

---

# **spotipy Documentation**

*Release 2.0*

**Paul Lamere**

**Jan 20, 2020**



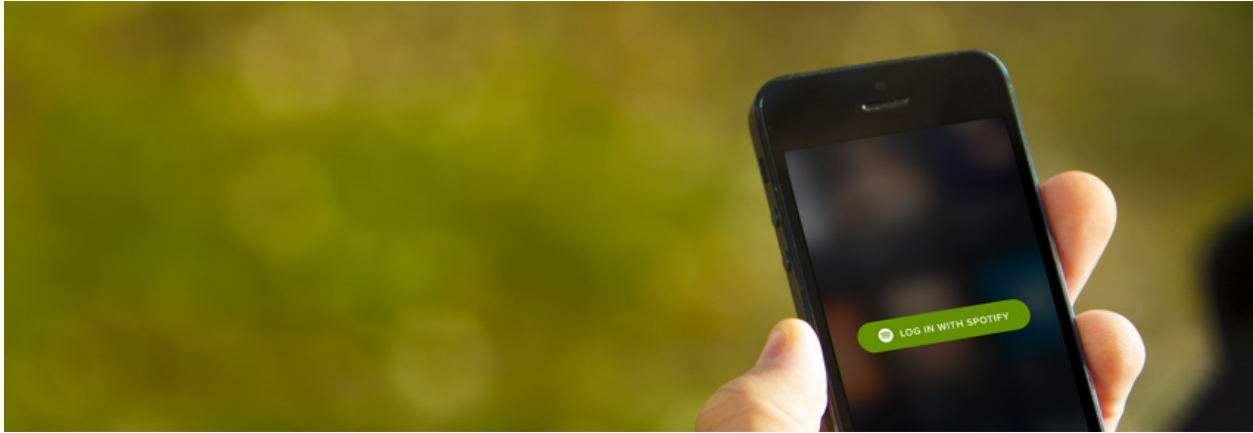
---

## Contents

---

<b>1</b>	<b>Features</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Getting Started</b>	<b>7</b>
<b>4</b>	<b>Authorization Code Flow</b>	<b>9</b>
<b>5</b>	<b>Client Credentials Flow</b>	<b>11</b>
<b>6</b>	<b>IDs URIs and URLs</b>	<b>13</b>
<b>7</b>	<b>Examples</b>	<b>15</b>
<b>8</b>	<b>More Examples</b>	<b>17</b>
<b>9</b>	<b>API Reference</b>	<b>19</b>
<b>10</b>	<b>client Module</b>	<b>21</b>
<b>11</b>	<b>oauth2 Module</b>	<b>33</b>
11.1	util Module .....	34
<b>12</b>	<b>Support</b>	<b>35</b>
<b>13</b>	<b>Contribute</b>	<b>37</b>
<b>14</b>	<b>License</b>	<b>39</b>
<b>15</b>	<b>Indices and tables</b>	<b>41</b>
	<b>Python Module Index</b>	<b>43</b>
	<b>Index</b>	<b>45</b>





*Spotipy* is a lightweight Python library for the [Spotify Web API](#). With *Spotipy* you get full access to all of the music data provided by the Spotify platform.

Here's a quick example of using *Spotipy* to list the names of all the albums released by the artist 'Birdy':

```
import spotipy

birdy_uri = 'spotify:artist:2WX2uTcsvV5OnS0inACecP'
spotify = spotipy.Spotify()

results = spotify.artist_albums(birdy_uri, album_type='album')
albums = results['items']
while results['next']:
    results = spotify.next(results)
    albums.extend(results['items'])

for album in albums:
    print(album['name'])
```

Here's another example showing how to get 30 second samples and cover art for the top 10 tracks for Led Zeppelin:

```
import spotipy

lz_uri = 'spotify:artist:36QJpDe2go2KgaRleHCdTp'

spotify = spotipy.Spotify()
results = spotify.artist_top_tracks(lz_uri)

for track in results['tracks'][:10]:
    print('track      : ' + track['name'])
    print('audio      : ' + track['preview_url'])
    print('cover art: ' + track['album']['images'][0]['url'])
    print()
```

Finally, here's an example that will get the URL for an artist image given the artist's name:

```
import spotipy
import sys

spotify = spotipy.Spotify()

if len(sys.argv) > 1:
    name = ' '.join(sys.argv[1:])
```

(continues on next page)

(continued from previous page)

```
else:
    name = 'Radiohead'

results = spotify.search(q='artist:' + name, type='artist')
items = results['artists']['items']
if len(items) > 0:
    artist = items[0]
    print(artist['name'], artist['images'][0]['url'])
```

# CHAPTER 1

---

## Features

---

*Spotipy* supports all of the features of the Spotify Web API including access to all end points, and support for user authorization. For details on the capabilities you are encouraged to review the [Spotify Web API](#) documentation.





## CHAPTER 2

---

### Installation

---

Install or upgrade *Spotipy* with:

```
pip install spotipy --upgrade
```

Or with:

```
easy_install spotipy
```

Or you can get the source from github at <https://github.com/plamere/spotipy>



---

## Getting Started

---

All methods require user authorization. You will need to register your app to get the credentials necessary to make authorized calls.

Even if your script does not have an accessible URL you will need to specify one when registering your application which the Spotify authentication server will redirect to after successful login. The URL doesn't need to be publicly accessible, so you can specify "<http://localhost>".

Register your app at [My Applications](#) and register the redirect URI mentioned in the above paragraph.

*spotipy* supports two authorization flows:

- The **Authorization Code flow** This method is suitable for long-running applications which the user logs into once. It provides an access token that can be refreshed.
- The **Client Credentials flow** The method makes it possible to authenticate your requests to the Spotify Web API and to obtain a higher rate limit than you would with the Authorization Code flow.



---

## Authorization Code Flow

---

This flow is suitable for long-running applications in which the user grants permission only once. It provides an access token that can be refreshed. Since the token exchange involves sending your secret key, perform this on a secure location, like a backend service, and not from a client such as a browser or from a mobile app.

To support the **Authorization Code Flow** *Spotipy* provides a utility method `util.prompt_for_user_token` that will attempt to authorize the user. You can pass your app credentials directly into the method as arguments:

```
util.prompt_for_user_token(username, scope, client_id='your-spotify-client-id', client_
↳secret='your-spotify-client-secret', redirect_uri='your-app-redirect-url')
```

or if you are reluctant to immortalize your app credentials in your source code, you can set environment variables like so:

```
export SPOTIPY_CLIENT_ID='your-spotify-client-id'
export SPOTIPY_CLIENT_SECRET='your-spotify-client-secret'
export SPOTIPY_REDIRECT_URI='your-app-redirect-url'
```

Call `util.prompt_for_user_token` method with the username and the desired scope (see [Using Scopes](#) for information about scopes) and credentials. After successfully authenticating your app, you can simply copy the “`http://localhost/?code=...`” URL from your browser and paste it to the console where your script is running. This will coordinate the user authorization via your web browser and callback to the `SPOTIPY_REDIRECT_URI` you were redirected to with the authorization token appended. The credentials are cached locally and are used to automatically re-authorized expired tokens.

Here’s an example of getting user authorization to read a user’s saved tracks:

```
import sys
import spotipy
import spotipy.util as util

scope = 'user-library-read'

if len(sys.argv) > 1:
    username = sys.argv[1]
```

(continues on next page)

(continued from previous page)

```
else:
    print("Usage: %s username" % (sys.argv[0],))
    sys.exit()

token = util.prompt_for_user_token(username, scope)

if token:
    sp = spotify.Spotify(auth=token)
    results = sp.current_user_saved_tracks()
    for item in results['items']:
        track = item['track']
        print(track['name'] + ' - ' + track['artists'][0]['name'])
else:
    print("Can't get token for", username)
```

---

## Client Credentials Flow

---

The Client Credentials flow is used in server-to-server authentication. Only endpoints that do not access user information can be accessed. The advantage here in comparison with requests to the Web API made without an access token, is that a higher rate limit is applied.

To support the **Client Credentials Flow** *Spotipy* provides a class `SpotifyClientCredentials` that can be used to authenticate requests like so:

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

client_credentials_manager = SpotifyClientCredentials()
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

playlists = sp.user_playlists('spotify')
while playlists:
    for i, playlist in enumerate(playlists['items']):
        print("%4d %s %s" % (i + 1 + playlists['offset'], playlist['uri'], playlist[
↪ 'name']))
    if playlists['next']:
        playlists = sp.next(playlists)
    else:
        playlists = None
```





---

## IDs URIs and URLs

---

*Spotipy* supports a number of different ID types:

- Spotify URI - The resource identifier that you can enter, for example, in the Spotify Desktop client's search box to locate an artist, album, or track. Example: `spotify:track:6rqhFgbbKwnb9MLmUQDhG6`
- Spotify URL - An HTML link that opens a track, album, app, playlist or other Spotify resource in a Spotify client. Example: <http://open.spotify.com/track/6rqhFgbbKwnb9MLmUQDhG6>
- Spotify ID - A base-62 number that you can find at the end of the Spotify URI (see above) for an artist, track, album, etc. Example: `6rqhFgbbKwnb9MLmUQDhG6`

In general, any *Spotipy* method that needs an artist, album, track or playlist ID will accept ids in any of the above form



Here are a few more examples of using *Spotipy*.

Add tracks to a playlist:

```
import sys

import spotipy
import spotipy.util as util

if len(sys.argv) > 3:
    username = sys.argv[1]
    playlist_id = sys.argv[2]
    track_ids = sys.argv[3:]
else:
    print("Usage: %s username playlist_id track_id ..." % (sys.argv[0],))
    sys.exit()

scope = 'playlist-modify-public'
token = util.prompt_for_user_token(username, scope)

if token:
    sp = spotipy.Spotify(auth=token)
    sp.trace = False
    results = sp.user_playlist_add_tracks(username, playlist_id, track_ids)
    print(results)
else:
    print("Can't get token for", username)
```

Shows the contents of every playlist owned by a user:

```
# shows a user's playlists (need to be authenticated via oauth)

import sys
import spotipy
import spotipy.util as util
```

(continues on next page)

```
def show_tracks(tracks):
    for i, item in enumerate(tracks['items']):
        track = item['track']
        print("  %d %32.32s %s" % (i, track['artists'][0]['name'],
            track['name']))

if __name__ == '__main__':
    if len(sys.argv) > 1:
        username = sys.argv[1]
    else:
        print("Whoops, need your username!")
        print("usage: python user_playlists.py [username]")
        sys.exit()

    token = util.prompt_for_user_token(username)

    if token:
        sp = spotipy.Spotify(auth=token)
        playlists = sp.user_playlists(username)
        for playlist in playlists['items']:
            if playlist['owner']['id'] == username:
                print()
                print(playlist['name'])
                print('  total tracks', playlist['tracks']['total'])
                results = sp.playlist(playlist['id'],
                    fields="tracks,next")
                tracks = results['tracks']
                show_tracks(tracks)
                while tracks['next']:
                    tracks = sp.next(tracks)
                    show_tracks(tracks)
            else:
                print("Can't get token for", username)
```

## CHAPTER 8

---

### More Examples

---

There are many more examples of how to use *Spotipy* in the [Examples Directory](#) on Github



## CHAPTER 9

---

API Reference

---





A simple and thin Python library for the Spotify Web API

```
class spotipy.client.Spotify (auth=None, requests_session=True,  
client_credentials_manager=None, proxies=None, re-  
quests_timeout=None)
```

Bases: object

Example usage:

```
import spotipy

urn = 'spotify:artist:3jOstUTkEu2JkjrRdBA5Gu'
sp = spotipy.Spotify()

sp.trace = True # turn on tracing
sp.trace_out = True # turn on trace out

artist = sp.artist(urn)
print(artist)

user = sp.user('plamere')
print(user)
```

```
__init__ (auth=None, requests_session=True, client_credentials_manager=None, proxies=None, re-  
quests_timeout=None)
```

Creates a Spotify API client.

### Parameters

- **auth** – An authorization token (optional)
- **requests\_session** – A Requests session object or a truthy value to create one. A falsy value disables sessions. It should generally be a good idea to keep sessions enabled for performance reasons (connection pooling).
- **client\_credentials\_manager** – SpotifyClientCredentials object

- **proxies** – Definition of proxies (optional)
- **requests\_timeout** – Tell Requests to stop waiting for a response after a given number of seconds

**album** (*album\_id*)

returns a single album given the album's ID, URIs or URL

**Parameters:**

- *album\_id* - the album ID, URI or URL

**album\_tracks** (*album\_id, limit=50, offset=0*)

Get Spotify catalog information about an album's tracks

**Parameters:**

- *album\_id* - the album ID, URI or URL
- *limit* - the number of items to return
- *offset* - the index of the first item to return

**albums** (*albums*)

returns a list of albums given the album IDs, URIs, or URLs

**Parameters:**

- *albums* - a list of album IDs, URIs or URLs

**artist** (*artist\_id*)

returns a single artist given the artist's ID, URI or URL

**Parameters:**

- *artist\_id* - an artist ID, URI or URL

**artist\_albums** (*artist\_id, album\_type=None, country=None, limit=20, offset=0*)

Get Spotify catalog information about an artist's albums

**Parameters:**

- *artist\_id* - the artist ID, URI or URL
- *album\_type* - 'album', 'single', 'appears\_on', 'compilation'
- *country* - limit the response to one particular country.
- *limit* - the number of albums to return
- *offset* - the index of the first album to return

**artist\_related\_artists** (*artist\_id*)

Get Spotify catalog information about artists similar to an identified artist. Similarity is based on analysis of the Spotify community's listening history.

**Parameters:**

- *artist\_id* - the artist ID, URI or URL

**artist\_top\_tracks** (*artist\_id, country='US'*)

Get Spotify catalog information about an artist's top 10 tracks by country.

**Parameters:**

- *artist\_id* - the artist ID, URI or URL
- *country* - limit the response to one particular country.

**artists** (*artists*)

returns a list of artists given the artist IDs, URIs, or URLs

**Parameters:**

- **artists** - a list of artist IDs, URIs or URLs

**audio\_analysis** (*track\_id*)

Get audio analysis for a track based upon its Spotify ID Parameters:

- **track\_id** - a track URI, URL or ID

**audio\_features** (*tracks=[]*)

Get audio features for one or multiple tracks based upon their Spotify IDs Parameters:

- **tracks** - a list of track URIs, URLs or IDs, maximum: 50 ids

**categories** (*country=None, locale=None, limit=20, offset=0*)

Get a list of new album releases featured in Spotify

**Parameters:**

- **country** - An ISO 3166-1 alpha-2 country code.
- **locale** - The desired language, consisting of an ISO 639 language code and an ISO 3166-1 alpha-2 country code, joined by an underscore.
- **limit** - The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 50
- **offset** - The index of the first item to return. Default: 0 (the first object). Use with limit to get the next set of items.

**category\_playlists** (*category\_id=None, country=None, limit=20, offset=0*)

Get a list of new album releases featured in Spotify

**Parameters:**

- **category\_id** - The Spotify category ID for the category.
- **country** - An ISO 3166-1 alpha-2 country code.
- **limit** - The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 50
- **offset** - The index of the first item to return. Default: 0 (the first object). Use with limit to get the next set of items.

**current\_playback** (*market=None*)

Get information about user's current playback.

**Parameters:**

- **market** - an ISO 3166-1 alpha-2 country code.

**current\_user** ()

Get detailed profile information about the current user. An alias for the 'me' method.

**current\_user\_followed\_artists** (*limit=20, after=None*)

Gets a list of the artists followed by the current authorized user

**Parameters:**

- **limit** - the number of artists to return
- **after** - the last artist ID retrieved from the previous request

**current\_user\_playing\_track** ()

Get information about the current users currently playing track.

**current\_user\_playlists** (*limit=50, offset=0*)

Get current user playlists without required getting his profile Parameters:

- limit - the number of items to return
- offset - the index of the first item to return

**current\_user\_recently\_played** (*limit=50*)

Get the current user's recently played tracks

**Parameters:**

- limit - the number of entities to return

**current\_user\_saved\_albums** (*limit=20, offset=0*)

Gets a list of the albums saved in the current authorized user's "Your Music" library

**Parameters:**

- limit - the number of albums to return
- offset - the index of the first album to return

**current\_user\_saved\_albums\_add** (*albums=[]*)

Add one or more albums to the current user's "Your Music" library. Parameters:

- albums - a list of album URIs, URLs or IDs

**current\_user\_saved\_albums\_contains** (*albums=[]*)

Check if one or more albums is already saved in the current Spotify user's "Your Music" library.

**Parameters:**

- albums - a list of album URIs, URLs or IDs

**current\_user\_saved\_albums\_delete** (*albums=[]*)

Remove one or more albums from the current user's "Your Music" library.

**Parameters:**

- albums - a list of album URIs, URLs or IDs

**current\_user\_saved\_tracks** (*limit=20, offset=0*)

Gets a list of the tracks saved in the current authorized user's "Your Music" library

**Parameters:**

- limit - the number of tracks to return
- offset - the index of the first track to return

**current\_user\_saved\_tracks\_add** (*tracks=None*)

Add one or more tracks to the current user's "Your Music" library.

**Parameters:**

- tracks - a list of track URIs, URLs or IDs

**current\_user\_saved\_tracks\_contains** (*tracks=None*)

Check if one or more tracks is already saved in the current Spotify user's "Your Music" library.

**Parameters:**

- tracks - a list of track URIs, URLs or IDs

**current\_user\_saved\_tracks\_delete** (*tracks=None*)

Remove one or more tracks from the current user's "Your Music" library.

**Parameters:**

- tracks - a list of track URIs, URLs or IDs

**current\_user\_top\_artists** (*limit=20, offset=0, time\_range='medium\_term'*)

Get the current user's top artists

**Parameters:**

- limit - the number of entities to return
- offset - the index of the first entity to return
- time\_range - Over what time frame are the affinities computed Valid-values: short\_term, medium\_term, long\_term

**current\_user\_top\_tracks** (*limit=20, offset=0, time\_range='medium\_term'*)

Get the current user's top tracks

**Parameters:**

- limit - the number of entities to return
- offset - the index of the first entity to return
- time\_range - Over what time frame are the affinities computed Valid-values: short\_term, medium\_term, long\_term

**currently\_playing** (*market=None*)

Get user's currently playing track.

**Parameters:**

- market - an ISO 3166-1 alpha-2 country code.

**devices** ()

Get a list of user's available devices.

**featured\_playlists** (*locale=None, country=None, timestamp=None, limit=20, offset=0*)

Get a list of Spotify featured playlists

**Parameters:**

- locale - The desired language, consisting of a lowercase ISO 639 language code and an uppercase ISO 3166-1 alpha-2 country code, joined by an underscore.
- country - An ISO 3166-1 alpha-2 country code.
- timestamp - A timestamp in ISO 8601 format: yyyy-MM-ddTHH:mm:ss. Use this parameter to specify the user's local time to get results tailored for that specific date and time in the day
- limit - The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 50
- offset - The index of the first item to return. Default: 0 (the first object). Use with limit to get the next set of items.

**max\_get\_retries** = 10**me** ()

Get detailed profile information about the current user. An alias for the 'current\_user' method.

**new\_releases** (*country=None, limit=20, offset=0*)

Get a list of new album releases featured in Spotify

**Parameters:**

- country - An ISO 3166-1 alpha-2 country code.

- limit - The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 50
- offset - The index of the first item to return. Default: 0 (the first object). Use with limit to get the next set of items.

**next** (*result*)

returns the next result given a paged result

**Parameters:**

- result - a previously returned paged result

**next\_track** (*device\_id=None*)

Skip user's playback to next track.

**Parameters:**

- device\_id - device target for playback

**pause\_playback** (*device\_id=None*)

Pause user's playback.

**Parameters:**

- device\_id - device target for playback

**playlist** (*playlist\_id, fields=None, market=None*)

Gets playlist by id.

**Parameters:**

- playlist - the id of the playlist
- fields - which fields to return
- market - An ISO 3166-1 alpha-2 country code or the string from\_token.

**playlist\_tracks** (*playlist\_id, fields=None, limit=100, offset=0, market=None*)

Get full details of the tracks of a playlist.

**Parameters:**

- playlist\_id - the id of the playlist
- fields - which fields to return
- limit - the maximum number of tracks to return
- offset - the index of the first track to return
- market - an ISO 3166-1 alpha-2 country code.

**playlist\_upload\_cover\_image** (*playlist\_id, image\_b64*)

Replace the image used to represent a specific playlist

**Parameters:**

- playlist\_id - the id of the playlist
- image\_b64 - image data as a Base64 encoded JPEG image string (maximum payload size is 256 KB)

**previous** (*result*)

returns the previous result given a paged result

**Parameters:**

- result - a previously returned paged result

**previous\_track** (*device\_id=None*)

Skip user's playback to previous track.

**Parameters:**

- *device\_id* - device target for playback

**recommendation\_genre\_seeds** ()

Get a list of genres available for the recommendations function.

**recommendations** (*seed\_artists=None, seed\_genres=None, seed\_tracks=None, limit=20, country=None, \*\*kwargs*)

Get a list of recommended tracks for one to five seeds.

**Parameters:**

- *seed\_artists* - a list of artist IDs, URIs or URLs
- *seed\_tracks* - a list of track IDs, URIs or URLs
- **seed\_genres** - a list of genre names. Available genres for recommendations can be found by calling *recommendation\_genre\_seeds*
- **country** - An ISO 3166-1 alpha-2 country code. If provided, all results will be playable in this country.
- **limit** - The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 100
- **min/max/target\_<attribute>** - For the tuneable track attributes listed in the documentation, these values provide filters and targeting on results.

**repeat** (*state, device\_id=None*)

Set repeat mode for playback.

**Parameters:**

- *state* - *track*, *context*, or *off*
- *device\_id* - device target for playback

**search** (*q, limit=10, offset=0, type='track', market=None*)

searches for an item

**Parameters:**

- *q* - the search query
- *limit* - the number of items to return
- *offset* - the index of the first item to return
- **type** - the type of item to return. One of 'artist', 'album', 'track' or 'playlist'
- **market** - An ISO 3166-1 alpha-2 country code or the string *from\_token*.

**seek\_track** (*position\_ms, device\_id=None*)

Seek to position in current track.

**Parameters:**

- *position\_ms* - position in milliseconds to seek to
- *device\_id* - device target for playback

**shuffle** (*state, device\_id=None*)

Toggle playback shuffling.

**Parameters:**

- state - true or false
- device\_id - device target for playback

**start\_playback** (*device\_id=None, context\_uri=None, uris=None, offset=None*)

Start or resume user's playback.

Provide a *context\_uri* to start playback or a album, artist, or playlist.

Provide a *uris* list to start playback of one or more tracks.

Provide *offset* as {"position": <int>} or {"uri": "<track uri>"} to start playback at a particular offset.

**Parameters:**

- device\_id - device target for playback
- context\_uri - spotify context uri to play
- uris - spotify track uris
- offset - offset into context by index or track

**trace = False**

**trace\_out = False**

**track** (*track\_id*)

returns a single track given the track's ID, URI or URL

**Parameters:**

- track\_id - a spotify URI, URL or ID

**tracks** (*tracks, market=None*)

returns a list of tracks given a list of track IDs, URIs, or URLs

**Parameters:**

- tracks - a list of spotify URIs, URLs or IDs
- market - an ISO 3166-1 alpha-2 country code.

**transfer\_playback** (*device\_id, force\_play=True*)

Transfer playback to another device. Note that the API accepts a list of device ids, but only actually supports one.

**Parameters:**

- device\_id - transfer playback to this device
- **force\_play - true: after transfer, play. false:** keep current state.

**user** (*user*)

Gets basic profile information about a Spotify User

**Parameters:**

- user - the id of the user

**user\_follow\_artists** (*ids=[]*)

Follow one or more artists Parameters:

- ids - a list of artist IDs

**user\_follow\_users** (*ids=[]*)

Follow one or more users Parameters:

- ids - a list of user IDs



**user\_playlist** (*user, playlist\_id=None, fields=None, market=None*)

**user\_playlist\_add\_tracks** (*user, playlist\_id, tracks, position=None*)  
Adds tracks to a playlist

**Parameters:**

- *user* - the id of the user
- *playlist\_id* - the id of the playlist
- *tracks* - a list of track URIs, URLs or IDs
- *position* - the position to add the tracks

**user\_playlist\_change\_details** (*user, playlist\_id, name=None, public=None, collaborative=None, description=None*)  
Changes a playlist's name and/or public/private state

**Parameters:**

- *user* - the id of the user
- *playlist\_id* - the id of the playlist
- *name* - optional name of the playlist
- *public* - optional is the playlist public
- *collaborative* - optional is the playlist collaborative
- *description* - optional description of the playlist

**user\_playlist\_create** (*user, name, public=True, description=""*)  
Creates a playlist for a user

**Parameters:**

- *user* - the id of the user
- *name* - the name of the playlist
- *public* - is the created playlist public
- *description* - the description of the playlist

**user\_playlist\_follow\_playlist** (*playlist\_owner\_id, playlist\_id*)  
Add the current authenticated user as a follower of a playlist.

**Parameters:**

- *playlist\_owner\_id* - the user id of the playlist owner
- *playlist\_id* - the id of the playlist

**user\_playlist\_is\_following** (*playlist\_owner\_id, playlist\_id, user\_ids*)  
Check to see if the given users are following the given playlist

**Parameters:**

- *playlist\_owner\_id* - the user id of the playlist owner
- *playlist\_id* - the id of the playlist
- **user\_ids** - the ids of the users that you want to check to see if they follow the playlist. Maximum: 5 ids.

**user\_playlist\_remove\_all\_occurrences\_of\_tracks** (*user, playlist\_id, tracks, snapshot\_id=None*)

Removes all occurrences of the given tracks from the given playlist

**Parameters:**

- user - the id of the user
- playlist\_id - the id of the playlist
- tracks - the list of track ids to remove from the playlist
- snapshot\_id - optional id of the playlist snapshot

**user\_playlist\_remove\_specific\_occurrences\_of\_tracks** (*user, playlist\_id, tracks, snapshot\_id=None*)

Removes all occurrences of the given tracks from the given playlist

**Parameters:**

- user - the id of the user
- playlist\_id - the id of the playlist
- **tracks - an array of objects containing Spotify URIs of the** tracks to remove with their current positions in the playlist. For example:

```
[ { "uri": "4iV5W9uYEdYUVa79Axb7Rh", "positions": [2] }, {
  "uri": "1301WleyT98MSxVHPZCA6M", "positions": [7] } ]
```

- snapshot\_id - optional id of the playlist snapshot

**user\_playlist\_reorder\_tracks** (*user, playlist\_id, range\_start, insert\_before, range\_length=1, snapshot\_id=None*)

Reorder tracks in a playlist

**Parameters:**

- user - the id of the user
- playlist\_id - the id of the playlist
- range\_start - the position of the first track to be reordered
- **range\_length - optional the number of tracks to be reordered** (default: 1)
- **insert\_before - the position where the tracks should be** inserted
- snapshot\_id - optional playlist's snapshot ID

**user\_playlist\_replace\_tracks** (*user, playlist\_id, tracks*)

Replace all tracks in a playlist

**Parameters:**

- user - the id of the user
- playlist\_id - the id of the playlist
- tracks - the list of track ids to add to the playlist

**user\_playlist\_tracks** (*user=None, playlist\_id=None, fields=None, limit=100, offset=0, market=None*)

**user\_playlist\_unfollow** (*user, playlist\_id*)

Unfollows (deletes) a playlist for a user

**Parameters:**

- user - the id of the user
- name - the name of the playlist

**user\_playlists** (*user, limit=50, offset=0*)

Gets playlists of a user

**Parameters:**

- user - the id of the user
- limit - the number of items to return
- offset - the index of the first item to return

**user\_unfollow\_artists** (*ids=[]*)

Unfollow one or more artists Parameters:

- ids - a list of artist IDs

**user\_unfollow\_users** (*ids=[]*)

Unfollow one or more users Parameters:

- ids - a list of user IDs

**volume** (*volume\_percent, device\_id=None*)

Set playback volume.

**Parameters:**

- volume\_percent - volume between 0 and 100
- device\_id - device target for playback

**exception** `spotipy.client.SpotifyException` (*http\_status, code, msg, headers=None*)

Bases: `exceptions.Exception`

**\_\_init\_\_** (*http\_status, code, msg, headers=None*)

`x.__init__(...)` initializes x; see `help(type(x))` for signature



`spotify.oauth2.is_token_expired(token_info)`

**class** `spotify.oauth2.SpotifyClientCredentials` (*client\_id=None, client\_secret=None, proxies=None*)

Bases: `object`

**OAUTH\_TOKEN\_URL** = `'https://accounts.spotify.com/api/token'`

**\_\_init\_\_** (*client\_id=None, client\_secret=None, proxies=None*)

You can either provide a `client_id` and `client_secret` to the constructor or set `SPOTIPY_CLIENT_ID` and `SPOTIPY_CLIENT_SECRET` environment variables

**get\_access\_token** ()

If a valid access token is in memory, returns it Else fetches a new token and returns it

**is\_token\_expired** (*token\_info*)

**class** `spotify.oauth2.SpotifyOAuth` (*client\_id, client\_secret, redirect\_uri, state=None, scope=None, cache\_path=None, proxies=None*)

Bases: `object`

Implements Authorization Code Flow for Spotify's OAuth implementation.

**OAUTH\_AUTHORIZE\_URL** = `'https://accounts.spotify.com/authorize'`

**OAUTH\_TOKEN\_URL** = `'https://accounts.spotify.com/api/token'`

**\_\_init\_\_** (*client\_id, client\_secret, redirect\_uri, state=None, scope=None, cache\_path=None, proxies=None*)

Creates a `SpotifyOAuth` object

**Parameters:**

- `client_id` - the client id of your app
- `client_secret` - the client secret of your app
- `redirect_uri` - the redirect URI of your app
- `state` - security state

- scope - the desired scope of the request
- cache\_path - path to location to save tokens

**get\_access\_token** (*code*)

Gets the access token for the app given the code

**Parameters:**

- code - the response code

**get\_authorize\_url** (*state=None, show\_dialog=False*)

Gets the URL to use to authorize this app

**get\_cached\_token** ()

Gets a cached auth token

**is\_token\_expired** (*token\_info*)

**parse\_response\_code** (*url*)

Parse the response code in the given response url

**Parameters:**

- url - the response url

**refresh\_access\_token** (*refresh\_token*)

**exception** `spotify.oauth2.SpotifyOauthError`

Bases: `exceptions.Exception`

## 11.1 util Module

Shows a user's playlists (need to be authenticated via oauth)

`spotify.util.prompt_for_user_token` (*username, scope=None, client\_id=None, client\_secret=None, redirect\_uri=None, cache\_path=None*)

prompts the user to login if necessary and returns the user token suitable for use with the `spotify.Spotify` constructor

**Parameters:**

- username - the Spotify username
- scope - the desired scope of the request
- client\_id - the client id of your app
- client\_secret - the client secret of your app
- redirect\_uri - the redirect URI of your app
- cache\_path - path to location to save tokens

## CHAPTER 12

---

### Support

---

You can ask questions about Spotipy on Stack Overflow. Don't forget to add the *Spotipy* tag, and any other relevant tags as well, before posting.

<http://stackoverflow.com/questions/ask>

If you think you've found a bug, let us know at [Spotify Issues](#)





# CHAPTER 13

---

## Contribute

---

Spotipy authored by Paul Lamere (plamere) with contributions by:

- Daniel Beaudry // danbeaudry
- Faruk Emre Sahin // fsahin
- George // rogueleaderr
- Henry Greville // sethaurus
- Hugo // hugovk
- José Manuel Pérez // JMPerez
- Lucas Nunno // lnunno
- Lynn Root // econchick
- Matt Dennewitz // mattdennewitz
- Matthew Duck // mattduck
- Michael Thelin // thelinmichael
- Ryan Choi // ryankicks
- Simon Metson // drsm79
- Steve Winton // swinton
- Tim Balzer // timbalzer
- corycorycory // corycorycory
- Nathan Coleman // nathancoleman
- Michael Birtwell // mbirtwell
- Harrison Hayes // Harrison97
- Stephane Bruckert // stephanebruckert
- Ritiek Malhotra // ritiek



## CHAPTER 14

---

### License

---

<https://github.com/plamere/spotipy/blob/master/LICENSE.md>



# CHAPTER 15

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



**S**

`spotify.client`, 21  
`spotify.oauth2`, 33  
`spotify.util`, 34





## Symbols

`__init__()` (*spotipy.client.Spotify method*), 21

`__init__()` (*spotipy.client.SpotifyException method*), 31

`__init__()` (*spotipy.oauth2.SpotifyClientCredentials method*), 33

`__init__()` (*spotipy.oauth2.SpotifyOAuth method*), 33

## A

`album()` (*spotipy.client.Spotify method*), 22

`album_tracks()` (*spotipy.client.Spotify method*), 22

`albums()` (*spotipy.client.Spotify method*), 22

`artist()` (*spotipy.client.Spotify method*), 22

`artist_albums()` (*spotipy.client.Spotify method*), 22

`artist_related_artists()`  
(*spotipy.client.Spotify method*), 22

`artist_top_tracks()` (*spotipy.client.Spotify method*), 22

`artists()` (*spotipy.client.Spotify method*), 22

`audio_analysis()` (*spotipy.client.Spotify method*), 23

`audio_features()` (*spotipy.client.Spotify method*), 23

## C

`categories()` (*spotipy.client.Spotify method*), 23

`category_playlists()` (*spotipy.client.Spotify method*), 23

`current_playback()` (*spotipy.client.Spotify method*), 23

`current_user()` (*spotipy.client.Spotify method*), 23

`current_user_followed_artists()`  
(*spotipy.client.Spotify method*), 23

`current_user_playing_track()`  
(*spotipy.client.Spotify method*), 23

`current_user_playlists()`  
(*spotipy.client.Spotify method*), 23

`current_user_recently_played()`  
(*spotipy.client.Spotify method*), 24

`current_user_saved_albums()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_albums_add()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_albums_contains()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_albums_delete()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_tracks()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_tracks_add()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_tracks_contains()`

(*spotipy.client.Spotify method*), 24

`current_user_saved_tracks_delete()`

(*spotipy.client.Spotify method*), 24

`current_user_top_artists()`

(*spotipy.client.Spotify method*), 25

`current_user_top_tracks()`

(*spotipy.client.Spotify method*), 25

`currently_playing()` (*spotipy.client.Spotify method*), 25

## D

`devices()` (*spotipy.client.Spotify method*), 25

## F

`featured_playlists()` (*spotipy.client.Spotify method*), 25

## G

`get_access_token()`

(*spotipy.oauth2.SpotifyClientCredentials method*), 33

`get_access_token()` (*spotipy.oauth2.SpotifyOAuth method*), 34

`get_authorize_url()`

(*spotipy.oauth2.SpotifyOAuth method*), 34

`get_cached_token()` (*spotipy.oauth2.SpotifyOAuth method*), 34

## I

`is_token_expired()` (in module `spotipy.oauth2`), 33

`is_token_expired()` (`spotipy.oauth2.SpotifyClientCredentials` method), 33

`is_token_expired()` (`spotipy.oauth2.SpotifyOAuth` method), 34

## M

`max_get_retries` (`spotipy.client.Spotify` attribute), 25

`me()` (`spotipy.client.Spotify` method), 25

## N

`new_releases()` (`spotipy.client.Spotify` method), 25

`next()` (`spotipy.client.Spotify` method), 26

`next_track()` (`spotipy.client.Spotify` method), 26

## O

`OAUTH_AUTHORIZE_URL` (`spotipy.oauth2.SpotifyOAuth` attribute), 33

`OAUTH_TOKEN_URL` (`spotipy.oauth2.SpotifyClientCredentials` attribute), 33

`OAUTH_TOKEN_URL` (`spotipy.oauth2.SpotifyOAuth` attribute), 33

## P

`parse_response_code()` (`spotipy.oauth2.SpotifyOAuth` method), 34

`pause_playback()` (`spotipy.client.Spotify` method), 26

`playlist()` (`spotipy.client.Spotify` method), 26

`playlist_tracks()` (`spotipy.client.Spotify` method), 26

`playlist_upload_cover_image()` (`spotipy.client.Spotify` method), 26

`previous()` (`spotipy.client.Spotify` method), 26

`previous_track()` (`spotipy.client.Spotify` method), 26

`prompt_for_user_token()` (in module `spotipy.util`), 34

## R

`recommendation_genre_seeds()` (`spotipy.client.Spotify` method), 27

`recommendations()` (`spotipy.client.Spotify` method), 27

`refresh_access_token()` (`spotipy.oauth2.SpotifyOAuth` method), 34

`repeat()` (`spotipy.client.Spotify` method), 27

## S

`search()` (`spotipy.client.Spotify` method), 27

`seek_track()` (`spotipy.client.Spotify` method), 27

`shuffle()` (`spotipy.client.Spotify` method), 27

`Spotify` (class in `spotipy.client`), 21

`SpotifyClientCredentials` (class in `spotipy.oauth2`), 33

`SpotifyException`, 31

`SpotifyOAuth` (class in `spotipy.oauth2`), 33

`SpotifyOAuthError`, 34

`spotipy.client` (module), 21

`spotipy.oauth2` (module), 33

`spotipy.util` (module), 34

`start_playback()` (`spotipy.client.Spotify` method), 28

## T

`trace` (`spotipy.client.Spotify` attribute), 28

`trace_out` (`spotipy.client.Spotify` attribute), 28

`track()` (`spotipy.client.Spotify` method), 28

`tracks()` (`spotipy.client.Spotify` method), 28

`transfer_playback()` (`spotipy.client.Spotify` method), 28

## U

`user()` (`spotipy.client.Spotify` method), 28

`user_follow_artists()` (`spotipy.client.Spotify` method), 28

`user_follow_users()` (`spotipy.client.Spotify` method), 28

`user_playlist()` (`spotipy.client.Spotify` method), 28

`user_playlist_add_tracks()` (`spotipy.client.Spotify` method), 29

`user_playlist_change_details()` (`spotipy.client.Spotify` method), 29

`user_playlist_create()` (`spotipy.client.Spotify` method), 29

`user_playlist_follow_playlist()` (`spotipy.client.Spotify` method), 29

`user_playlist_is_following()` (`spotipy.client.Spotify` method), 29

`user_playlist_remove_all_occurrences_of_tracks()` (`spotipy.client.Spotify` method), 29

`user_playlist_remove_specific_occurrences_of_tracks()` (`spotipy.client.Spotify` method), 30

`user_playlist_reorder_tracks()` (`spotipy.client.Spotify` method), 30

`user_playlist_replace_tracks()` (`spotipy.client.Spotify` method), 30

`user_playlist_tracks()` (`spotipy.client.Spotify` method), 30

`user_playlist_unfollow()` (`spotipy.client.Spotify` method), 30

`user_playlists()` (*spotipy.client.Spotify method*),  
31

`user_unfollow_artists()` (*spotipy.client.Spotify  
method*), 31

`user_unfollow_users()` (*spotipy.client.Spotify  
method*), 31

## V

`volume()` (*spotipy.client.Spotify method*), 31