Spotipy Documentation

Classes

1. Album: Corresponds to the Album JSON object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - albumType: String. What "type" of album it is, "single", "album", or "compilation".
    - artists: Dictionary of artists on the album, with the key being the name and the id being the value ({name:id})
    - externalIds: 2-String Dictionary. IDs for the album, with the key being the ‘type’ (ie: Spotify), and the actual id being the value. ({type:id})
    - externalUrls: 2-String Dictionary. URLs for the album, with the key being the 'location' (eg: Spotify/iTunes/etc.) and the actual url being the value. ({location:url})
    - genres: String Array. Each string is a genre that the album is associated with.
    - List<String>. Each string is a URL to an image associated with the album.
    - name: String. The name of the album.
    - releaseDate: String. The Date and Time an album was released, with a structure of: (YYYY-MM-DD HH:MM:SS.SSZ)
    - tracks: 2-String Dictionary. Contains all of the tracks associated with the dictionary, with their name being the key, and id being the value. Show here is the structure: ({name:id})
    - URI: String. The Spotify URI for the Album.

2. Artist: Corresponds to the Album JSON Object given in the Spotify Web API documents, as well as the simplified version. Listed below are the object's attributes and their corresponding types:
    - externalURLs: 2-String Dictionary. URLs for the artist, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
    - followers: Integer. Number of followers this artist has.
    - genres: String Array. Each string is a genre that the artist is associated with.
    - href: String. A URL linking to the Web API's Spotify information about the Artist.
    - artistId: String. The Spotify ID for the Artist.
    - images: List<String>. Each string is a URL to an image associated with the Artist.
    - artistName: String. The name of the artist.
    - popularity: Integer. A value ranging from 0-100, with a higher value being a higher overall popularity, which is calculated by Spotify by all of the artist's tracks.
    - type: String. The JSON Object type, which should always return "artist".
    - userUri: String. The Spotify URI for the Artist.

3. Category: Corresponds to the Album JSON object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:

- href: String. What "type" of album it is, "single", "album", or "compilation".
- icons: Dictionary of artists on the album, with the key being the name and the id being the value ({name:id})
- id: A reference to the ExternalID Object, which contains all of the IDs for the associated Album.
- name: 2-String Dictionary. URLs for the album, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})

4. CurrentlyPlayingObject: Corresponds to the CurrentlyPlayingObject JSON Object given in the Spotify Web API documents. Listed below are the object's attributes and their corresponding types:
   - context: 2-String Dictionary. Contains the href, type, and uri for the context of whatever is currently playing. ({'href':'url', 'type':'context', 'uri':'url'})
   - currentlyPlayingType: String. The object Type of whatever item is currently playing; either "track", "episode", "ad", or "unknown".
   - isPlaying: Boolean. Returns true if something is playing, false otherwise.
   - item: String. The currently playing track. Can potentially be None.
   - progressMs: Integer. The progress into the currently playing track, in milliseconds. Can potentially be None.
   - timestamp: Integer. The Unix millisecond Timestamp as to when the data was obtained.

5. DeviceObject: Corresponds to the DeviceObject JSON Object given in the Spotify Web API documents. Listed below are the object's attributes and their corresponding types:
    - deviceId: String. The ID of the device playing Spotify. Can potentially be None.
    - isActive: Boolean. Returns true if the device is currently the active device, false otherwise.
    - isPrivateSession: Boolean. Returns true if the current session is private, false otherwise.
    - isRestricted: Boolean. Returns true if the device will not accept Web API commands, false otherwise.
    - name: String. The name of the device.
    - type: String. The type of the device, eg: "Computer", "Smartphone", "Speaker", etc.
    - volumePercent: Integer. The current volume, in a percentage (eg: 75 = 75%). Can potentially be None.

6. External ID: Corresponds to the External ID JSON object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - ean: String. The International Article Number for the External ID.
    - isrc: String. The International Standard Recording Code for the External ID.
    - upc: String. The Universal Product Code for the External ID.

7. PlaylistObject: Corresponds to the Playlist JSON Object given in the Spotify Web API documents, as well as the simplified version. Listed below are the object's attributes and their corresponding types:
    - collaborative: Boolean. Returns true if the owner allows other users to modify the Playlist, false otherwise.
    - externalUrls: 2-String Dictionary. URLs for the playlist, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
    - href: String. A URL linking to the Web API's Spotify information about the Playlist.
    - playlistId: String. The Spotify ID for the Playlist.
    - images: List<String>. Each string is a URL to an image associated with the Playlist.
    - name: String. The name of the Playlist.
    - owner: 2-String Dictionary. The key is the name of the owner, while the value is their Spotify ID. ({name:id})
    - public: Boolean. Returns true if the playlist is public, false if it is private, and None if it is not relevant.
    - snapshotId: String. The version ID for the playlist. Used to help target a specific version of a playlist.
    - tracks: 2-String Dictionary. The key is the name of the track, whereas the value is the track's Spotify ID ({name:id}).
    - type: String. The JSON Object type, which should always return "playlist".
    - uri: String. The Spotify URI for the Playlist.

8. PrivateUserObject: Corresponds to the PrivateUserObject JSON Object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - country: String. The ISO 3166-1 aplha-2 country code set by the user.
    - displayName: String. The name displayed on the user's profile. Can potentially be None.
    - email: String. The email address set by the user. There is no proof this actually belongs to the user, so use with caution.
    - externalUrls: 2-String Dictionary. Known URLs for the user, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
    - href: String. A URL linking to the Web API's Spotify information about the User.
    - userId: String. The Spotify ID for the user.
    - images: List<String>. Each string is a URL to an image associated with the Playlist.
    - product: String. The user's Spotify subscription level. Can be "premium", "free", "open", etc.
    - type: String. The JSON Object type, which should always return "user".
    - userUri: String. The Spotify URI for the User.

9. PublicUserObject: Corresponds to the PublicUserObject JSON Object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:

- displayName: String. The name displayed on the user's profile. Can potentially be None.
- externalUrls: 2-String Dictionary. Known URLs for the user, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
- followers: Integer. The number of followers the user has.
- href: String. A URL linking to the Web API's Spotify information about the User.
- userPublicObjectId: String. The Spotify ID for the user.
- userPublicProfileImages: List<String>. Each string is a URL to an image associated with the Playlist.
- userType: String. The JSON Object type, which should always return "user".
- userUri: String. The Spotify URI for the User.

10. Track: Corresponds to the TrackObject JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - albumId: String. The Spotify ID of the album the track is on.
    - artists: List<String>. Each String is an artist name who performs on the track.
    - availableMarkets: List<String>. A list of the countries in which the track can be played, each identified by their ISO 3166-1 aplha-2 code.
    - discNum: Integer. The disc on which the track is on (usually 1, may be more if an album contains >1 disc).
    - durationMs: Integer. The length of the track in milliseconds.

- explicit: Boolean. Returns true if the track is known to contain explicit lyrics, false otherwise.
- externalIds: 2-String Dictionary. IDs for the Track, with the key being the ‘type’ (eg: Spotify), and the actual id being the value. ({type:id})
- externalUrls: 2-String Dictionary. URLs for the track, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
- href: String. A URL linking to the Web API's Spotify information about the Track.
- trackId: String. The Spotify ID for the Track.
- isPlayable: Boolean. Returns true if the Track is playable in the user's current market, false otherwise.
- linkedFrom: String. A URL linking to another version of the track that was unavailable in the user's current market.
- restrictions: 2-String Dictionary. Contains the restriction for the 'linkedFrom' track and their reasoning for each market. Key is the reason, and value is the market itself. ({"reason" : "market"})
- name: String. The name of the Track.
- popularity: Integer. A value ranging from 0-100, with a higher value being a higher overall popularity, which is calculated by Spotify.
- previewUrl: String. A URL linking to a 30-second preview of the track. Can potentially be None.
- trackNum: Integer. The number of the track on its album. If an album has several discs, the track number specifically on that disc (eg: A track can have discNum = 2, trackNum = 1.)

- type: String. The JSON Object type, which should always return "track".
- uri: String. The Spotify URI for the Track.
- isLocal: Boolean. Returns true if the track is from a local file, false otherwise.

11. SavedTrack: Corresponds to the TrackObject JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - albumId: String. The Spotify ID of the album the track is on.
    - artists: List<String>. Each String is an artist name who performs on the track.
    - availableMarkets: List<String>. A list of the countries in which the track can be played, each identified by their ISO 3166-1 alpha-2 code.
    - discNum: Integer. The disc on which the track is on (usually 1, may be more if an album contains >1 disc).
    - durationMs: Integer. The length of the track in milliseconds.
    - explicit: Boolean. Returns true if the track is known to contain explicit lyrics, false otherwise.
    - externalIds: 2-String Dictionary. IDs for the Track, with the key being the ‘type’ (eg: Spotify), and the actual id being the value. ({type:id})
    - externalUrls: 2-String Dictionary. URLs for the track, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
    - href: String. A URL linking to the Web API's Spotify information about the Track.

- trackId: String. The Spotify ID for the Track.
- isPlayable: Boolean. Returns true if the Track is playable in the user's current market, false otherwise.
- linkedFrom: String. A URL linking to another version of the track that was unavailable in the user's current market.
- restrictions: 2-String Dictionary. Contains the restriction for the 'linkedFrom' track and their reasoning for each market. Key is the reason, and value is the market itself. ({"reason" : "market"})
- name: String. The name of the Track.
- popularity: Integer. A value ranging from 0-100, with a higher value being a higher overall popularity, which is calculated by Spotify.
- previewUrl: String. A URL linking to a 30-second preview of the track. Can potentially be None.
- trackNum: Integer. The number of the track on its album. If an album has several discs, the track number specifically on that disc (eg: A track can have discNum = 2, trackNum = 1.)
- type: String. The JSON Object type, which should always return "track".
- uri: String. The Spotify URI for the Track.
- isLocal: Boolean. Returns true if the track is from a local file, false otherwise.
- timestamp: String. The date and time at which the track was saved, in ISO 8601 format, ie: (YYYY-MM-DDTHH:MM:SSZ). May be imprecise.

12. AudioFeatureObject: Corresponds to the Audio Features JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - acousticness: Float. A value from 0.0 to 1.0, measuring how acoustic the track is. A higher value means it is more likely acoustic.
    - analysisUrl: String. A URL with access to the full audio analysis of the track. An access token is required to be able to use this.
    - danceability: Float. A value from 0.0 to 1.0, measuring how danceable the track is. A higher value means the track is more danceable.
    - durationMS: Integer. The duration of the track in milliseconds.
    - energy: Float. A value from 0.0 to 1.0 that measures the intensity of the track. A higher value means the track is more intense.
    - trackId: String. The Spotify ID for the Track.
    - instrumentalness: Float. A value from 0.0 to 1.0 to predict whether the track contains vocals or not. A value of 0.5 or higher is intended to represent instrumental tracks, with higher values meaning fewer vocals.
    - key: Integer. The music key that the Track is in, using standard Pitch Class Notation. (EG: C = 0, C# = 1, D = 2, etc.)
    - liveness: Float. A value from 0.0 to 1.0 that measures the chance that there is an audience in the track. A higher value means there is a greater chance, with values at or above 0.8 meaning there is an almost certain chance there is an audience (and the track is being performed live).

- loudness: Float. The overall loudness of the track in decibels. Values typically range between -60 and 0 db.
- mode: Integer. Determines whether a track is in a Major Key or Minor Key. 1 is Major, 0 is Minor.
- speechiness: Float. A value between 0.0 and 1.0 that determines how much normal speech is in the track. Values below 0.33 likely have almost no spoken lines, values between 0.33 and 0.66 may be more of a mix of the two (see: rap), and values above 0.66 likely contain only spoken words. A higher value means there are more spoken words and fewer sung words.
- tempo: Float. The overall estimated tempo of the track in Beats per Minute (BPM). A higher values means an overall faster track.
- timeSignature: Integer. An overall estimated value of the track's time signature. The time signature is a convention used to determine how many beats are in a single measure (bar).
- trackHref: String. A link to the Web API endpoint for the full details of the Track.
- type: String. The JSON Object Type, which should always return "audio_features".
- uri: String. The Spotify URI for the Track.
- valence: Float. A value between 0.0 and 1.0 describing the overall positiveness of the track. A higher value means that the track feels more positive overall.

13. PagingObject: Corresponds to the Paging JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - href: String. A URL linking to the Spotify Web API's information about the item within the Paging Object.
    - items: List<Object>. A list of the items in the Paging Object.
    - limit: Integer. The maximum number of items within the response. Either set by the query or the default value (which is not known atm).
    - nextItem: String. A URL to the next page of items. Can potentially be None if there are no more items left.
    - offset: Integer. The offset of the items returned. Either set by the query or the default value (which is not known atm).
    - previousItem: String. A URL to the previous page of items. Can potentially be None if the item is the first one in the list.
    - total: Integer. The total number of items available to return.

14. CursorBasedPagingObject: Corresponds to the Cursor Paging JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - href: String. A URL linking to the Spotify Web API's information about the item within the Paging Object.
    - items: List<Object>. A list of the items in the Paging Object.
    - nextPage: String. A URL to the next page of items. Can potentially be None if there are no more items left.

- cursors: String. The value of the entry in the dictionary that Spotify returns. Generally a url.
- total: Integer. The total number of items available to return.

15. PlayHistoryObject: Corresponds to the Play History JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - context: ContextObject. The context in which the track was played from.
    - playedAt: String. The date and time at which the track began to play, with a structure of: (YYYY-MM-DD HH:MM:SS.SSZ).
    - tracks: 2-String Dictionary. Contains all of the tracks associated with the play history, with their name being the key, and id being the value. ({name:id}).

16. PlaylistTrackObject: Corresponds to the Playlist Track JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - addedAt: String. The date and time at which the track was added to the playlist, with a structure of: (YYYY-MM-DD HH:MM:SS.SSZ). May be null if the track/playlist was extremely old.
    - addedBy: String. The Spotify display name for the user who added the Track to the Playlist.
    - isLocal: Boolean. Returns true if the track is from a local file, false otherwise. Can potentially be None.

- track: TrackObject. Is the Track associated with the PlaylistTrackObject.

17. RecommendationSeedObject: Corresponds to the Recommendation Seed JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - initPoolSize: Integer. The number of recommended tracks available for this seed.
    - afterFilterSize: Integer. The number of tracks after the min and max filters have been applied.
    - afterRelinkingSize: Integer. The number of tracks available after relinking for regional availability.
    - href: String. The href to the TrackObject or ArtistObject associated with this seed. For genre-based seeds, will be None.
    - recommendationId: String. The id used to select this seed.
    - recommendationType: String. The type of this seed. Will be “artist”, “track”, or “genre”.

18. RecommendationsResponseObject: Corresponds to the Recommendations Response JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - seeds: List<RecommendationSeedObject>. A list of the seed objects associated with the Response.

- tracks: List<TrackObject>. A list of tracks given back to the user.

19. ContextObject: Corresponds to the Context JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - uri: String. The Spotify uri for the context.
    - href: String. A link to the Web API endpoint with full details of the item.
    - externalUrls: 2-String Dictionary. URLs for the item, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})
    - type: String. The type of item. Can be “artist”, “playlist”, “album”, “track”, or others.

20. Saved Album: Corresponds to the Album JSON object given in the Spotify Web API documents, with some types omitted for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - albumType: String. What "type" of album it is, "single", "album", or "compilation".
    - artists: Dictionary of artists on the album, with the key being the name and the id being the value ({name:id})
    - externalIds: 2-String Dictionary. IDs for the album, with the key being the ‘type’ (eg: Spotify), and the actual id being the value. ({type:id})
    - externalUrls: 2-String Dictionary. URLs for the album, with the key being the 'location' (eg: Spotify/ITunes/etc.) and the actual url being the value. ({location:url})

- genres: String Array. Each string is a genre that the album is associated with.
- List<String>. Each string is a URL to an image associated with the album.
- name: String. The name of the album.
- releaseDate: String. The Date and Time an album was released, with a structure of: (YYYY-MM-DD HH:MM:SS.SSZ)
- tracks: 2-String Dictionary. Contains all of the tracks associated with the dictionary, with their name being the key, and id being the value. Show here iis the structure: ({name:id})
- URI: String. The Spotify URI for the Album.

21. TunableTrackObject: Corresponds to the Tunable Track JSON Object given in the Spotify Web API documents, with some types modified for simplicity's sake. Listed below are the object's attributes and their corresponding types:
    - acousticness: Float. A value from 0.0 to 1.0, measuring how acoustic the track is. A higher value means it is more likely acoustic.
    - danceability: Float. A value from 0.0 to 1.0, measuring how danceable the track is. A higher value means the track is more danceable.
    - durationMS: Integer. The duration of the track in milliseconds.
    - energy: Float. A value from 0.0 to 1.0 that measures the intensity of the track. A higher value means the track is more intense.
    - instrumentalness: Float. A value from 0.0 to 1.0 to predict whether the track contains vocals or not. A value of 0.5 or higher is intended to

represent instrumental tracks, with higher values meaning fewer vocals.

- key: Integer. The music key that the Track is in, using standard Pitch Class Notation. (EG: C = 0, C# = 1, D = 2, etc.)
- liveness: Float. A value from 0.0 to 1.0 that measures the chance that there is an audience in the track. A higher value means there is a greater chance, with values at or above 0.8 meaning there is an almost certain chance there is an audience (and the track is being performed live).
- loudness: Float. The overall loudness of the track in decibels. Values typically range between -60 and 0 db.
- mode: Integer. Determines whether a track is in a Major Key or Minor Key. 1 is Major, 0 is Minor.
- popularity: Integer. A value ranging from 0-100, with a higher value being a higher overall popularity, which is calculated by Spotify.
- speechiness: Float. A value between 0.0 and 1.0 that determines how much normal speech is in the track. Values below 0.33 likely have almost no spoken lines, values between 0.33 and 0.66 may be more of a mix of the two (see: rap), and values above 0.66 likely contain only spoken words. A higher value means there are more spoken words and fewer sung words.
- tempo: Float. The overall estimated tempo of the track in Beats per Minute (BPM). A higher value means an overall faster track.
- timeSignature: Integer. An overall estimated value of the track's time signature. The time signature is a convention used to determine how many beats are in a single measure (bar).

- valence: Float. A value between 0.0 and 1.0 describing the overall positiveness of the track. A higher value means that the track feels more positive overall.

Functions

Preface: All of these functions require a user’s refresh token, the client id and the client secret for the Spotify application requesting this information. All three of these should be strings.

Library API

1. removeAlbumsforUser(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_modify scope. Takes in a string which should be a comma-separated list of album ids with no spaces and returns a 200 response if it successfully removes the albums from the user’s library. Returns a 403 response if you do not have permission.

2. removeUsersSavedTracks(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_modify scope. Takes in a string which should be a be a comma-separated list of track ids with no spaces and returns a 200 response if it successfully removes the tracks from the user’s library. Returns a 403 response if you do not have permission.

3. checkUsersSavedAlbums(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_read scope. Takes in a string which should be

a be a comma-separated list of album ids with no spaces and returns a list of Boolean values with the same order as the album ids given. True corresponds to the user having the album, False corresponds to the user not having the album in their library. Returns a 403 response if you do not have permission.

4. saveAlbumsforUser(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_modify scope. Takes in a string which should be a comma-separated list of album ids with no spaces and returns a 200 response if it successfully adds the albums to the user's library. Returns a 403 response if you do not have permission.

5. getUsersSavedTracks(refresh_token, client_id, client_secret, num=20, start=0):
   NOTE: Requires user_library_read scope. Takes in two optional integer arguments, num, which determines the number of tracks to return, and start, which determines the first track to return, and then returns a Paging Object containing a List of Track Objects in the user's library. Returns a 403 response if you do not have permission.

6. checkUsersSavedAlbums(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_read scope. Takes in a string which should be a be a comma-separated list of track ids with no spaces and returns a list of Boolean values with the same order as the track ids given. True corresponds to the user having the track, False corresponds to the user not having the track in their library. Returns a 403 response if you do not have permission.

7. getUsersSavedAlbums(refresh_token, client_id, client_secret, num=20, start=0):
   NOTE: Requires user_library_read scope. Takes in two optional integer arguments, num, which determines the number of albums to return, and start, which determines the first album to return, and then returns a Paging Object containing a List of Album Objects in the user's library. Returns a 403 response if you do not have permission.

8. saveTracksforUser(refresh_token, client_id, client_secret, ids):
   NOTE: Requires user_library_modify scope. Takes in a string which should be a comma-separated list of track ids with no spaces and returns a 200 response if it successfully adds the tracks to the user's library. Returns a 403 response if you do not have permission.

Playlists API

9. replaceAPlaylistsItems(refresh_token, client_id, client_secret, uris, playlistId):
   NOTE: Requires playlist_modify_public and playlist_modify_private scopes. Takes in a string which should be a comma-separated list of uris with no spaces and a playlistId in a string. Will return a 201 response if the playlist has been successfully replaced with the user's given items. Returns a 403 response if you do not have permission.

10. getAListOfAUsersPlaylists(refresh_token, client_id, client_secret, userID):
    NOTE: Requires playlist_read_private and playlist_read_collaborative scopes. Takes in a user's Spotify ID as a String and returns a Python List of Playlist Objects that belong to that user. Returns a 403 response if you do

not have permission.

11. changeAPlaylistsDetails(refresh_token, client_id, client_secret, playlistID, name=None, public=None, collaborative=None, description=None):
    NOTE: Requires playlist_modify_public and playlist_modify_private scopes. Takes in a playlistID as a String, as well as at least one of the four optional parameters as Strings with the exception of collaborative, which must be a Boolean. Returns a 200 response if the Playlists' attributes have been changed. Returns a 403 response if you do not have permission.

12. uploadACustomPlaylistCoverImage(refresh_token, client_id, client_secret, playlistID, imageB64):
    NOTE: Requires ugc_image_upload, playlist_modify_private, and playlist_modify_public scopes. Takes in a playlistID as a String, and a base-64 encoded jpeg as a String. Returns a 202 response if the Cover Image has been successfully uploaded to the Playlist. Returns a 403 response if you do not have permission. Returns a 429 response if you are being rate-limited.

13. reorderAPlaylistsItems(refresh_token, client_id, client_secret, playlistID, range_start, insert_before, range_length=None, snapshot_id=None):
    NOTE: Requires playlist_modify_private and playlist_modify_public scopes. Takes in a playlistID as a String, and a starting location and insert location as Integer values. Optionally, can be given a third Integer value for the number of items to be reordered, and a snapshot_id of the playlist as a String. If the items have been successfully reordered, returns a 200 response. Returns a 403 response if you do not have permission.

14. addItemsToAPlaylist(refresh_token, client_id, client_secret, playlistID, listOfUris):
    NOTE: Requires playlist_modify_private and playlist_modify_public scopes. Takes in a playlistID as a String, and a string which should be a comma-separated list of uris with no spaces. Returns a 201 response if the items are successfully added to the Playlist. Returns a 403 response if you do not have permission.

15. getAPlaylistsItems(refresh_token, client_id, client_secret, playlistID):
    Takes in a playlistID as a String and returns a PagingObject containing a Python List of TrackObjects associated with the Playlist given. If you do not have access to the playlist, returns a 403 response.

16. getAPlaylistCoverImage(refresh_token, client_id, client_secret, playlistID):
    Takes in a playlistID as a String and returns a Python List of urls to the images.

17. removeItemsFromAPlaylist(refresh_token, client_id, client_secret, playlistID, listOfUris):
    NOTE: Requires playlist_modify_private and playlist_modify_public scopes. Takes in a playlistID as a String and a Python List of uris. If the items are successfully removed from the Playlist, returns a 200 response. Returns a 403 response if you do not have permission.

18. getAListOfAUsersPlaylists(refresh_token, client_id, client_secret, userID):
    NOTE: Requires playlist_read_private and playlist_read_collaborative scopes. Takes in a user ID as a String and returns a List of PlaylistObjects.

Returns a 403 response if you do not have permission.

19. getAPlaylist(refresh_token, client_id, client_secret, playlistID):
Takes in a playlist ID as a String and returns the PlaylistObject associated with the given ID.

20. createAPlaylist(refresh_token, client_id, client_secret, userID, name, public=None, collaborative=None, description=None):
NOTE: Requires playlist_modify_private and playlist_modify_public scopes. Takes in a user ID and playlist name as 2 separate Strings. Optionally, can be given a public Boolean, a collaborative Boolean, and a description String as well. Returns the PlaylistObject that the user creates. Returns a 403 response if you do not have permission.

Album API

21. getAnAlbum(refresh_token, client_id, client_secret, albumId):
Takes in an album ID as a String. Returns an Album object associated with the given ID.

22. getMultipleAlbums(refresh_token, client_id, client_secret, albumIds):
Takes a string which should be a comma-separated list of album IDs with no spaces. Returns a list of Album objects, each associated with the IDs given.

23. getAnAlbumsTracks(refresh_token, client_id, client_secret, albumId):
Takes in an album ID as a String. Returns a PagingObject which contains a list of Track objects associated with the given album ID.

Track API

24. getTrack(refresh_token, client_id, client_secret, trackId):
    Takes in a track ID as a String. Returns a Track Object for the associated Track.

25. getManyTracks(refresh_token, client_id, client_secret, list_of_ids):
    Takes in a string which should be a comma-separated list of track IDs with no spaces. Returns a list of Track objects, each associated with the IDs given.

26. getAudioAnalysisForTrack(refresh_token, client_id, client_secret, trackId):
    Takes in a track ID as a String. Returns the raw json Spotify does. (Currently a placeholder).

27. getAudioFeaturesForTrack(refresh_token, client_id, client_secret, trackId):
    Takes in a track ID as a String. Returns an AudioFeatureObject for the associated track.

28. getAudioFeaturesForSeveralTracks(refresh_token, client_id, client_secret, list_of_ids):
    Takes in a string which should be a comma-separated list of track IDs with no spaces. Returns an AudioFeatureObject for each of the associated tracks.

Browse API

29. getManyCategories(refresh_token, client_id, client_secret, num=20, start=0):
    Optionally takes in two Integer values as the number of categories to return (default 20) and the starting category's position (default 0). Returns a list of Category objects.

30. getACategory(refresh_token, client_id, client_secret, category_id):
    Takes in a category ID as a String, and returns the Category object associated with that ID.

31. getACategorysPlaylists(refresh_token, client_id, client_secret, category_id, num=20, start=0):
    Takes in a category ID as a String, and optionally, takes in two Integer values as the number of Playlist Objects to Return (default 20) and the starting Playlist's position (default 0). Returns a list of Playlist Objects.

32. getRecommendations(refresh_token, client_id, client_secret, input_artists, input_genres, input_tracks, num=20):
    Takes in three different Strings, which can be empty, or can be comma-separated lists of artist, genre, or track ids, respectively. There is a limit of 5 ids between the three options (ie: 3 artist ids, 1 genre id, and 1 track id is fine, but 3 artists, 2, genres, and 2 tracks will return an error). Optionally, can also be given a limit as to how many tracks are returned (default 20). Returns a RecommendationsResponseObject, which contains a list of RecommendationSeedObjects, each of which correspond to the given ids,

and a list of Track Objects.

33. getRecommendationGenres(refresh_token, client_id, client_secret):
Returns a Python List of strings, each of which is a recommended genre for the user.

34. getAllNewReleases(refresh_token, client_id, client_secret, num=20, start=0):
Optionally takes in two Integer values as the number of albums to return (default 20) and the starting album's position (default 0). Returns a PagingObject which contains a list of Album Objects corresponding to the most recently released Albums on Spotify.

35. getAllFeaturedPlaylists(refresh_token, client_id, client_secret, num=20, start=0):
Optionally takes in two Integer values as the number of playlists to return (default 20) and the starting playlist's position (default 0). Returns a PagingObject which contains a list of Featured Playlists.

User Profile API

36. getOtherUsersProfile(refresh_token, client_id, client_secret, user_id):
Takes in a Spotify User ID as a String, and returns a PublicUserObject associated with the user ID.

37. getCurrentUsersProfile(refresh_token, client_id, client_secret):
    Returns a PrivateUserObject for the user who gave their refresh token.

Follow API

38. getFollowingState(refresh_token, client_id, client_secret, type, ids):
    Takes in a String which should be the type of id to check, either "artist" or "user". Also takes a String which should be a comma-separated list of ids with no spaces. Returns a list of Booleans; True says the user who gave their refresh token follows that artist/user, False says the user does not follow that artist/user.

39. checkIfUsersFollowAPlaylist(refresh_token, client_id, client_secret, playlist_id, ids):
    NOTE: Requires the playlist-read-private scope. Takes in a playlist ID as a String, and a second String which should be a comma-separated list of user ids with no spaces. Returns a list of Booleans; True says the associated user that playlist, False says the associated user does not follow that playlist. Returns a 403 response if you do not have permission.

40. followAnArtistOrUser(refresh_token, client_id, client_secret, type, ids):
    NOTE: Requires the user_follow_modify scope. Takes in a String which should be the type of id to check, either "artist" or "user". Also takes a String which should be a comma-separated list of ids with no spaces. Returns a 204 response on success, if the user successfully follows the given artists or ids. Returns a 403 response if you do not have permission.

41. followAPlaylist(refresh_token, client_id, client_secret, playlist_id):
    NOTE: Requires the user_follow_modify scope. Takes in a String which should be a Spotify Playlist ID. Returns a 204 response on success, if the user successfully follows the given playlist. Returns a 403 response if you do not have permission.

42. getUsersFollowedArtists(refresh_token, client_id, client_secret, num=20):
    NOTE: Requires the user_follow_modify scope. Optionally takes an Integer value as the number of artists to return (default 20). Returns a CursorBasedPagingObject containing a list of Artist Objects corresponding with the user who gave their refresh token's followed artists.

43. unfollowArtistOrUsers(refresh_token, client_id, client_secret, type, ids):
    NOTE: Requires the user_follow_modify scope. Takes in a String which should be the type of id to check, either "artist" or "user". Also takes a String which should be a comma-separated list of ids with no spaces. Returns a 204 response on success, if the user successfully unfollows the given artists or ids. Returns a 403 response if you do not have permission.

44. unfollowAPlaylist(refresh_token, client_id, client_secret, playlist_id):
    NOTE: Requires the playlist_modify_public and playlist_modify_private scopes. Takes in a String which should be a Spotify Playlist ID. Returns a 204 response on success, if the user successfully follows the given playlist. Returns a 403 response if you do not have permission.

Artist API

45. getArtist(refresh_token, client_id, client_secret, artist_id):
    Takes in an artist ID as a String. Returns an Artist Object associated with the given ID.

46. getManyArtists(refresh_token, client_id, client_secret, list_of_ids):
    Takes in a String which should be a comma-separated list of artist IDs with no spaces. Returns a Python List of Artist Objects associated with the given IDs.

47. getArtistsAlbums(refresh_token, client_id, client_secret, artist_id):
    Takes in an artist ID as a String. Returns a Python List of Album Objects associated with the given artist.

48. getArtistsTopTracks(refresh_token, client_id, client_secret, artist_id, country):
    Takes in two Strings, once of which should be an Artist ID and an ISO 3166 alpha-2 country code for a country. Returns a Python List of Track Objects (up to 10) which are the top tracks for that artist in the given country.

49. getArtistRelatedArtists(refresh_token, client_id, client_secret, artist_id):
    Takes in an artist ID as a String. Returns a Python List of Artist Objects which are associated with the given artist according to Spotify.

Search API

50. searchForAnItem(refresh_token, client_id, client_secret, q, type):
    Takes in two Strings, one of which should be a query, and another which

should be the type of item to search for, which could be: “track”, “album”, “artist”, or “playlist”. Returns a PagingObject which contains a List of Track Objects (if type=”track”) or a Dictionary for all other types requested that is of the ({name:id structure}). If there is nothing found, instead returns a String saying “No Results Found.”

Personalization API

51. getUsersTopArtistsandTracks(refresh_token, client_id, client_secret, type, num=20, start=0):
NOTE: Requires the user_top_read scope. Takes in a string which should be the type of item to return, either “artists” or “tracks”. Optionally, can also be given two Integers for the number of items to return (default 20) and the position of the first item to start at (default 0). Returns a Paging object containing a list of Artist Objects (if type=”artists”) or Track Objects (if type=”tracks”), associated with the user’s top Artists or Tracks.