ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Emosic

Report

Spotify API

GVHD: Nguyễn Trần Quang Minh
Sinh viên: Nguyễn Trần Quang Minh - 1811083

Thành phố Hồ Chí Minh, Ngày 29 tháng 9 năm 2021

Mục lục

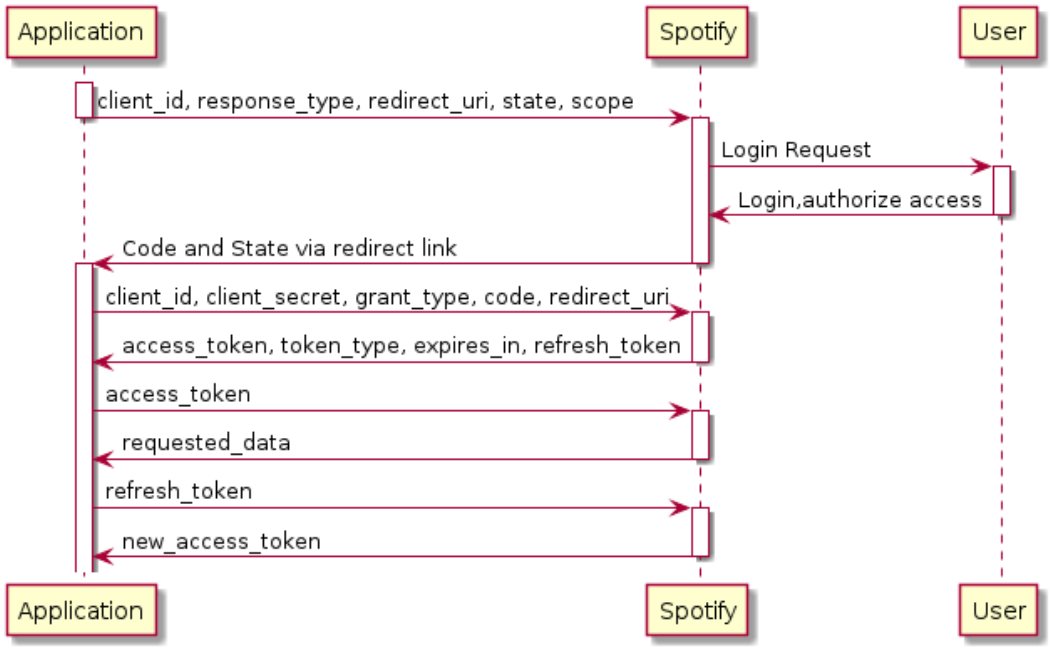
Danh sách hình vẽ

Danh sách bảng

1 Authorization flow:

1.1 Authorization code:

Using the client ID and cient secret, we can request for an authorization code which can be then, exchanged for an access token and a refresh token (can be used to refresh access token after expiration). This flow is described in Hardt 2012 as OAuth 2.0 Authorization Framework



Hình 1: Code Authorization flow

First we sent a request to https://accounts.spotify.com/authorize with the parameters: client_id, response_type, redirect_uri, state, scope.

The redirect_uri is a url for Spotify to redirect user after prompting login and getting authorization. We can set up a custom protocol handler to reopen app from the redirect_uri. Load the registry for the protocol:

```
HKEY_CLASSES_ROOT
    <protocol_name>
        @<protocol_description>
        "URL Protocol" : ""
        shell
            open
                command
                    @<shell command or path to the exe app>
```

The redirect_uri would be something like <protocol_name>::/<url>?<param>=<value>

The state parameter is a mean to prevent cross-site scripting where attacker inject script into the response login page sent by Spotify server. We can check if the response received was sent from Spotify via the state parameter, which Spotify return along the authorization code.

The scope includes values of scope that Application want to be granted access on. Various scopes are listed for different access needs.

Scope	Description	User	Task Required
ugc-image-upload	Write access to user-provided images.	Upload images to Spotify on your behalf.	Upload custom Playlist Cover image.
playlist-modify-private	Write access to a user's private playlists.	Manage your private playlists.	Follow and unfollow a playlist Add items to a playlist Change playlist's detail Create a playlist Remove items from a playlist Reorder a Playlist's items Replace a Playlist's items Upload custom Playlist Cover image.
playlist-read-private	Read access to user's private playlists.	Access your private playlists.	Check if Users Follow a Playlist Get a List of Current User's Playlists Get a List of a User's Playlists
user-read-private	Read access to user's subscription details	Access your subscription details.	Search for an Item Get Current User's Profile
user-read-playback-state	Read access to a user's player state.	Read your currently playing content and Spotify Connect devices information.	Get a User's Available Devices Get Information About The User's Current Playba Get the User's Currently Playing Track
user-library-modify	Write/delete access to a user's "Your Music" library.	Manage your saved content.	Remove Albums for Current User Remove User's Saved Tracks Remove User's Saved Episodes Save Albums for Current User Save Tracks for User Save Episodes for User

Bảng 1: Authorization Scopes

user-read-playback-position	Read access to a user's playback position in a content.	Read your position in content you have played.	Get an Episodes Get Several Episodes Get a Show Get Several Shows Get a Show's Episodes
user-read-recently-played	Read access to a user's recently played tracks.	Access your recently played items.	Get Current User's Recently Played Tracks
app-remote-control	Remote control playback of Spotify. This scope is currently available to Spotify iOS and Android SDKs.	Communicate with the Spotify app on your device.	iOS SDK Android SDK
user-modify-playback-state	Write access to a user's playback state	Control playback on your Spotify clients and Spotify Connect devices.	Pause a User's Playback Seek To Position In Currently Playing Track Set Repeat Mode On User's Playback Set Volume For User's Playback Skip User's Playback To Next Track Skip User's Playback To Previous Track Start/Resume a User's Playback Toggle Shuffle For User's Playback Transfer a User's Playback Append item to User's Playback Queue
user-read-email	Read access to user's email address.	Get your real email address.	Get Current User's Profile
user-follow-modify	Write/delete access to the list of artists and other users that the user follows.	Manage who you are following.	Follow Artists or Users Unfollow Artists or Users

Bảng 2: Authorization Scopes

playlist-modify-public	Write access to a user's public playlists.	Manage your public playlists.	Follow a Playlist Unfollow a Playlist Add Items to a Playlist Change a Playlist's Details Create a Playlist Remove Items from a Playlist Reorder a Playlist's Items Replace a Playlist's Items Upload a Custom Playlist Cover Image
user-follow-read	Read access to the list of artists and other users that the user follows.	Access your followers and who you are following.	Check if Current User Follows Artists or Users Get User's Followed Artists
user-read-currently-playing	Read access to a user's currently playing content.	Read your currently playing content.	Get the User's Currently Playing Track
playlist-read-collaborative	Include collaborative playlists when requesting a user's playlists.	Access your collaborative playlists.	Get a List of Current User's Playlists Get a List of a User's Playlists
user-library-read	Read access to a user's library.	Access your saved content.	Check User's Saved Albums Check User's Saved Tracks Get Current User's Saved Albums Get a User's Saved Tracks Check User's Saved Episodes Get User's Saved Episodes
streaming	Control playback of a Spotify track. This scope is currently available to the Web Playback SDK. The user must have a Spotify Premium account.	Play content and control playback on your other devices.	Web Playback SDK
user-top-read	Read access to a user's top artists and tracks.	Read your top artists and content.	Get a User's Top Artists and Tracks

Bảng 3: Authorization Scopes

The Spotify API prompt the user to login and authorize access. After getting authorization from user, it brings the user to the redirect URI (might restart the app with the protocol handler above).

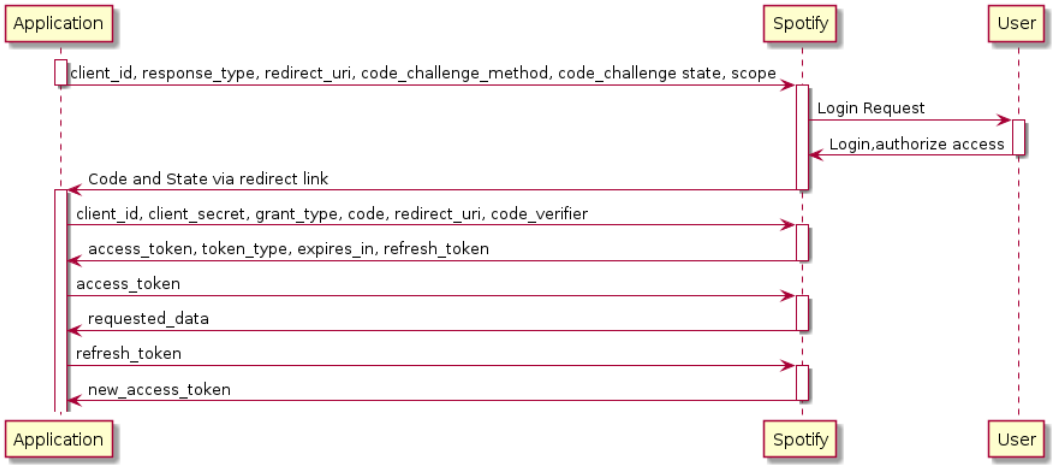
Then the application sent a post request conatining client_id and client_secret, authorization from the previous request and the redirect_uri (must be the same as the redirect_uri registeredon the app's dashboard as well as the redirect_uri sent earlier. The response contain an access token its expiration time, and refresh_token for future access_token refresh.

The remaining process is the same except we have to sent code_verifier which is the same string hashed before along with the Authorization Code.

1.2 Authorizaion code with Proof Key Code Exchange (PKCE):

Pretty much the same as the Authorization Code, but this time we don't want to store the client secret on the local app, so we can create challenge code and compare them.

The challenge code here is a string with length of 43 to 128 characters containing alphanumerical characters, sunderscores, hyphenes, periods, tildes, etc. The string should be then hashed with SHA256 and encoded with base64url

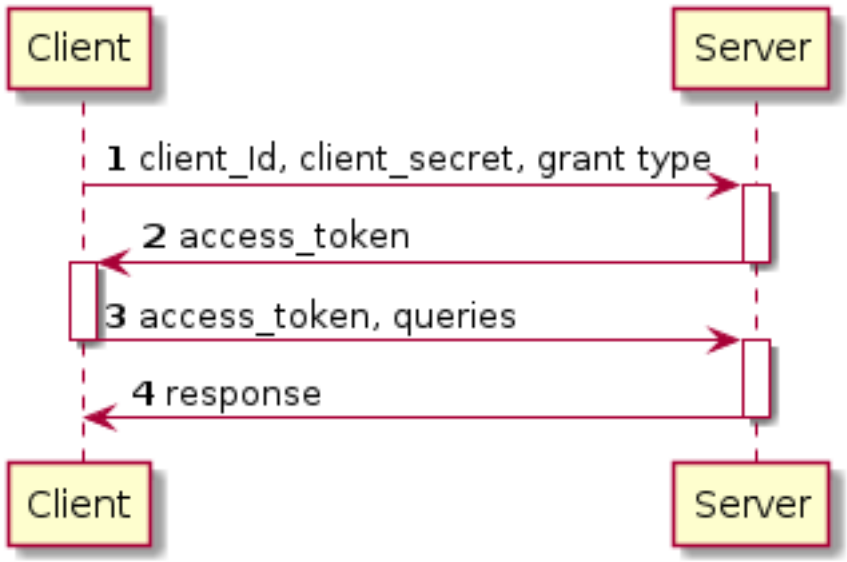


Hình 2: Code Authorization flow with PKCE

1.3 Client Credentials:

Spotify's client credentials authorization flow is meant for app to get authorized access to Spotify API's metadata of music, not including user's private data like playlist or profile information.

The app's client id and client secret is provided in the API's `DASHBOARD` after registering the app.



Hình 3: Client Credential flow

The process is implemented by firstly take the client id and client secret to make a HTTP post request to the url https://accounts.spotify.com/api/token

The structure of the request is as follow:

Header	'Accept'	'application/json'	optional
	'Content-Type	'application/json'	optional
	'Authorization'	'Basic '+ (Base64) <client_id>:<client_secret>	required
Body	'grant_type'	'client_credentials'	required

Bảng 4: Client credential authorization flow

Example:

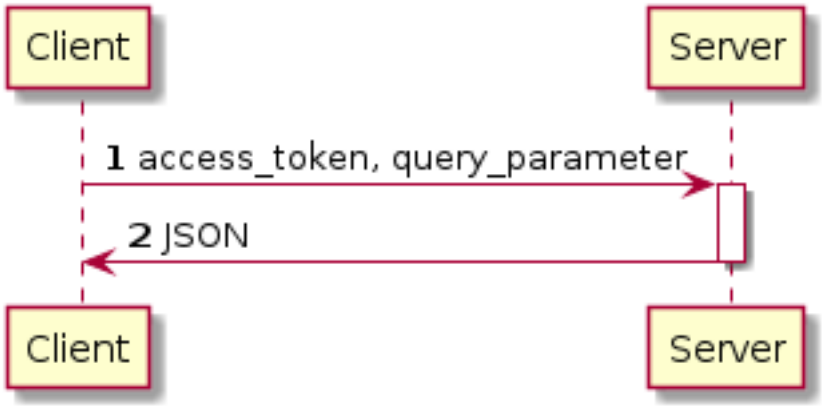
```
Header = {"Authorization": "Basic␣abcnajskaklsnsoansdknlsad:
    asnkdnakdsspofjaosdasdfgag"}
Body = {"grant_type" : "client_credentials"}
requests.post(url="https://accounts.spotify.com/api/token", data=Body, headers=
    Header)
```

2 Example request:

2.1 Request for a track information/audio features/audio analysis:

Request for metadata from Spotify database is done with the same protocol. We provide the access token acquired from either of the 4 authorization flow (metadata is among the general

scope and requires no specific access scope like user data).



Hình 4: Metadata request

The structure of the HTTP get request is as follow:

URL	Track Information Audio Features Audio Analysis	'https://api.spotify.com/v1/track/<track_id>' 'https://api.spotify.com/v1/audio-features/<track_id>' 'https://api.spotify.com/v1/audio-analysis/<track_id>'	send to one of these
Header	'Accept' 'Content-Type 'Authorization'	'application/json' 'application/json' 'Bearer '+ (Base64) <access_token>	optional optional required

Bảng 5: Request metadata from Spotify API

Example:

```
Header = {"Authorization": "Bearer␣abcnajsasnkdnakdsspofjaosdasdfgag"}
requests.post(url="https://api.spotify.com/v1/track/dfsakjdbkasdbkasdk",
    headers=Header)
```

Returned response contain a JSON object which is the data we need.

Example:

Track information:

```
{'album':
   {'album_type': 'album',
    'artists': [....],
    'external_urls': {...},
    'href': ...,
    'id': '...',
    'images': [...],
    'name': "...",
```

```
      'release_date': '...',
      'release_date_precision': '...',
      'total_tracks': ..,
      'type': '...',
      'uri': '..'
     },
  'artists': [...],
  'disc_number': ...,
  'duration_ms': ...,
  'explicit': ...,
  'external_ids': {'isrc': '...'},
  'external_urls': {...},
  'href': '...',
  'id': '...',
  'is_local': ..,
  'is_playable': ...,
  'name': "...",
  'popularity': ...,
  'preview_url':'...',
  'track_number': ...,
  'type': 'track',
  'uri': '...'}
```

2.2 Search for data points:

Much like the previous functions with HTTP get request.

The structure of the HTTP get request is as follow:

URL	https://api.spotify.com/v1/search?q=<key_words>&type=<types>&market=<market>&limit=<limit>&offset=<offset>		
Header	'Accept'	'application/json'	optional
	'Content-Type	'application/json'	optional
	'Authorization'	'Bearer '+ (Base64) <access_token>	required

Bảng 6: Request metadata from Spotify API

The type in the url is a list of types of entries we want to search (track, album, artist, etc) separated by a comma (%2C), market is by default VN, which means the data we can search is available in Vietnamese market. The limit is the maximum number of entries we want to get, must be larger than 0 and no more than 50. The offset sets the first entry to be return (can be 51 and larger to get more than just 50 entries over multiple queries).

Example:

```
Header = {"Authorization": "Bearer␣abcnajsasnkdnakdsspofjaosdasdfgag"}
requests.post(url="https://api.spotify.com/v1/track/dfsakjdbkasdbkasdk",
    headers=Header)
```

Returned response contain a JSON object which is the data we need. The data we need are put in a list of tuple in the value of key 'items' in the value of key 'tracks'

The example is kind of long, and sample return can be found in the returnsample folder.

Tài liệu tham khảo

Hardt, D. (Oct. 2012). *The OAuth 2.0 Authorization Framework*. RFC 6749. http://www.rfc-editor.org/rfc/rfc6749.txt. RFC Editor. URL: http://www.rfc-editor.org/rfc/rfc6749.txt.