Authorization Guide

To have the end user approve your app for access to their Spotify data and features, or to have your app fetch data from Spotify, you need to authorize your application.

Your app can be authorized by Spotify in two ways:

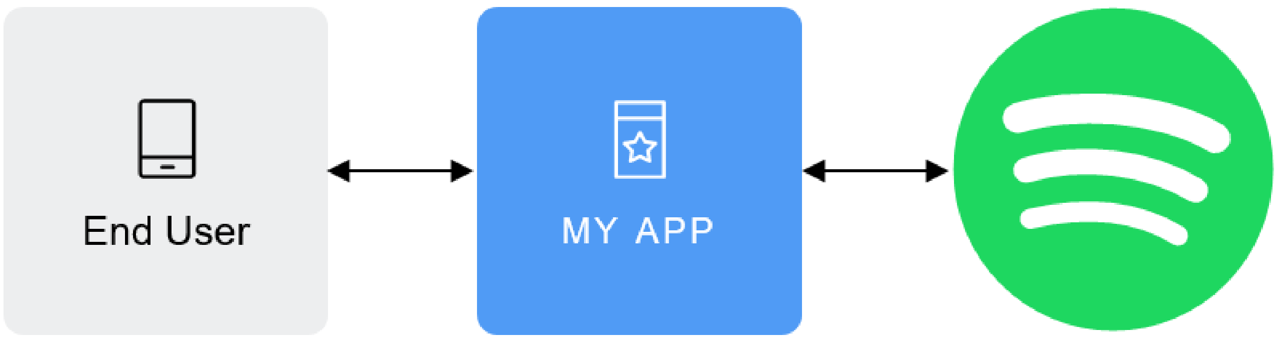
- **App Authorization**: Spotify authorizes your app to access the Spotify Platform (APIs, SDKs and Widgets).
- **User Authorization**: Spotify, as well as the user, grant your app permission to access and/or modify the user's own data. For information about User Authentication, see User Authentication with OAuth 2.0 (https://oauth.net/articles/authentication/). Calls to the Spotify Web API (/documentation/web-api/) require authorization by your application user. To get that authorization, your application generates a call to the Spotify Accounts Service `/authorize` endpoint, passing along a list of the **scopes** for which access permission is sought.

Obtaining Authorization

Making authorized requests to the Spotify platform requires that you are granted permission to access data.

In accordance with RFC-6749 (http://tools.ietf.org/html/rfc6749#section-4.1), 3 parties are involved in the authorization process:

- Server: the Spotify server
- Client: your application
- Resource: the end user data and controls



Scopes

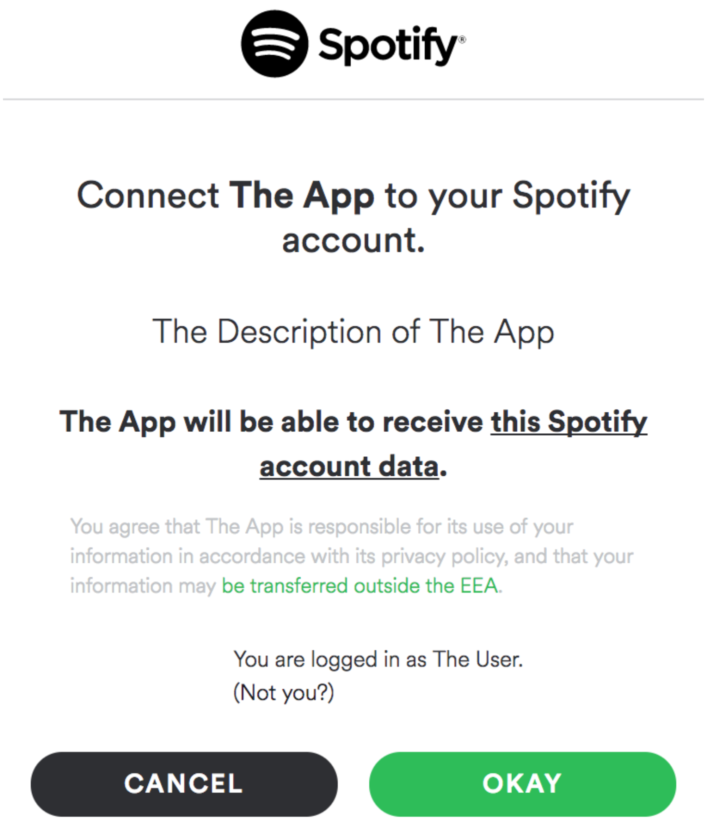
Scopes enable your application to access specific API endpoints on behalf of a user. The set of scopes you pass in your call determines the access permissions that the user is required to grant. See available scopes.

Example

The following code generates a request for the scopes `user-read-private` and `user-read-email` :

```
app.get('/login', function(req, res) {
var scopes = 'user-read-private user-read-email';
res.redirect('https://accounts.spotify.com/authorize' +
  '?response_type=code' +
  '&client_id=' + my_client_id +
  (scopes ? '&scope=' + encodeURIComponent(scopes) : '') +
  '&redirect_uri=' + encodeURIComponent(redirect_uri));
});
```

On execution, the user is redirected to a page where the requested information is presented:



Spotify

Connect **The App** to your Spotify account.

The Description of The App

The App will be able to receive <u>this Spotify account data</u>.

You agree that The App is responsible for its use of your information in accordance with its privacy policy, and that your information may be transferred outside the EEA.

You are logged in as The User.
(Not you?)

To Obtain Authorization:

1. Register your application (/documentation/general/guides/app-settings/#register-your-app).

2. Follow one of the 3 Spotify authorization flows.

Authorization Flows

There are 3 optional flows to obtaining app authorization:

- Refreshable user authorization: **Authorization Code**
- Temporary user authorization: **Implicit Grant**
- Refreshable app authorization: **Client Credentials Flow**

FLOW	ACCESS USER RESOURCES	REQUIRES SECRET KEY (SERVER-SIDE)	ACCESS TOKEN REFRESH
Authorization Code	Yes	Yes	Yes
Client Credentials	No	Yes	No
Implicit Grant	Yes	No	No

For further information and examples of these flows, read our step-by-step tutorial (/documentation/web-api/quick-start/). In addition, see a list of handy wrappers and tools (/documentation/web-api/libraries/#web-api-wrappers) for your language of choice.

Authorization Code Flow

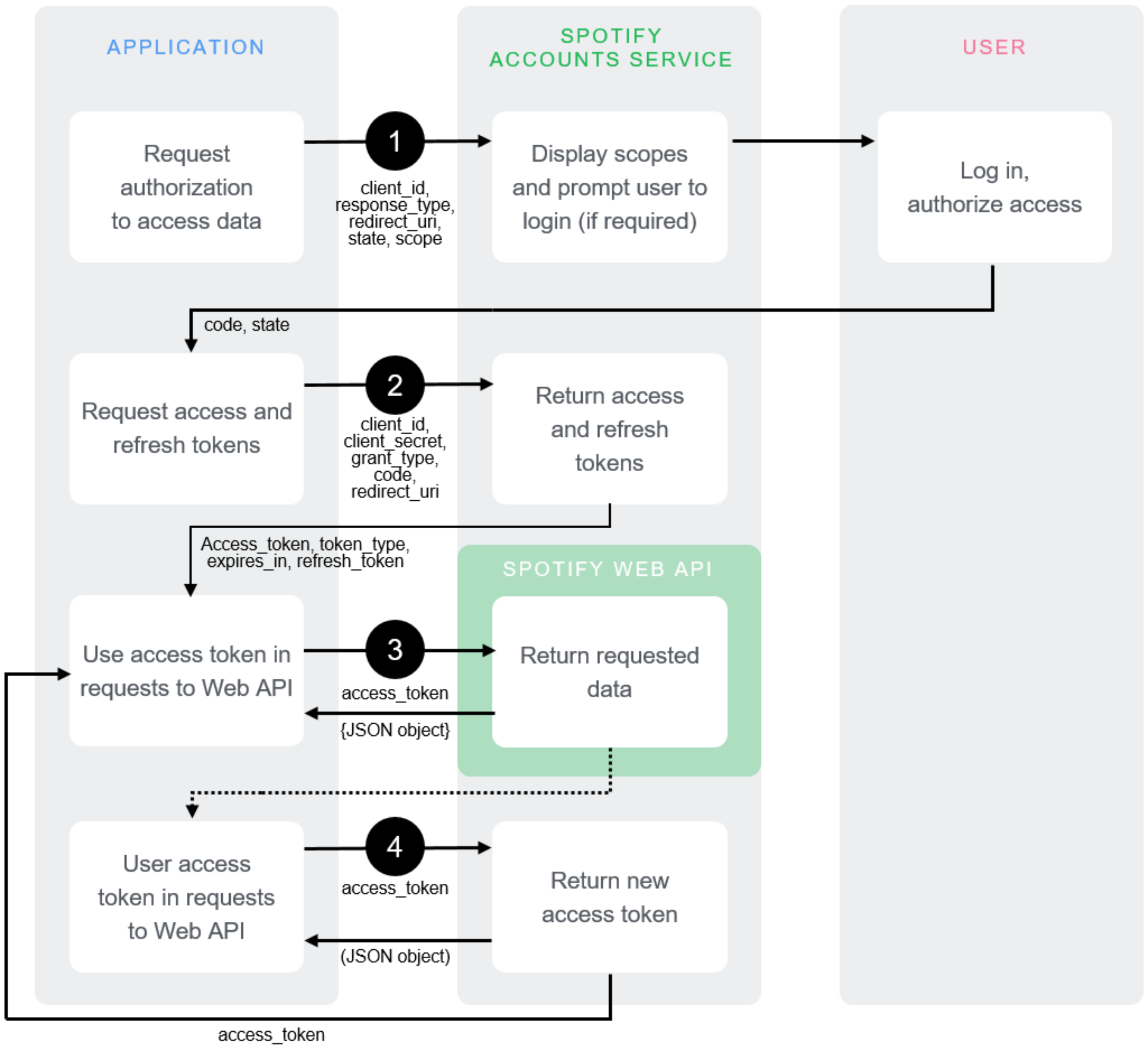
This flow is suitable for long-running applications in which the user grants permission only once. It provides an **access token** that can be *refreshed*. Since the token exchange involves sending your secret key, perform this on a secure location, like a backend service, and not from a client such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

For further information about this flow, see RFC-6749 (http://tools.ietf.org/html/rfc6749#section-4.1), and Web API tutorial (/documentation/web-api/quick-start/).

You do	Prompt your user to a webpage where they can choose to grant you access to their data.
You get	An **access token** and a **refresh token**.

Since the exchange uses your client secret key, to keep the integrity of the key, you should make that request server-side.

The advantage of this flow is that you can use **refresh tokens** to extend the validity of the access token.


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt user to login (if required)
Log in, authorize access
code, state
Request access and refresh tokens
2
client_id, client_secret, grant_type, code, redirect_uri
Return access and refresh tokens
Access_token, token_type, expires_in, refresh_token
SPOTIFY WEB API
Use access token in requests to Web API
3
access_token
{JSON object}
Return requested data
User access token in requests to Web API
4
access_token
(JSON object)
Return new access token
access_token

Authorization Code Flow

1. Have your application request authorization; the user logs in and authorizes access

Your application sends a request to the Spotify Accounts service. The reason your application sends this request may vary:

- A step in the initialization of your application.
- A response to a user action, like a button click.

The `GET` request is sent to the `/authorize` endpoint of the Accounts service:

```
GET https://accounts.spotify.com/authorize
```

QUERY PARAMETER	VALUE
client_id	*Required.* When you register your application, Spotify provides you a Client ID.
response_type	*Required.* Set to `code`.
redirect_uri	*Required.* The URI to redirect to after the user grants or denies permission. This URI needs to have been entered in the Redirect URI whitelist that you specified when you registered your application. The value of `redirect_uri` here must exactly match one of the values you entered when you registered your application, including upper or lowercase, terminating slashes, and such.
state	*Optional, but strongly recommended.* The state can be useful for correlating requests and responses. Because your `redirect_uri` can be guessed, using a state value can increase your assurance that an incoming connection is the result of an authentication request. If you generate a random string, or encode the hash of some client state, such as a cookie, in this state variable, you can validate the response to additionally ensure that both the request and response originated in the same browser. This provides protection against attacks such as cross-site request forgery. See RFC-6749 (http://tools.ietf.org/html/rfc6749#section-4.1).
scope	*Optional.* A space-separated list of scopes.If no scopes are specified, authorization will be granted only to access publicly available information: that is, only information normally visible in the Spotify desktop, web, and mobile players.
show_dialog	*Optional.* Whether or not to force the user to approve the app again if they've already done so. If `false` (default), a user who has already approved the application may be automatically redirected to the URI specified by `redirect_uri`. If `true`, the user will not be automatically redirected and will have to approve the app again.

Example

A typical request is the `GET` request of the `/authorize` endpoint, followed by the query:

```
GET https://accounts.spotify.com/authorize?
client_id=5fe01282e44241328a84e7c5cc169165&response_type=code&redirect_uri=https%3A
%2F%2Fexample.com%2Fcallback&scope=user-read-private%20user-read-
email&state=34fFs29kd09
```

This query performs a couple of things:

1. The user is asked to authorize access within the scopes.

   The Spotify Accounts service presents details of the scopes (/documentation/general/guides/authorization-guide/#list-of-scopes) for which access is being sought.

   - If the user is not logged in, they are prompted to do so using their Spotify credentials.
   - When the user is logged in, they are asked to authorize access to the data sets defined in the scopes.

2. The user is redirected back to your specified `redirect_uri`.

   After the user accepts, or denies your request, the Spotify Accounts service redirects the user back to your `redirect_uri`. In this example, the redirect address is: `https://example.com/callback`

If the user accepts your request, the response query string, for example `https://example.com/callback?code=NApCCg..BkWtQ&state=profile%2Factivity`, contains the following parameters:

QUERY PARAMETER	VALUE
code	An authorization code that can be exchanged for an access token.
state	The value of the `state` parameter supplied in the request.

If the user does not accept your request or an error has occurred, the response query string, for example `https://example.com/callback?error=access_denied&state=STATE`, contains the following parameters:

QUERY PARAMETER	VALUE
error	The reason authorization failed, for example: “access_denied”
state	The value of the `state` parameter supplied in the request.

2. Have your application request refresh and access tokens; Spotify returns access and refresh tokens

When the authorization code has been received, you will need to exchange it with an access token by making a POST request to the Spotify Accounts service, this time to its /api/token endpoint: `POST https://accounts.spotify.com/api/token` The body of this POST request must contain the following parameters encoded in ´*application/x-www-form-urlencoded* as defined in the OAuth 2.0 specification:

REQUEST BODY PARAMETER	VALUE
grant_type	*Required.* As defined in the OAuth 2.0 specification, this field must contain the value `"authorization_code"`.
code	*Required.* The authorization code returned from the initial request to the Account `/authorize endpoint`.
redirect_uri	*Required.* This parameter is used for validation only (there is no actual redirection). The value of this parameter must exactly match the value of `redirect_uri` supplied when requesting the authorization code.

HEADER PARAMETER	VALUE
Authorization	*Required.* Base 64 encoded string that contains the client ID and client secret key. The field must have the format: `Authorization: Basic *<base64 encoded client_id:client_secret>*`

An alternative way to send the client id and secret is as request parameters (`client_id` and `client_secret`) in the POST body, instead of sending them base64-encoded in the header. On success, the response from the Spotify Accounts service has the status code `200` OK in the response header, and the following JSON data in the response body:

KEY	VALUE TYPE	VALUE DESCRIPTION
access_token	string	An access token that can be provided in subsequent calls, for example to Spotify Web API services.

KEY	VALUE TYPE	VALUE DESCRIPTION
token_type	string	How the access token may be used: always "Bearer".
scope	string	A space-separated list of scopes which have been granted for this `access_token`
expires_in	int	The time period (in seconds) for which the access token is valid.
refresh_token	string	A token that can be sent to the Spotify Accounts service in place of an authorization code. (When the access code expires, send a POST request to the Accounts service `/api/token` endpoint, but use this code in place of an authorization code. A new access token will be returned. A new refresh token might be returned too.)

An example cURL (http://en.wikipedia.org/wiki/CURL) request and response from the token endpoint will look something like this:

```
curl -H "Authorization: Basic ZjM...zE=" -d grant_type=authorization_code -d
code=MQCbtKe...44KN -d redirect_uri=https%3A%2F%2Fwww.foo.com%2Fauth
https://accounts.spotify.com/api/token
```

```
{
   "access_token": "NgCXRK...MzYjw",
   "token_type": "Bearer",
   "scope": "user-read-private user-read-email",
   "expires_in": 3600,
   "refresh_token": "NgAagA...Um_SHo"
}
```

3. Use the access token to access the Spotify Web API; Spotify returns requested data

The access token allows you to make requests to the Spotify Web API (/documentation/web-api/) on behalf of a user, for example:

```
curl -H "Authorization: Bearer NgCXRK...MzYjw" https://api.spotify.com/v1/me
```

```
{
  "display_name":"JMWizzler",
  "email":"email@example.com",
  "external_urls":{
  "spotify":"https://open.spotify.com/user/wizzler"
  },
  "href":"https://api.spotify.com/v1/users/wizzler",
  "id":"wizzler",
  "images":[{
  "height":null,
  "url":"https://fbcdn...2330_n.jpg",
  "width":null
  }],
  "product":"premium",
  "type":"user",
  "uri":"spotify:user:wizzler"
}
```

4. Requesting a refreshed access token; Spotify returns a new access token to your app

Access tokens are deliberately set to expire after a short time, after which new tokens may be granted by supplying the refresh token originally obtained during the authorization code exchange.

The request is sent to the token endpoint of the Spotify Accounts service:

```
POST https://accounts.spotify.com/api/token
```

The body of this POST request must contain the following parameters encoded in `application/x-www-form-urlencoded` as defined in the OAuth 2.0 specification:

REQUEST BODY PARAMETER	VALUE
grant_type	*Required.* Set it to `refresh_token` .
refresh_token	*Required.* The refresh token returned from the authorization code exchange.

The header of this POST request must contain the following parameter:

HEADER PARAMETER	VALUE

HEADER PARAMETER	VALUE
Authorization	*Required.* Base 64 encoded string that contains the client ID and client secret key. The field must have the format: `Authorization: Basic <base64 encoded client_id:client_secret>`

Example

```
 curl -H "Authorization: Basic ZjM4Zj...Y0MzE=" -d grant_type=refresh_token -d
refresh_token=NgAagA...NUm_SHo https://accounts.spotify.com/api/token
```

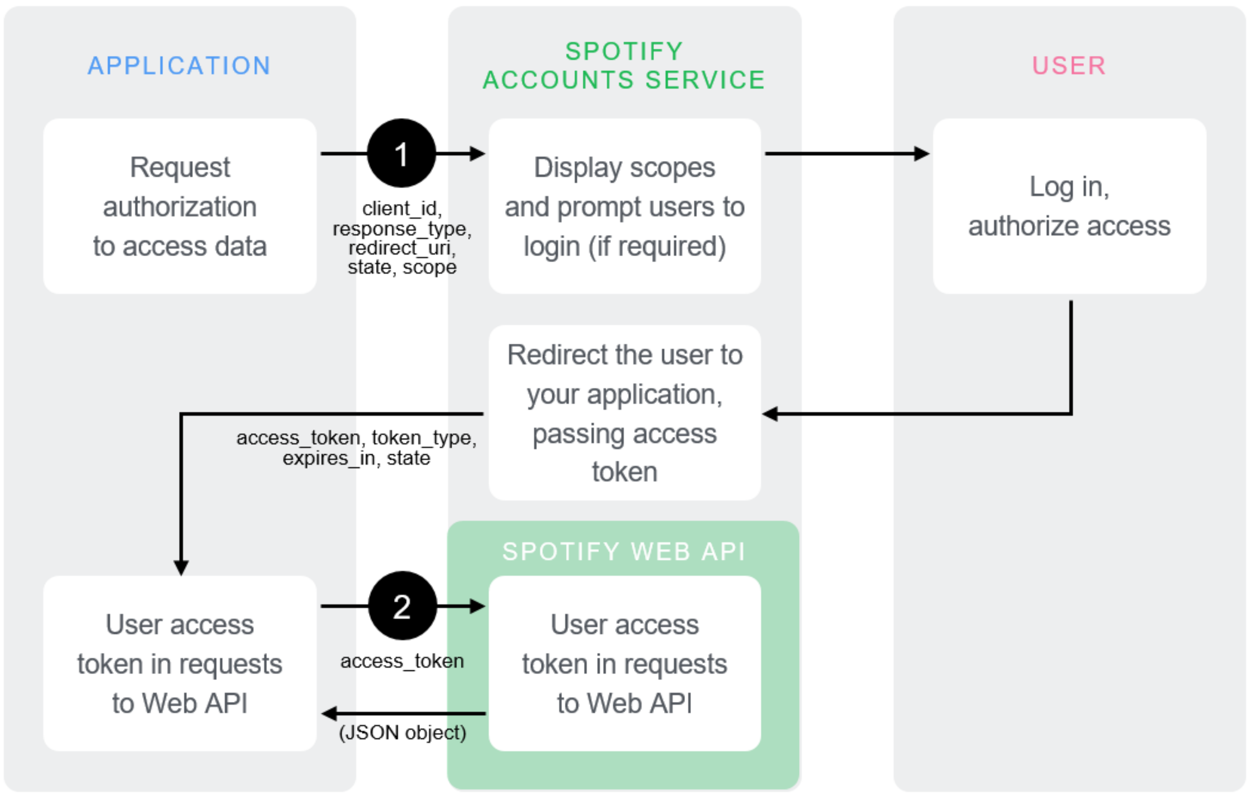
```
{
   "access_token": "NgA6ZcYI...ixn8bUQ",
   "token_type": "Bearer",
   "scope": "user-read-private user-read-email",
   "expires_in": 3600
}
```

Implicit Grant Flow

Implicit grant flow is for clients that are implemented entirely using JavaScript and running in the resource owner's browser. You do not need any server-side code to use it. Rate limits for requests are improved but there is no refresh token provided. This flow is described in RFC-6749 (http://tools.ietf.org/html/rfc6749#section-4.2).

You do	You direct user to Spotify Accounts Service.
You get	**Access Token**.

The Implicit Grant flow is carried out client-side and does not involve secret keys. The access tokens that are issued are short-lived and there are no refresh tokens to extend them when they expire.


APPLICATION
SPOTIFY ACCOUNTS SERVICE
USER
Request authorization to access data
1
client_id, response_type, redirect_uri, state, scope
Display scopes and prompt users to login (if required)
Log in, authorize access
Redirect the user to your application, passing access token
access_token, token_type, expires_in, state
SPOTIFY WEB API
User access token in requests to Web API
2
access_token
(JSON object)
User access token in requests to Web API

Implicit Grant Flow

1. Have your application request authorization

- Redirect the user to the `/authorize` endpoint of the Accounts service:

```
GET https://accounts.spotify.com/authorize
```

The request will include parameters in the query string:

QUERY PARAMETER	VALUE
client_id	*Required.* The client ID provided to you by Spotify when you register your application.
response_type	*Required.* Set it to “token”.
redirect_uri	*Required.* The URI to redirect to after the user grants/denies permission. This URI needs to be entered in the URI whitelist that you specify when you register your application.

QUERY PARAMETER	VALUE
state	*Optional,* but strongly recommended. The state can be useful for correlating requests and responses. Because your `redirect_uri` can be guessed, using a state value can increase your assurance that an incoming connection is the result of an authentication request. If you generate a random string or encode the hash of some client state (e.g., a cookie) in this state variable, you can validate the response to additionally ensure that the request and response originated in the same browser. This provides protection against attacks such as cross-site request forgery. See RFC-6749 (http://tools.ietf.org/html/rfc6749#section-10.12).
scope	*Optional.* A space-separated list of scopes: see Using Scopes.
show_dialog	*Optional.* Whether or not to force the user to approve the app again if they've already done so. If false (default), a user who has already approved the application may be automatically redirected to the URI specified by `redirect_uri`. If true, the user will not be automatically redirected and will have to approve the app again.

Example

You redirect the user:

```
https://accounts.spotify.com/authorize?
client_id=5fe01282e94241328a84e7c5cc169164&redirect_uri=http:%2F%2Fexample.com%2Fca
llback&scope=user-read-private%20user-read-email&response_type=token&state=123
```

This performs a couple of actions:

1. The user is asked to authorize access within the scopes. The Spotify Accounts service presents details of the scopes for which access is being sought.
   - If the user is not logged in, they are prompted to do so using their Spotify username and password.
   - When the user is logged in, they are asked to authorize access to the data sets defined in the scopes.
2. The user is redirected back to your specified URI. After the user grants (or denies) access, the Spotify Accounts service redirects the user to the `redirect_uri`. In this example, the redirect address is: `https://example.com/callback`

If the user grants access, the final URL will contain a **hash fragment** with the following data encoded as a query string. For example:
`https://example.com/callback#access_token=NwAExz...BV3O2Tk&token_type=Bearer&expires_in=3600&state=123`

QUERY PARAMETER	VALUE
access_token	An access token that can be provided in subsequent calls, for example to Spotify Web API services.
token_type	Value: “Bearer”
expires_in	The time period (in seconds) for which the access token is valid.
state	The value of the state parameter supplied in the request.

If the user denies access, access token is not included and the final URL includes a query string `https://example.com/callback?error=access_denied&state=123` , containing the following parameters:

QUERY PARAMETER	VALUE
error	The reason authorization failed, for example: “access_denied”.
state	The value of the state parameter supplied in the request.

2. Use the access token to access the Spotify Web API

The access token allows you to make requests to the Spotify Web API (/documentation/web-api/). For example, if you are using jQuery, you would do:

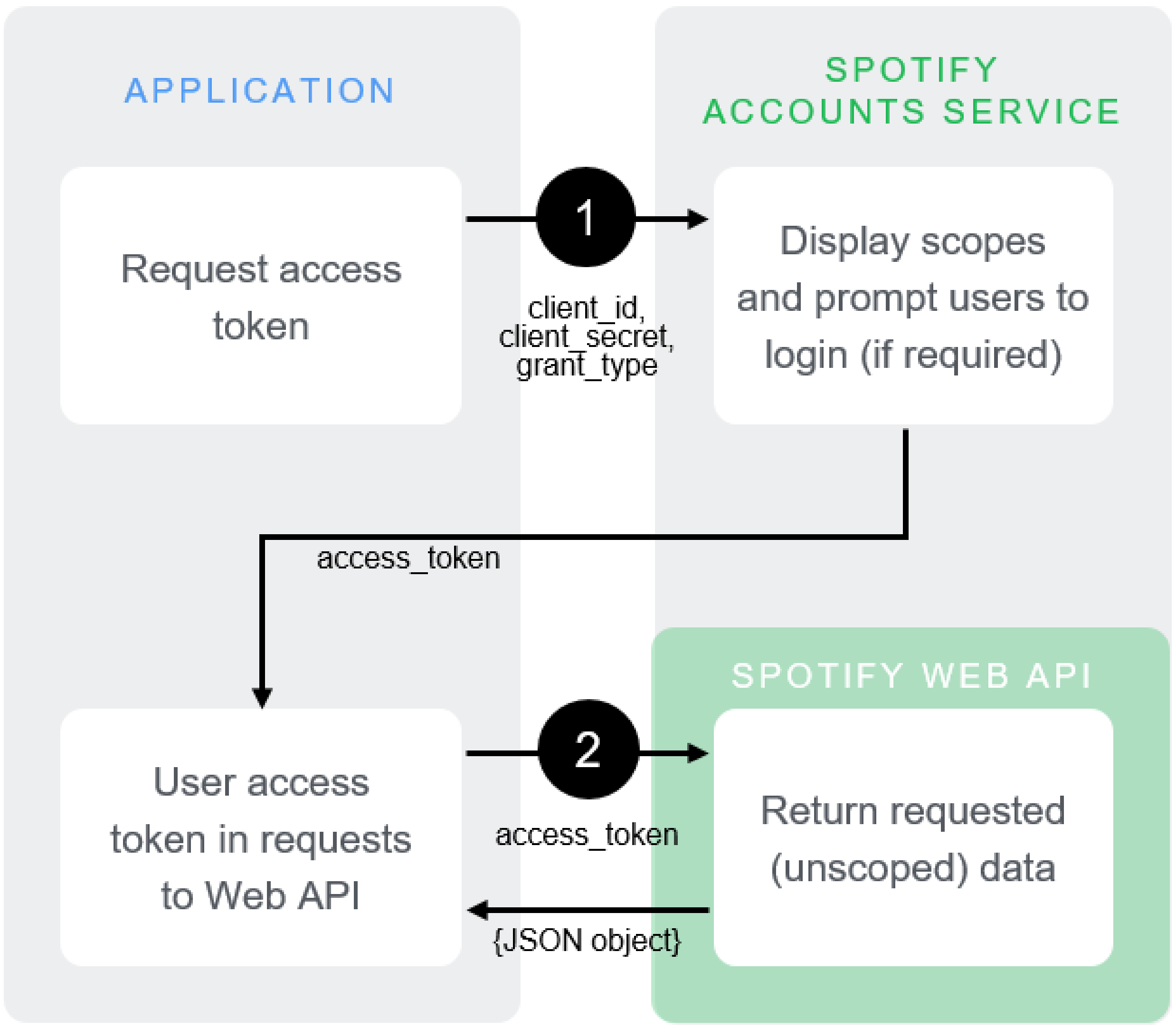
```
$.ajax({
   url: 'https://api.spotify.com/v1/me',
   headers: {
       'Authorization': 'Bearer ' + accessToken
   },
   success: function(response) {
       ...
   }
```

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.

You do	Login with your **Client ID** and **Secret Key**.
You get	**Access token**.

This flow makes it possible to authenticate your requests to the Spotify Web API and to obtain a higher rate limit than you would get without authentication. **Note:** However that this flow does not include **authorization** and therefore cannot be used to access or to manage a user private data. For further information about this flow, see RFC-6749 (http://tools.ietf.org/html/rfc6749#section-4.4).


APPLICATION
SPOTIFY ACCOUNTS SERVICE
Request access token
1
client_id, client_secret, grant_type
Display scopes and prompt users to login (if required)
access_token
SPOTIFY WEB API
User access token in requests to Web API
2
access_token
{JSON object}
Return requested (unscoped) data

Client Credentials Flow

1. Have your application request authorization

The request is sent to the `/api/token` endpoint of the Accounts service:

```
POST https://accounts.spotify.com/api/token
```

The body of this POST request must contain the following parameters encoded in `application/x-www-form-urlencoded` as defined in the OAuth 2.0 specification:

REQUEST BODY PARAMETER	VALUE
grant_type	*Required.* Set it to `client_credentials`.

The header of this POST request must contain the following parameter:

HEADER PARAMETER	VALUE
Authorization	*Required.* Base 64 encoded string that contains the client ID and client secret key. The field must have the format: Authorization: `Basic <base64 encoded client_id:client_secret>`

Example

```
curl -X "POST" -H "Authorization: Basic ZjM4ZjAw...WY0MzE=" -d
grant_type=client_credentials https://accounts.spotify.com/api/token
```

```
  {
     "access_token": "NgCXRKc...MzYjw",
     "token_type": "bearer",
     "expires_in": 3600,
  }
```

2. Use the access token to access the Spotify Web API

The access token allows you to make requests to the Spotify Web API endpoints that do *not* require user authorization such as the Get a track (/documentation/web-api/reference/tracks/get-track/) endpoint, for example:

```
 curl -H "Authorization: Bearer NgCXRKc...MzYjw"
https://api.spotify.com/v1/tracks/2TpxZ7JUBn3uw46aR7qd6V
```

```
  {
    "album" : {
    "album_type" : "album",
    "external_urls" : {
      "spotify" :
  "https://open.spotify.com/album/6akEvsycLGftJxYudPjmqK"
    },
    "href" : "https://api.spotify.com/v1/albums/6akEvsycLGftJxYudPjmqK",
    "id" : "6akEvsycLGftJxYudPjmqK",
    "images" : [ {
      "height" : 640,
      "url" :
  "https://i.scdn.co/image/f2798ddab0c7b76dc2d270b65c4f67ddef7f6718",
      "width" : 640
    }, {
  ...
```

List of scopes

The full list of scopes is in the Authorization Scopes (/documentation/general/guides/scopes/) page. Alternatively, each endpoint reference (/documentation/web-api/reference/) page contains the necessary scope required to perform a particular action.

Frequently Asked Questions

Accessing your data without showing a login form

I want to interact with the web API and show some data on my website. I see that the endpoints I require authorization, but I don't need/want a login window to pop-up, because I want to grant my own app access to my own playlists once. Is there any way of doing this?

You basically need an access token and a refresh token issued for your user account. To obtain a pair of access token - refresh token, follow the Authorization Code Flow (if you need a certain scope to be approved) or Client Credentials (if you just need to sign your request, like when fetching a certain playlist). Once you obtain them, you can use your access token and refresh it when it expires without having to show any login form.

I want to create a quick script to add a new song every day to my playlist. Is there a way I can do this without having to open the browser and log in every day? I could set my user and password in the script.

The Spotify Web API does not support authorization through username and password. For this use case you would obtain an access token through the Authorization code. See the response above.

Is there any way to override the HTTP verb such as sending a method=delete query parameter in a GET request?

The Web API does not support method override at the moment. If you want to consume the API from IE9 and below, using XDomainRequest, which does not support custom headers, you will need to proxy those requests or make them server-side.

DOCS (/DOCUMENTATION/)

General (/documentation/general/guides/)

Web API (/documentation/web-api/)

Web Playback SDK (/documentation/web-playback-sdk/)

iOS (/documentation/ios/)

Android (/documentation/android/)

Widgets (/documentation/widgets/)

COMMUNITY (/COMMUNITY/NEWS/)

News (/community/news/)

Showcase (/community/showcase/)

USE CASES (/USE-CASES)

Mobile Apps (/use-cases/mobile-apps/)

Hardware (/use-cases/hardware/)

SUPPORT (HTTPS://DEVELOPER.SPOTIFY.COM/SUPPORT/)

DISCOVER (/DISCOVER/)

CONSOLE (/CONSOLE/)

DASHBOARD (/DASHBOARD/)

USE CASES (/USE-CASES)

DESIGN & BRANDING GUIDELINES (/BRANDING-GUIDELINES)

LEGAL (/TERMS/)

Terms of Service (/terms/)

Third Party Licenses (/legal/third-party-licenses/)

Legal (https://www.spotify.com/legal/)

Cookies (https://www.spotify.com/legal/privacy-policy/#cookies)