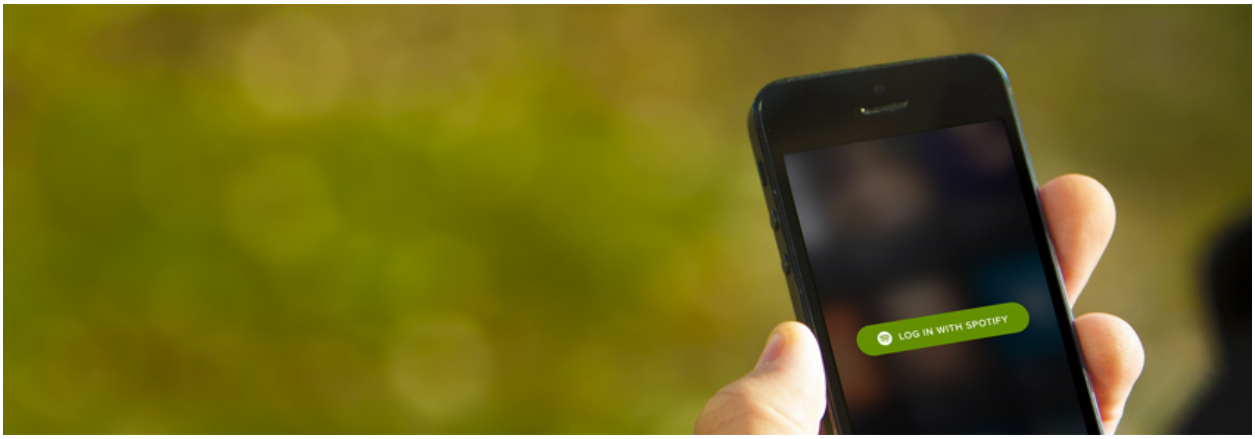
spotipy Documentation

Release 2.0

Paul Lamere

Oct 29, 2022

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 is a video explaining how to do so), 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(auth_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 at My Dashboard to get the credentials necessary to make authorized calls (a *client id* and *client secret*).

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.

  Note: Requires you to add a redirect URI to your application at My Dashboard. See *Redirect URI* for more details.

- 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.

4.1 Quick start

To support the **Client Authorization Code Flow** *Spotipy* provides a class SpotifyOAuth that can be used to authenticate requests like so:

```
import spotipy
from spotipy.oauth2 import SpotifyOAuth

scope = "user-library-read"

sp = spotipy.Spotify(auth_manager=SpotifyOAuth(scope=scope))

results = sp.current_user_saved_tracks()
for idx, item in enumerate(results['items']):
    track = item['track']
    print(idx, track['artists'][0]['name'], " - ", track['name'])
```

or if you are reluctant to immortalize your app credentials in your source code, you can set environment variables like so (use `$env:"credentials"` instead of `export` on Windows):

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

4.2 Scopes

See Using Scopes for information about scopes.

4.3 Redirect URI

The **Authorization Code Flow** needs you to add a **redirect URI** to your application at My Dashboard (navigate to your application and then *[Edit Settings]*).

The `redirect_uri` argument or `SPOTIPY_REDIRECT_URI` environment variable must match the redirect URI added to your application in your Dashboard. The redirect URI can be any valid URI (it does not need to be accessible) such as `http://example.com`, `http://localhost` or `http://127.0.0.1:9090`.

Note: If you choose an *http*-scheme URL, and it's for *localhost* or *127.0.0.1*, **AND** it specifies a port, then spotispy will instantiate

> a server on the indicated response to receive the access token from the response at the end of the oauth flow [see the code](https://github.com/plamere/spotipy/blob/master/spotipy/oauth2.py#L483-L490).

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

auth_manager = SpotifyClientCredentials()
sp = spotipy.Spotify(auth_manager=auth_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

Customized token caching

Tokens are refreshed automatically and stored by default in the project main folder. As this might not suit everyone's needs, spotipy provides a way to create customized cache handlers.

https://github.com/plamere/spotipy/blob/master/spotipy/cache_handler.py

The custom cache handler would need to be a class that inherits from the base cache handler `CacheHandler`. The default cache handler `CacheFileHandler` is a good example. An instance of that new class can then be passed as a parameter when creating `SpotifyOAuth`, `SpotifyPKCE` or `SpotifyImplicitGrant`. The following handlers are available and defined in the URL above.

- `CacheFileHandler`
- `MemoryCacheHandler`
- `DjangoSessionCacheHandler`
- `FlaskSessionCacheHandler`
- `RedisCacheHandler`

Feel free to contribute new cache handlers to the repo.

CHAPTER 8

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=5*, *status_forcelist=None*, *retries=3*, *status_retries=3*, *backoff_factor=0.3*, *language=None*)

Bases: object

Example usage:

```
import spotipy

urn = 'spotify:artist:3jOstUTkEu2JkjvRdBA5Gu'
sp = spotipy.Spotify()

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=5*, *status_forcelist=None*, *retries=3*, *status_retries=3*, *backoff_factor=0.3*, *language=None*)

Creates a Spotify API client.

Parameters

- **auth** – An access 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, SpotifyClientCredentials, or SpotifyImplicitGrant 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
- **status_forcelist** – Tell requests what type of status codes retries should occur on
- **retries** – Total number of retries to allow
- **status_retries** – Number of times to retry on bad status codes
- **backoff_factor** – A backoff factor to apply between attempts after the second try See urllib3 https://urllib3.readthedocs.io/en/latest/reference/urllib3.util.html
- **language** – The language parameter advertises what language the user prefers to see. See ISO-639-1 language code: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

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*, *market=None*)

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

Parameters:

- album_id - the album ID, URI or URL
- market - an ISO 3166-1 alpha-2 country code

album_tracks(*album_id*, *limit=50*, *offset=0*, *market=None*)

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

albums(*albums*, *market=None*)

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

Parameters:

- albums - a list of album IDs, URIs or URLs
- market - an ISO 3166-1 alpha-2 country code

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: 100 ids

auth_manager

available_markets()
Get the list of markets where Spotify is available. Returns a list of the countries in which Spotify is available, identified by their ISO 3166-1 alpha-2 country code with additional country codes for special territories.

categories(*country=None*, *locale=None*, *limit=20*, *offset=0*)
Get a list of categories

Parameters:

- country - An ISO 3166-1 alpha-2 country code.
- locale - The desired language, consisting of an ISO 639-1 alpha-2 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 (*category_id*, *country=None*, *locale=None*)
Get info about a category

Parameters:

- category_id - The Spotify category ID for the category.
- country - An ISO 3166-1 alpha-2 country code.
- locale - The desired language, consisting of an ISO 639-1 alpha-2 language code and an ISO 3166-1 alpha-2 country code, joined by an underscore.

category_playlists (*category_id=None*, *country=None*, *limit=20*, *offset=0*)
Get a list of playlists for a specific Spotify category

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.

country_codes = ['AD', 'AR', 'AU', 'AT', 'BE', 'BO', 'BR', 'BG', 'CA', 'CL', 'CO', 'CR

current_playback (*market=None*, *additional_types=None*)
Get information about user's current playback.

Parameters:

- market - an ISO 3166-1 alpha-2 country code.
- additional_types - *episode* to get podcast track information

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

current_user_follow_playlist (*playlist_id*)
Add the current authenticated user as a follower of a playlist.

Parameters:

- playlist_id - the id of the playlist

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_following_artists(*ids=None*)
Check if the current user is following certain artists

Returns list of booleans respective to ids

Parameters:

- ids - a list of artist URIs, URLs or IDs

current_user_following_users(*ids=None*)
Check if the current user is following certain users

Returns list of booleans respective to ids

Parameters:

- ids - a list of user URIs, URLs or IDs

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*, *market=None*)
Gets a list of the albums saved in the current authorized user's "Your Music" library

Parameters:

- limit - the number of albums to return (MAX_LIMIT=50)
- offset - the index of the first album to return
- market - an ISO 3166-1 alpha-2 country code.

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_episodes(*limit=20*, *offset=0*, *market=None*)
Gets a list of the episodes saved in the current authorized user's "Your Music" library

Parameters:

- limit - the number of episodes to return
- offset - the index of the first episode to return
- market - an ISO 3166-1 alpha-2 country code

current_user_saved_episodes_add(*episodes=None*)
Add one or more episodes to the current user's "Your Music" library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

current_user_saved_episodes_contains(*episodes=None*)
Check if one or more episodes is already saved in the current Spotify user's "Your Music" library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

current_user_saved_episodes_delete(*episodes=None*)
Remove one or more episodes from the current user's "Your Music" library.

Parameters:

- episodes - a list of episode URIs, URLs or IDs

current_user_saved_shows(*limit=20*, *offset=0*, *market=None*)
Gets a list of the shows saved in the current authorized user's "Your Music" library

Parameters:

- limit - the number of shows to return
- offset - the index of the first show to return
- market - an ISO 3166-1 alpha-2 country code

current_user_saved_shows_add(*shows=[]*)
Add one or more albums to the current user's "Your Music" library. Parameters:

- shows - a list of show URIs, URLs or IDs

current_user_saved_shows_contains(*shows=[]*)
Check if one or more shows is already saved in the current Spotify user's "Your Music" library.

Parameters:

- shows - a list of show URIs, URLs or IDs

current_user_saved_shows_delete(*shows=[]*)
Remove one or more shows from the current user's "Your Music" library.

Parameters:

- shows - a list of show URIs, URLs or IDs

current_user_saved_tracks(*limit=20*, *offset=0*, *market=None*)
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
- market - an ISO 3166-1 alpha-2 country code

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

current_user_unfollow_playlist(*playlist_id*)
Unfollows (deletes) a playlist for the current authenticated user

Parameters:

- name - the name of the playlist

currently_playing(*market=None*, *additional_types=None*)
Get user's currently playing track.

Parameters:

- market - an ISO 3166-1 alpha-2 country code.
- additional_types - *episode* to get podcast track information

default_retry_codes = (429, 500, 502, 503, 504)

devices()
Get a list of user's available devices.

episode(*episode_id*, *market=None*)
returns a single episode given the episode's ID, URIs or URL

Parameters:

- episode_id - the episode ID, URI or URL
- **market - an ISO 3166-1 alpha-2 country code.** The episode must be available in the given market. If user-based authorization is in use, the user's country takes precedence. If neither market nor user country are provided, the content is considered unavailable for the client.

episodes(*episodes*, *market=None*)
returns a list of episodes given the episode IDs, URIs, or URLs

Parameters:

- episodes - a list of episode IDs, URIs or URLs
- **market - an ISO 3166-1 alpha-2 country code.** Only episodes available in the given market will be returned. If user-based authorization is in use, the user's country takes precedence. If neither market nor user country are provided, the content is considered unavailable for the client.

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 639-1 alpha-2 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_retries = 3

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*, *additional_types=('track',)*)
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.
- **additional_types - list of item types to return.** valid types are: track and episode

playlist_add_items(*playlist_id*, *items*, *position=None*)
Adds tracks/episodes to a playlist

Parameters:

- playlist_id - the id of the playlist
- items - a list of track/episode URIs, URLs or IDs
- position - the position to add the tracks

playlist_change_details(*playlist_id*, *name=None*, *public=None*, *collaborative=None*, *description=None*)
Changes a playlist's name and/or public/private state, collaborative state, and/or description

Parameters:

- 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

playlist_cover_image(*playlist_id*)
Get cover image of a playlist.

Parameters:

- playlist_id - the playlist ID, URI or URL

playlist_is_following(*playlist_id*, *user_ids*)
Check to see if the given users are following the given playlist

Parameters:

- 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.

playlist_items (*playlist_id*, *fields=None*, *limit=100*, *offset=0*, *market=None*, *additional_types=('track', 'episode')*)

Get full details of the tracks and episodes of a playlist.

Parameters:

- playlist_id - the playlist ID, URI or URL
- 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.
- **additional_types - list of item types to return.** valid types are: track and episode

playlist_remove_all_occurrences_of_items (*playlist_id*, *items*, *snapshot_id=None*)

Removes all occurrences of the given tracks/episodes from the given playlist

Parameters:

- playlist_id - the id of the playlist
- items - list of track/episode ids to remove from the playlist
- snapshot_id - optional id of the playlist snapshot

playlist_remove_specific_occurrences_of_items (*playlist_id*, *items*, *snapshot_id=None*)

Removes all occurrences of the given tracks from the given playlist

Parameters:

- playlist_id - the id of the playlist
- **items - an array of objects containing Spotify URIs of the** tracks/episodes 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

playlist_reorder_items (*playlist_id*, *range_start*, *insert_before*, *range_length=1*, *snapshot_id=None*)

Reorder tracks in a playlist

Parameters:

- 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

playlist_replace_items (*playlist_id*, *items*)

Replace all tracks/episodes in a playlist

Parameters:

- playlist_id - the id of the playlist
- items - list of track/episode ids to comprise playlist

playlist_tracks(*playlist_id*, *fields=None*, *limit=100*, *offset=0*, *market=None*, *additional_types=('track',)*)

Get full details of the tracks of a playlist.

Parameters:

- playlist_id - the playlist ID, URI or URL
- 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.
- **additional_types - list of item types to return.** valid types are: track and episode

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/) # noqa
- **limit - the number of items to return (min = 1, default = 10, max = 50). The limit is applied** within each type, not on the total response.
- offset - the index of the first item to return
- **type - the types of items to return. One or more of 'artist', 'album',** 'track', 'playlist', 'show', and 'episode'. If multiple types are desired, pass in a comma separated string; e.g., 'track,album,episode'.
- **market - An ISO 3166-1 alpha-2 country code or the string** from_token.

search_markets (*q*, *limit=10*, *offset=0*, *type='track'*, *markets=None*, *total=None*)
(experimental) Searches multiple markets 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/) # noqa
- **limit - the number of items to return (min = 1, default = 10, max = 50). If a search is to be done on multiple** markets, then this limit is applied to each market. (e.g. search US, CA, MX each with a limit of 10).
- offset - the index of the first item to return
- **type - the types of items to return. One or more of 'artist', 'album',** 'track', 'playlist', 'show', or 'episode'. If multiple types are desired, pass in a comma separated string.
- markets - A list of ISO 3166-1 alpha-2 country codes. Search all country markets by default.
- **total - the total number of results to return if multiple markets are supplied in the search.** If multiple types are specified, this only applies to the first type.

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

set_auth (*auth*)

show (*show_id*, *market=None*)
returns a single show given the show's ID, URIs or URL

Parameters:

- show_id - the show ID, URI or URL
- **market - an ISO 3166-1 alpha-2 country code.** The show must be available in the given market. If user-based authorization is in use, the user's country takes precedence. If neither market nor user country are provided, the content is considered unavailable for the client.

show_episodes (*show_id*, *limit=50*, *offset=0*, *market=None*)
Get Spotify catalog information about a show's episodes

Parameters:

- show_id - the show ID, URI or URL
- limit - the number of items to return
- offset - the index of the first item to return
- **market - an ISO 3166-1 alpha-2 country code.** Only episodes available in the given market will be returned. If user-based authorization is in use, the user's country takes precedence. If neither market nor user country are provided, the content is considered unavailable for the client.

shows (*shows*, *market=None*)
returns a list of shows given the show IDs, URIs, or URLs

Parameters:

- shows - a list of show IDs, URIs or URLs
- **market - an ISO 3166-1 alpha-2 country code.** Only shows available in the given market will be returned. If user-based authorization is in use, the user's country takes precedence. If neither market nor user country are provided, the content is considered unavailable for the client.

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.

track(*track_id*, *market=None*)
returns a single track given the track's ID, URI or URL

Parameters:

- track_id - a spotify URI, URL or ID
- market - an ISO 3166-1 alpha-2 country code.

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*)

user_playlist_change_details(*user*, *playlist_id*, *name=None*, *public=None*, *collaborative=None*, *description=None*)

user_playlist_create(*user*, *name*, *public=True*, *collaborative=False*, *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
- collaborative - is the created playlist collaborative
- 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 from a user

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 for a user

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*, *reason=None*, *headers=None*)
Bases: exceptions.Exception

__init__(*http_status*, *code*, *msg*, *reason=None*, *headers=None*)
x.__init__(...) initializes x; see help(type(x)) for signature

CHAPTER 11

oauth2 Module

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

Bases: spotipy.oauth2.SpotifyAuthBase

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

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

Creates a Client Credentials Flow Manager.

The Client Credentials flow is used in server-to-server authentication. Only endpoints that do not access user information can be accessed. This means that endpoints that require authorization scopes cannot be accessed. The advantage, however, of this authorization flow is that it does not require any user interaction

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

Parameters:

- client_id: Must be supplied or set as environment variable
- client_secret: Must be supplied or set as environment variable
- proxies: Optional, proxy for the requests library to route through
- requests_session: A Requests session
- **requests_timeout: Optional, tell Requests to stop waiting for a response after** a given number of seconds
- **cache_handler: An instance of the *CacheHandler* class to handle** getting and saving cached authorization tokens. Optional, will otherwise use *CacheFileHandler*. (takes precedence over *cache_path* and *username*)

get_access_token(*as_dict=True*, *check_cache=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.

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_session=True*, *requests_timeout=None*, *open_browser=True*, *cache_handler=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_session=True*, *requests_timeout=None*, *open_browser=True*, *cache_handler=None*)

Creates a SpotifyOAuth object

Parameters:

- client_id: Must be supplied or set as environment variable
- client_secret: Must be supplied or set as environment variable
- redirect_uri: Must be supplied or set as environment variable
- state: Optional, no verification is performed
- **scope: Optional, either a list of scopes or comma separated string of scopes.** e.g, "playlist-read-private,playlist-read-collaborative"
- **cache_path: (deprecated) Optional, will otherwise be generated** (takes precedence over *username*)
- **username: (deprecated) Optional or set as environment variable** (will set *cache_path* to *.cache-{username}*)
- proxies: Optional, proxy for the requests library to route through
- show_dialog: Optional, interpreted as boolean
- requests_session: A Requests session
- **requests_timeout: Optional, tell Requests to stop waiting for a response after** a given number of seconds
- **open_browser: Optional, whether or not the web browser should be opened to** authorize a user
- **cache_handler: An instance of the *CacheHandler* class to handle** getting and saving cached authorization tokens. Optional, will otherwise use *CacheFileHandler*. (takes precedence over *cache_path* and *username*)

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(*open_browser=None*)

get_authorization_code(*response=None*)

get_authorize_url(*state=None*)
Gets the URL to use to authorize this app

get_cached_token()

static parse_auth_response_url(*url*)

parse_response_code(*url*)
Parse the response code in the given response url

Parameters:

- url - the response url

refresh_access_token(*refresh_token*)

validate_token(*token_info*)

exception spotipy.oauth2.**SpotifyOauthError**(*message*, *error=None*, *error_description=None*, **args*, ***kwargs*)

Bases: exceptions.Exception

Error during Auth Code or Implicit Grant flow

__init__(*message*, *error=None*, *error_description=None*, **args*, ***kwargs*)
x.__init__(...) initializes x; see help(type(x)) for signature

exception spotipy.oauth2.**SpotifyStateError**(*local_state=None*, *remote_state=None*, *message=None*, *error=None*, *error_description=None*, **args*, ***kwargs*)

Bases: *spotipy.oauth2.SpotifyOauthError*

The state sent and state recieved were different

__init__(*local_state=None*, *remote_state=None*, *message=None*, *error=None*, *error_description=None*, **args*, ***kwargs*)
x.__init__(...) initializes x; see help(type(x)) for signature

class spotipy.oauth2.**SpotifyImplicitGrant**(*client_id=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *show_dialog=False*, *cache_handler=None*)

Bases: spotipy.oauth2.SpotifyAuthBase

Implements Implicit Grant Flow for client apps

This auth manager enables *user and non-user* endpoints with only a client secret, redirect uri, and username. The user will need to copy and paste a URI from the browser every hour.

The OAuth standard no longer recommends the Implicit Grant Flow for client-side code. Spotify has implemented the OAuth-suggested PKCE extension that removes the need for a client secret in the Authentication Code flow. Use the SpotifyPKCE auth manager instead of SpotifyImplicitGrant.

SpotifyPKCE contains all of the functionality of SpotifyImplicitGrant, plus automatic response retrieval and refreshable tokens. Only a few replacements need to be made:

- get_auth_response()['access_token'] -> get_access_token(get_authorization_code())
- get_auth_response() -> get_access_token(get_authorization_code()); get_cached_token()
- parse_response_token(url)['access_token'] -> get_access_token(parse_response_code(url))
- parse_response_token(url) -> get_access_token(parse_response_code(url)); get_cached_token()

The security concern in the Implict Grant flow is that the token is returned in the URL and can be intercepted through the browser. A request with an authorization code and proof of origin could not be easily intercepted without a compromised network.

OAUTH_AUTHORIZE_URL = 'https://accounts.spotify.com/authorize'

__init__(*client_id=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *show_dialog=False*, *cache_handler=None*)

Creates Auth Manager using the Implicit Grant flow

See help(SpotifyImplictGrant) for full Security Warning

- client_id: Must be supplied or set as environment variable
- redirect_uri: Must be supplied or set as environment variable
- state: May be supplied, no verification is performed
- **scope: Optional, either a list of scopes or comma separated string of scopes.** e.g, "playlist-read-private,playlist-read-collaborative"
- **cache_handler: An instance of the *CacheHandler* class to handle** getting and saving cached authorization tokens. May be supplied, will otherwise use *CacheFileHandler*. (takes precedence over *cache_path* and *username*)
- **cache_path: (deprecated) May be supplied, will otherwise be generated** (takes precedence over *username*)
- **username: (deprecated) May be supplied or set as environment variable** (will set *cache_path* to *.cache-{username}*)
- show_dialog: Interpreted as boolean

get_access_token(*state=None*, *response=None*, *check_cache=True*)

Gets Auth Token from cache (preferred) or user interaction

- state: May be given, overrides (without changing) self.state
- response: URI with token, can break expiration checks
- check_cache: Interpreted as boolean

get_auth_response(*state=None*)

Gets a new auth **token** with user interaction

get_authorize_url(*state=None*)

Gets the URL to use to authorize this app

get_cached_token()

static parse_auth_response_url(*url*)

parse_response_token(*url*, *state=None*)

Parse the response code in the given response url

validate_token(*token_info*)

class spotipy.oauth2.**SpotifyPKCE**(*client_id=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *proxies=None*, *requests_timeout=None*, *requests_session=True*, *open_browser=True*, *cache_handler=None*)

Bases: spotipy.oauth2.SpotifyAuthBase

Implements PKCE Authorization Flow for client apps

This auth manager enables *user and non-user* endpoints with only a client secret, redirect uri, and username. When the app requests an an access token for the first time, the user is prompted to authorize the new client app. After authorizing the app, the client app is then given both access and refresh tokens. This is the preferred way of authorizing a mobile/desktop client.

OAUTH_AUTHORIZE_URL = 'https://accounts.spotify.com/authorize'

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

__init__(*client_id=None*, *redirect_uri=None*, *state=None*, *scope=None*, *cache_path=None*, *username=None*, *proxies=None*, *requests_timeout=None*, *requests_session=True*, *open_browser=True*, *cache_handler=None*)

Creates Auth Manager with the PKCE Auth flow.

Parameters:

- client_id: Must be supplied or set as environment variable
- redirect_uri: Must be supplied or set as environment variable
- state: Optional, no verification is performed
- **scope: Optional, either a list of scopes or comma separated string of scopes.** e.g, "playlist-read-private,playlist-read-collaborative"
- **cache_path: (deprecated) Optional, will otherwise be generated** (takes precedence over *username*)
- **username: (deprecated) Optional or set as environment variable** (will set *cache_path* to *.cache-{username}*)
- proxies: Optional, proxy for the requests library to route through
- **requests_timeout: Optional, tell Requests to stop waiting for a response after** a given number of seconds
- requests_session: A Requests session
- **open_browser: Optional, thether or not the web browser should be opened to** authorize a user
- **cache_handler: An instance of the *CacheHandler* class to handle** getting and saving cached authorization tokens. Optional, will otherwise use *CacheFileHandler*. (takes precedence over *cache_path* and *username*)

get_access_token(*code=None*, *check_cache=True*)

Gets the access token for the app

If the code is not given and no cached token is used, an authentication window will be shown to the user to get a new code.

Parameters:

- code - the response code from authentication
- **check_cache - if true, checks for a locally stored token** before requesting a new token

get_authorization_code(*response=None*)

get_authorize_url(*state=None*)

Gets the URL to use to authorize this app

get_cached_token()

get_pkce_handshake_parameters()

static parse_auth_response_url(*url*)

parse_response_code(*url*)
Parse the response code in the given response url

Parameters:

- url - the response url

refresh_access_token(*refresh_token*)

validate_token(*token_info*)

11.1 util Module

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

spotipy.util.**prompt_for_user_token**(*username=None*, *scope=None*, *client_id=None*, *client_secret=None*, *redirect_uri=None*, *cache_path=None*, *oauth_manager=None*, *show_dialog=False*)

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

If you are a developer with Python experience, and you would like to contribute to Spotipy, please be sure to follow the guidelines listed below:

Export the needed Environment variables::: export SPOTIPY_CLIENT_ID=client_id_here export SPOTIPY_CLIENT_SECRET=client_secret_here export SPOTIPY_CLIENT_USERNAME=client_username_here # This is actually an id not spotify display name export SPOTIPY_REDIRECT_URI=http://localhost:8080 # Make url is set in app you created to get your ID and SECRET

Create virtual environment, install dependencies, run tests::: $ virtualenv –python=python3.7 env (env) $ pip install –user -e . (env) $ python -m unittest discover -v tests

Lint

To automatically fix the code style::: pip install autopep8 autopep8 –in-place –aggressive –recursive .

To verify the code style::: pip install flake8 flake8 .

To make sure if the import lists are stored correctly::: pip install isort isort . -c -v

Publishing (by maintainer)

- Bump version in setup.py
- Bump and date changelog
- Add to changelog:

:: ## Unreleased

// Add your changes here and then delete this line

- Commit changes
- Package to pypi:

:: python setup.py sdist bdist_wheel python3 setup.py sdist bdist_wheel twine check dist/* twine upload –repository-url https://upload.pypi.org/legacy/ –skip-existing dist/.*(whl|gz|zip)~dist/*linux*.whl

- Create github release https://github.com/plamere/spotipy/releases with the changelog content for the version and a short name that describes the main addition
- Build the documentation again to ensure it's on the latest version

Changelog

Don't forget to add a short description of your change in the CHANGELOG!

CHAPTER 14

License

(Taken from https://github.com/plamere/spotipy/blob/master/LICENSE.md):

```
MIT License
Copyright (c) 2021 Paul Lamere
Permission is hereby granted, free of charge, to any person obtaining a copy of this␣
↪software and associated documentation files
(the "Software"), to deal in the Software without restriction, including without␣
↪limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software, and to permit␣
↪persons to whom the Software is furnished to do
so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies␣
↪or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,␣
↪INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT␣
↪SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,␣
↪TORT OR OTHERWISE, ARISING FROM, OUT OF OR
IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

CHAPTER 15

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

C

D

E

F

G

M

N

O

P

R

S

T

U

V