmusicapi.YTMusic method*), 39
get_lyrics() (*ytmusicapi.YTMusic method*), 30
get_mood_categories() (*ytmusicapi.YTMusic method*), 31
get_mood_playlists() (*ytmusicapi.YTMusic method*), 32
get_playlist() (*ytmusicapi.YTMusic method*), 41
get_podcast() (*ytmusicapi.YTMusic method*), 45
get_saved_episodes() (*ytmusicapi.YTMusic method*), 39
get_search_suggestions() (*ytmusicapi.YTMusic method*), 17
get_song() (*ytmusicapi.YTMusic method*), 25
get_song_related() (*ytmusicapi.YTMusic method*), 29
get_tasteprofile() (*ytmusicapi.YTMusic method*), 30
get_user() (*ytmusicapi.YTMusic method*), 24
get_user_playlists() (*ytmusicapi.YTMusic method*), 25

get_watch_playlist() (*ytmusicapi.YTMusic method*), 34

R

rate_playlist() (*ytmusicapi.YTMusic method*), 40
rate_song() (*ytmusicapi.YTMusic method*), 39
remove_history_items() (*ytmusicapi.YTMusic method*), 39
remove_playlist_items() (*ytmusicapi.YTMusic method*), 44

S

search() (*ytmusicapi.YTMusic method*), 15
set_tasteprofile() (*ytmusicapi.YTMusic method*), 31
setup() (*in module ytmusicapi*), 14
setup_oauth() (*in module ytmusicapi*), 14
subscribe_artists() (*ytmusicapi.YTMusic method*), 40

U

unsubscribe_artists() (*ytmusicapi.YTMusic method*), 40
upload_song() (*ytmusicapi.YTMusic method*), 49

Y

YTMusic (*class in ytmusicapi*), 13