

---

# **ytmusicapi**

***Release 0.7.0***

**May 03, 2020**



---

## Contents

---

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contents</b>	<b>7</b>
3.1	Setup . . . . .	7
3.2	Usage . . . . .	8
3.3	Reference . . . . .	8
	<b>Index</b>	<b>17</b>



The purpose of this library is to automate interactions with [YouTube Music](#), such as retrieving your library content or creating large playlists.

**This project is not supported nor endorsed by Google**



# CHAPTER 1

---

## Features

---

- **Browsing:** get artist information and releases (songs, videos, albums, singles), get albums
- **Library management:**
  - get the playlists, songs, artists, albums and subscriptions in your library
  - rate songs, albums and playlists
  - subscribe/unsubscribe artists
- **Playlists:** create, delete, and modify playlists and playlist items
- **Search:** Search for songs on YouTube Music
- **Uploads:**
  - Upload songs and remove them again
  - List uploaded songs, artists and albums





## CHAPTER 2

---

### Usage Example

---

For a complete documentation of available functions, refer to the *Reference*

```
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.



## 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()` and paste the request headers and it will create configuration file in the correct format in the current directory.

### 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",
```

(continues on next page)

(continued from previous page)

```
"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](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*)

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*)

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/](https://myaccount.google.com/b/user_id/)

### 3.3.1 Setup

See also the *Setup* page

**classmethod** YTMusic.setup (*filepath: 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.

**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": "UCmMUZbaYdNH0bEd1PA1AqsA"
```

(continues on next page)

(continued from previous page)

```

    }
  ],
  "album": {
    "name": "(What's The Story) Morning Glory? (Remastered)",
    "id": "MPREb_9nqEki4ZDpp"
  },
  "duration": "4:19",
  "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:

```

{
  "name": "Oasis",
  "description": "Oasis were ...",
  "views": "1,838,795,605",
  "songs": {
    "browseId": "VLPLMpM3Z0118S42R1npOhcjoakLIvlaqnS1",
    "results": [
      {
        "videoId": "ZrOKjDZotkA",
        "title": "Wonderwall (Remastered)",
        "artist": "Oasis",
        "album": "(What's The Story) Morning Glory? (Remastered)"
      }
    ]
  },
  "albums": {
    "results": [
      {
        "title": "Familiar To Millions",
        "year": "2018",
        "browseId": "MPREb_AYetWMZunqA"
      }
    ],
    "browseId": "UCmMUZbaYdNH0bEd1PAlAqsA",
    "params": "6gPTAUNwc0JDsndLYlFBQV..."
  },
  "singles": {
    "results": [
      {

```

(continues on next page)

(continued from previous page)

```

        "title": "Stand By Me (Mustique Demo)",
        "year": "2016",
        "browseId": "MPREb_7MPKLhibN5G"
    },
    ],
    "browseId": "UCmMUZbaYdNH0bEd1PA1AqsA",
    "params": "6gPTAUNwc0JDsndLY1FBQV..."
},
"videos": {
    "results": [
        {
            "title": "Wonderwall",
            "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)",
    "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:

```

{
    "index": "1",
    "title": "WIEE (feat. Mesto)",
    "artists": "Martin Garrix",
    "videoId": "8xMNeXI9wxI",
}

```

(continues on next page)

(continued from previous page)

```
"lengthMs": "203406"
}
```

### 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_0U1XGEWf-bFys',
  'title': 'Playlist title',
  '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"
}
```

`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** List of playlistItem dictionaries

Each item is in the following format:

```
{
  "id": "PLQwVilKxHM6qv-o99iX9R85og7IzF9YS_",
  "privacy": "PUBLIC",
  "title": "New EDM This Week 03/13/2020",
  "description": "Weekly r/EDM new release roundup. Created with github.com/
↪sigma67/spotifyplaylist_to_gmusic",
  "author": "sigmatic",
  "year": "2020",
  "duration": "6+ hours",
  "trackCount": 237,
  "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"
    }
  ]
}
```

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'*) → *str*  
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'

**Returns** ID of the YouTube playlist

`YTMusic.edit_playlist` (*playlistId: str, title: str = None, description: str = None, privacyStatus: str = None*) → *Union[str, Dict[KT, VT]]*  
Edit title, description or `privacyStatus` of a 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

**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"
}
```

`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` –

: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

:returns List of albums as returned by `get_library_albums()`

`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_uploaded_song(uploaded_song: Dict[KT, VT]) → Union[str, Dict[KT, VT]]`

Deletes a previously uploaded song

**Parameters** `uploaded_song` – The uploaded song to delete, e.g. retrieved from  
`get_library_upload_songs()`

**Returns** Status String or error

## Symbols

`__init__()` (*ytmusicapi.YTMusic method*), 9

## A

`add_playlist_items()` (*ytmusicapi.YTMusic method*), 15

## C

`create_playlist()` (*ytmusicapi.YTMusic method*), 14

## D

`delete_playlist()` (*ytmusicapi.YTMusic method*), 15

`delete_uploaded_song()` (*ytmusicapi.YTMusic method*), 16

## E

`edit_playlist()` (*ytmusicapi.YTMusic method*), 14

## G

`get_album()` (*ytmusicapi.YTMusic method*), 11

`get_artist()` (*ytmusicapi.YTMusic method*), 10

`get_artist_albums()` (*ytmusicapi.YTMusic method*), 11

`get_history()` (*ytmusicapi.YTMusic method*), 12

`get_library_albums()` (*ytmusicapi.YTMusic method*), 12

`get_library_artists()` (*ytmusicapi.YTMusic method*), 12

`get_library_playlists()` (*ytmusicapi.YTMusic method*), 12

`get_library_songs()` (*ytmusicapi.YTMusic method*), 12

`get_library_upload_albums()` (*ytmusicapi.YTMusic method*), 15

`get_library_upload_artists()` (*ytmusicapi.YTMusic method*), 15

`get_library_upload_songs()` (*ytmusicapi.YTMusic method*), 15

`get_liked_songs()` (*ytmusicapi.YTMusic method*), 12

`get_playlist()` (*ytmusicapi.YTMusic method*), 13

## R

`rate_playlist()` (*ytmusicapi.YTMusic method*), 13

`rate_song()` (*ytmusicapi.YTMusic method*), 13

`remove_playlist_items()` (*ytmusicapi.YTMusic method*), 15

## S

`search()` (*ytmusicapi.YTMusic method*), 9

`setup()` (*ytmusicapi.YTMusic class method*), 9

`subscribe_artists()` (*ytmusicapi.YTMusic method*), 13

## U

`unsubscribe_artists()` (*ytmusicapi.YTMusic method*), 13

`upload_song()` (*ytmusicapi.YTMusic method*), 16

## Y

`YTMusic` (*class in ytmusicapi*), 8