ytmusicapi

Release 0.8.1

Jul 09, 2020

Contents

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 1

Features

Browsing:

- get artist information and releases (songs, videos, albums, singles)
- get albums
- search (including all filters)

Library management:

- get library contents: playlists, songs, artists, albums and subscriptions
- add/remove library content: rate songs, albums and playlists, subscribe/unsubscribe artists

Playlists:

- create, delete, and modify playlists
- get playlist contents, add/remove tracks

Uploads:

- Upload songs and remove them again
- List uploaded songs, artists and albums

CHAPTER 2

Usage

```
from ytmusicapi import YTMusic

ytmusic = YTMusic('headers_auth.json')
playlistId = ytmusic.create_playlist("test", "test description")
search_results = ytmusic.search("Oasis Wonderwall")
ytmusic.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 3

Contents

3.1 Setup

3.1.1 Installation

```
pip install ytmusicapi
```

3.1.2 Authenticated requests

To run authenticated requests you need to set up you need to copy your request headers from a POST request in your YTMusic Web Client. To do so, follow these steps:

- Open https://music.youtube.com in Firefox
- Go to the developer tools (Ctrl-Shift-I) and find an authenticated POST request. You can filter for /browse to easily find a suitable request.
- Copy the request headers (right click > copy > copy request headers)

Now call `YTMusic.setup()` with the parameter `filepath=headers_auth.json` and paste the request headers to the terminal input. If you don't want terminal interaction you can pass the request headers with the `headers_raw` parameter.

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 `headers_auth.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).

3.1.3 Manual file creation

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

```
{
    "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"
}
```

3.2 Usage

3.2.1 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`

3.2.2 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('headers_auth.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('headers_auth.json', 101234161234936123473)
```

3.3 Reference

Reference for the YTMusic class.

class ytmusicapi.**YTMusic**(*auth: str = None*, *user: str = None*, *proxies: dict = None*, *language: str = 'en'*)

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 = None*, *user: str = None*, *proxies: dict = None*, *language: str = 'en'*)

Create a new instance to interact with YouTube Music.

Parameters

- **auth** – Optional. Provide a string or path to file. Authentication credentials are needed to manage your library. Should be an adjusted version of *headers_auth.json.example* in the project root. 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 and selecting your brand account. The user ID will be in the URL: https://myaccount.google.com/b/user_id/
- **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.

3.3.1 Setup

See also the *Setup* page

classmethod YTMusic.**setup**(*filepath: str = None*, *headers_raw: str = None*)

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

3.3.2 Search

YTMusic.**search**(*query: str*, *filter: str = None*) → List[Dict[KT, VT]]

Search YouTube music Returns up to 20 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'. Default: Default search, including all types of items.

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:


```
[
  {
    "videoId": "ZrOKjDZOtkA",
    "title": "Wonderwall (Remastered)",
    "artists": [
      {
        "name": "Oasis",
        "id": "UCmMUZbaYdNH0bEd1PAlAqsA"
      }
    ],
    "album": {
      "name": "(What's The Story) Morning Glory? (Remastered)",
      "id": "MPREb_9nqEki4ZDpp"
    },
    "duration": "4:19",
    "thumbnails": [...],
    "resultType": "song"
  }
]
```


3.3.3 Browsing

`YTMusic.`**`get_artist`**(*channelId: str*) → Dict[KT, VT]

Get information about an artist and their top releases (songs, albums, singles and videos). The top lists contain pointers for getting the full list of releases. For songs/videos, pass the browseId to *`get_playlist()`*. For albums/singles, pass browseId and params to *`get_artist_albums()`*.

Parameters **`channelId`** – channel id of the artist

Returns Dictionary with requested information.

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)"
```


(continues on next page)

(continued from previous page)


```
            }
        ]
    },
    "albums": {
        "results": [
            {
                "title": "Familiar To Millions",
                "thumbnails": [...],
                "year": "2018",
                "browseId": "MPREb_AYetWMZunqA"
            }
        ],
        "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
        "params": "6gPTAUNwc0JDbndLYlFBQV..."
    },
    "singles": {
        "results": [
            {
                "title": "Stand By Me (Mustique Demo)",
                "thumbnails": [...],
                "year": "2016",
                "browseId": "MPREb_7MPKLhibN5G"
            }
        ],
        "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
        "params": "6gPTAUNwc0JDbndLYlFBQV..."
    },
    "videos": {
        "results": [
            {
                "title": "Wonderwall",
                "thumbnails": [...],
                "views": "358M",
                "videoId": "bx1Bh8ZvH84",
                "playlistId": "PLMpM3Z0118S5xuNckw1HUcj1D021AnMEB"
            }
        ],
        "browseId": "VLPLMpM3Z0118S5xuNckw1HUcj1D021AnMEB"
    }
}
```


YTMusic.**get_artist_albums**(*channelId: str*, *params: str*) → List[Dict[KT, VT]]

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

Parameters

- **channelId** – channel Id of the artist
- **params** – params obtained by *get_artist()*

Returns List of albums or singles

Example:


```
{
    "browseId": "MPREb_0rtvKhqeCY0",
    "artist": "Armin van Buuren",
    "title": "This I Vow (feat. Mila Josef)",
```


(continues on next page)

(continued from previous page)

```
    "thumbnails": [...],
    "type": "EP",
    "year": "2020"
}
```

YTMusic.**get_album**(*browseId: str*) → Dict[KT, VT]

Get information and tracks of an album

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

Returns Dictionary with title, description, artist and tracks.

Each track is in the following format:

```
{
  "title": "Seven",
  "trackCount": "7",
  "durationMs": "1439579",
  "playlistId": "OLAK5uy_kGnhwT08mQMGw8fArBowdtlew3DpgUt9c",
  "releaseDate": {
    "year": 2016,
    "month": 10,
    "day": 28
  },
  "description": "Seven is ...",
  "thumbnails": [...],
  "artist": [
    {
      "name": "Martin Garrix",
      "id": "UCqJnSdHjKtfsrHi9aI-9d3g"
    }
  ],
  "tracks": [
    {
      "index": "1",
      "title": "WIEE (feat. Mesto)",
      "artists": "Martin Garrix",
      "videoId": "8xMNeXI9wxI",
      "lengthMs": "203406",
      "likeStatus": "INDIFFERENT"
    }
  ]
}
```

3.3.4 Library

YTMusic.**get_library_playlists**(*limit: int = 25*) → List[Dict[KT, VT]]

Retrieves the playlists in the user's library.

Parameters **limit** – Number of playlists to retrieve

Returns List of owned playlists.

Each item is in the following format:

```
{
    'playlistId': 'PLQwVIlKxHM6rz0fDJVv_0UlXGEWf-bFys',
```

(continues on next page)

(continued from previous page)

```
    'title': 'Playlist title',
    'thumbnails: [...],
    'count': 5
}
```

YTMusic.**get_library_songs**(*limit: int = 25*) → List[Dict[KT, VT]]
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

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

YTMusic.**get_library_artists**(*limit: int = 25*) → List[Dict[KT, VT]]
Gets the artists of the songs in the user's library.

Parameters **limit** – Number of artists to return

Returns List of artists.

Each item is in the following format:

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

YTMusic.**get_library_albums**(*limit: int = 25*) → List[Dict[KT, VT]]
Gets the albums in the user's library.

Parameters **limit** – Number of albums to return

Returns List of albums

YTMusic.**get_liked_songs**(*limit: int = 100*) → Dict[KT, VT]
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[KT, VT]]
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

YTMusic.**rate_song**(*videoId: str*, *rating: str = 'INDIFFERENT'*) → Dict[KT, VT]
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.**rate_playlist**(*playlistId: str*, *rating: str = 'INDIFFERENT'*) → Dict[KT, VT]
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[KT, VT]
Subscribe to artists. Adds the artists to your library

Parameters **channelIds** – Artist channel ids

Returns Full response

YTMusic.**unsubscribe_artists**(*channelIds: List[str]*) → Dict[KT, VT]
Unsubscribe from artists. Removes the artists from your library

Parameters **channelIds** – Artist channel ids

Returns Full response

3.3.5 Playlists

YTMusic.**get_playlist**(*playlistId: str*, *limit: int = 100*) → Dict[KT, VT]
Returns a list of playlist items

Parameters

- **playlistId** – Playlist id
- **limit** – How many songs to return. Default: 100

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

Each item 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",
  "trackCount": 237,
  "tracks": [
    {
```

(continues on next page)

(continued from previous page)

```
      "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": [...]
    }
  ]
}
```

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[T] = None*, *source_playlist: str = None*) → Union[str, Dict[KT, VT]]

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*, *description: str = None*, *privacyStatus: str = None*, *moveItem: Tuple[str, str] = None*, *addPlaylistId: str = None*) → Union[str, Dict[KT, VT]]

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

Returns Status String or full response

YTMusic.**delete_playlist**(*playlistId: str*) → Union[str, Dict[KT, VT]]
Delete a playlist.

Parameters **playlistId** – Playlist id

Returns Status String or full response

YTMusic.**add_playlist_items**(*playlistId: str, videoIds: List[str]*) → Union[str, Dict[KT, VT]]
Add songs to an existing playlist

Parameters

- **playlistId** – Playlist id
- **videoIds** – List of Video ids

Returns Status String or full response

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

3.3.6 Uploads

YTMusic.**get_library_upload_songs**(*limit: int = 25*) → List[Dict[KT, VT]]
Returns a list of uploaded songs

Parameters **limit** – How many songs to return. Default: 25

Returns List of uploaded songs.

Each item is in the following format:

```
{
  "entityId": "t_po_CICr2crg7OWpchDpjPjrBA",
  "videoId": "Uise6RPKoek",
  "artist": "Coldplay",
  "title": "A Sky Full Of Stars",
  "album": "Ghost Stories",
  "likeStatus": "LIKE",
  "thumbnails": [...]
}
```

YTMusic.**get_library_upload_artists**(*limit: int = 25*) → List[Dict[KT, VT]]
Gets the artists of uploaded songs in the user's library.

Parameters **limit** – Number of artists to return. Default: 25

Returns List of artists as returned by *get_library_artists()*

YTMusic.**get_library_upload_albums**(*limit: int = 25*) → List[Dict[KT, VT]]
Gets the albums of uploaded songs in the user's library.

Parameters **limit** – Number of albums to return. Default: 25

Returns List of albums as returned by *get_library_albums()*

YTMusic.**get_library_upload_artist**(*browseId: str*) → List[Dict[KT, VT]]
Returns a list of uploaded tracks for the artist.

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

Returns List of uploaded songs.

Example List:

```
[
  {
    "entityId": "t_po_CICr2crg7OWpchDKwoakAQ",
    "videoId": "Dtffhy8WJgw",
    "title": "Hold Me (Original Mix)",
    "artist": [
      {
        "name": "Jakko",
        "id": "FEmusic_library_privately_owned_artist_detaila_po_
↪CICr2crg7OWpchIFamFra28"
      }
    ],
    "album": null,
    "likeStatus": "LIKE",
    "thumbnails": [...]
  }
]
```

YTMusic.**get_library_upload_album**(*browseId: str*) → Dict[KT, VT]
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": "Hard To Stop - Single",
  "thumbnails": [...]
  "year": "2013",
  "trackCount": 1,
  "duration": "4 minutes, 2 seconds",
  "tracks": [
    {
      "entityId": "t_po_CICr2crg7OWpchDN6tnYBw",
      "videoId": "VBQVcjJM7ak",
      "title": "Hard To Stop (Vicetone x Ne-Yo x Daft Punk)",
      "likeStatus": "LIKE"
    }
  ]
}
```

YTMusic.**upload_song**(*filepath: str*) → Union[str, requests.models.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*) → Union[str, Dict[KT, VT]]
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

Index

Symbols

A

C

D

E

G

R

S

U

Y