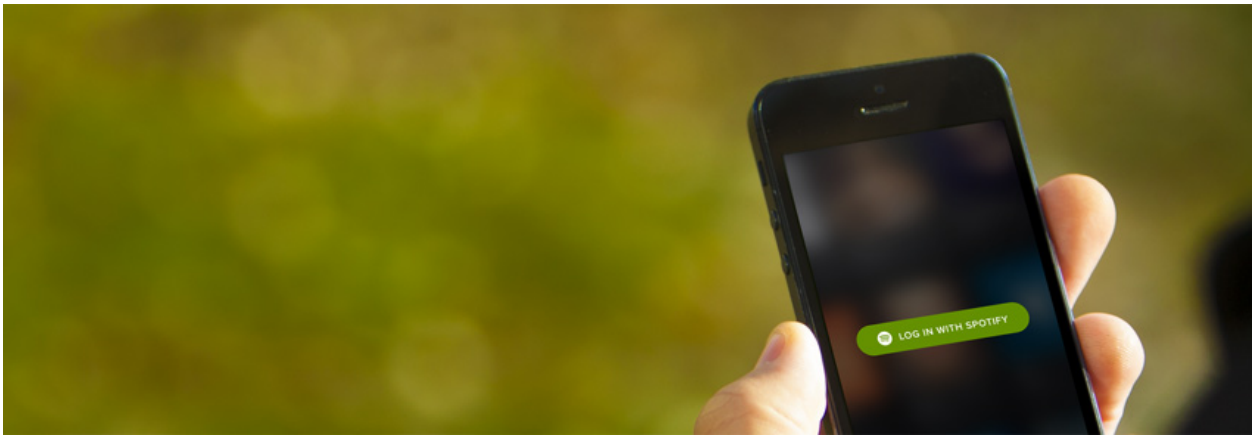
spotipy Documentation

Release 2.0

Paul Lamere

Apr 11, 2020

Contents



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.

Assuming you set the `SPOTIPY_CLIENT_ID` and `SPOTIPY_CLIENT_SECRET` environment variables, here's a quick example of using *Spotipy* to list the names of all the albums released by the artist 'Birdy':

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials

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

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
from spotipy.oauth2 import SpotifyClientCredentials

lz_uri = 'spotify:artist:36QJpDe2go2KgaRleHCDTp'

spotify = spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials())
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
from spotipy.oauth2 import SpotifyClientCredentials

```

(continues on next page)

(continued from previous page)

```
spotify = spotipy.Spotify(client_credentials_manager=SpotifyClientCredentials())

if len(sys.argv) > 1:
    name = ' '.join(sys.argv[1:])
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 you can get the source from github at https://github.com/plamere/spotipy

CHAPTER 3

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 paragragh.

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.

CHAPTER 4

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 succesfully 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'
```

(continues on next page)

(continued from previous page)

```
if len(sys.argv) > 1:
    username = sys.argv[1]
else:
    print("Usage: %s username" % (sys.argv[0],))
    sys.exit()

token = util.prompt_for_user_token(username, scope)

if token:
    sp = spotipy.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)
```

CHAPTER 5

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.

As opposed to the Authorization Code Flow, you will not need to set `SPOTIPY_REDIRECT_URI`, which means you will never be redirected to the sign in page in your browser.

> export SPOTIPY_CLIENT_ID='your-spotify-client-id' export SPOTIPY_CLIENT_SECRET='your-spotify-client-secret'

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
```

CHAPTER 6

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

CHAPTER 7

Examples

Here are a few more examples of using *Spotipy*, this time using the Authorization Code Flow to access your personal Spotify account data.

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
```

(continues on next page)

(continued from previous page)

```
import spotipy
import spotipy.util as util

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

CHAPTER 10

client Module

A simple and thin Python library for the Spotify Web API

class spotipy.client.**Spotify**(*auth=None*, *requests_session=True*, *client_credentials_manager=None*, *oauth_manager=None*, *auth_manager=None*, *proxies=None*, *requests_timeout=None*)

Bases: object

Example usage:

```
import spotipy

urn = 'spotify:artist:3jOstUTkEu2JkjvRdBA5Gu'
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*, *oauth_manager=None*, *auth_manager=None*, *proxies=None*, *requests_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

- **oauth_manager** – SpotifyOAuth object
- **auth_manager** – SpotifyOauth object or SpotifyClientCredentials object
- **proxies** – Definition of proxies (optional). See Requests doc https://2.python-requests.org/en/master/user/advanced/#proxies
- **requests_timeout** – Tell Requests to stop waiting for a response after a given number of seconds

add_to_queue(*uri*, *device_id=None*)
Adds a song to the end of a user's queue

If device A is currently playing music and you try to add to the queue and pass in the id for device B, you will get a 'Player command failed: Restriction violated' error I therefore reccomend leaving device_id as None so that the active device is targeted

Parameters

- **uri** – song uri, id, or url
- **device_id** – the id of a Spotify device. If None, then the active device is used.

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

auth_manager

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*, *after=None*, *before=None*)
Get the current user's recently played tracks

Parameters:

- limit - the number of entities to return
- **after - unix timestamp in milliseconds. Returns all items** after (but not including) this cursor position. Cannot be used if before is specified.
- **before - unix timestamp in milliseconds. Returns all items** before (but not including) this cursor position. Cannot be used if after is specified

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_cover_image(*playlist_id*)
Get cover of a playlist.

Parameters:

- playlist_id - the id of the playlist

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. (at least one of *seed_artists*, *seed_tracks* and *seed_genres* are needed)

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 (see how to write a query in the** official documentation https://developer.spotify.com/documentation/web-api/reference/search/search/) # noqa
- limit - the number of items to return (min = 1, default = 10, max = 50)
- 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*, *position_ms=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
- **position_ms - (optional) indicates from what position to start playback.** Must be a positive number. Passing in a position that is greater than the length of the track will cause the player to start playing the next song.

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. Maximum: 50 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 usr

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 usr
- 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

CHAPTER 11

oauth2 Module

spotipy.oauth2.**is_token_expired**(*token_info*)

class spotipy.oauth2.**SpotifyClientCredentials**(*client_id=None*, *client_secret=None*, *proxies=None*, *requests_timeout=None*)

Bases: spotipy.oauth2.SpotifyAuthBase

OAUTH_TOKEN_URL = 'https://accounts.spotify.com/api/token'

__init__(*client_id=None*, *client_secret=None*, *proxies=None*, *requests_timeout=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(*as_dict=True*)

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

Parameters: - as_dict - a boolean indicating if returning the access token

as a token_info dictionary, otherwise it will be returned as a string.

is_token_expired(*token_info*)

class spotipy.oauth2.**SpotifyOAuth**(*client_id=None*, *client_secret=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *proxies=None*, *show_dialog=False*, *requests_timeout=None*)

Bases: spotipy.oauth2.SpotifyAuthBase

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=None*, *client_secret=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *proxies=None*, *show_dialog=False*, *requests_timeout=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
- **requests_timeout - tell Requests to stop waiting for a response** after a given number of seconds
- username - username of current client

get_access_token(*code=None*, *as_dict=True*, *check_cache=True*)
Gets the access token for the app given the code

Parameters:

- code - the response code
- **as_dict - a boolean indicating if returning the access token** as a token_info dictionary, otherwise it will be returned as a string.

get_auth_response()

get_authorization_code(*response=None*)

get_authorize_url(*state=None*)
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 spotipy.oauth2.**SpotifyOauthError**
Bases: exceptions.Exception

11.1 util Module

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

spotipy.util.**prompt_for_user_token**(*username*, *scope=None*, *client_id=None*, *client_secret=None*, *redirect_uri=None*, *cache_path=None*, *oauth_manager=None*, *show_dialog=False*)
prompts the user to login if necessary and returns the user token suitable for use with the spotipy.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
- oauth_manager - Oauth manager object.

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

Python Module Index

S

Index

Symbols

A

C

D

F

G

V