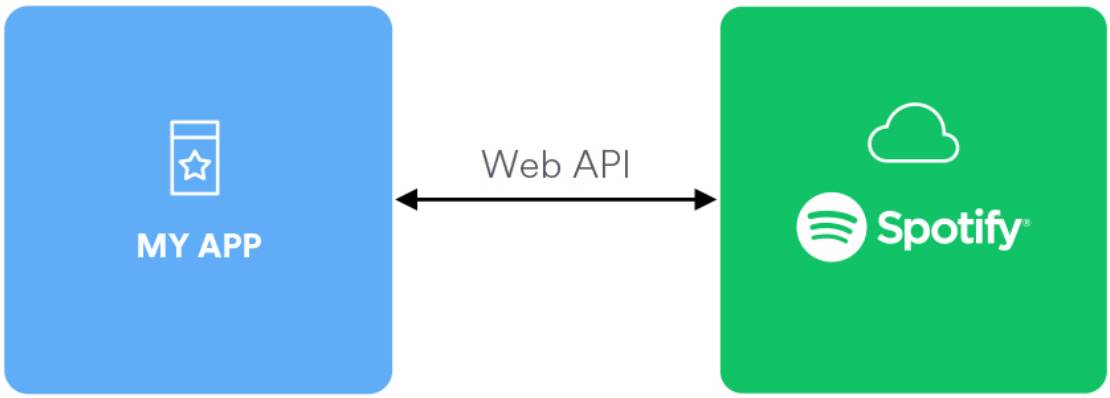
Web API

Note: By using Spotify developer tools, you accept the Spotify Developer Terms of Service (/terms/).

Based on simple REST principles, the Spotify Web API endpoints return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalogue.



Web API also provides access to user related data, like playlists and music that the user saves in the **Your Music** library. Such access is enabled through selective authorization, by the user.

The base address of Web API is https://api.spotify.com (https://api.spotify.com). The API provides a set of endpoints (/documentation/web-api/reference/), each with its own unique path. To access private data through the Web API, such as user profiles and playlists, an application must get the user's permission to access the data. Authorization (/documentation/general/guides/authorization-guide/) is via the Spotify Accounts service (https://accounts.spotify.com).

Requests

The Spotify Web API is based on REST (http://en.wikipedia.org/wiki/Representational_state_transfer) principles. Data resources are accessed via standard HTTPS requests in UTF-8 format to an API endpoint. Where possible, Web API uses appropriate HTTP verbs for each action:

METHOD	ACTION
GET	Retrieves resources
POST	Creates resources
PUT	Changes and/or replaces resources or collections
DELETE	Deletes resources

Spotify URIs and IDs

In requests to the Web API and responses from it, you will frequently encounter the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
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. To find a Spotify URI simply right-click (on Windows) or Ctrl-Click (on a Mac) on the artist's or album's or track's name.	`spotify:track:6rqhFgbbKwnb9MLmUQDhG6`
Spotify ID	The base-62 identifier that you can find at the end of the Spotify URI (see above) for an artist, track, album, playlist, etc. Unlike a Spotify URI, a Spotify ID does not clearly identify the type of resource; that information is provided elsewhere in the call.	`6rqhFgbbKwnb9MLmUQDhG6`
Spotify category ID	The unique string identifying the Spotify category.	`party`
Spotify user ID	The unique string identifying the Spotify user that you can find at the end of the Spotify URI for the user. The ID of the current user can be obtained via the Web API endpoint (/documentation/web-api/reference/).	`wizzler`
Spotify URL	An HTML link that opens a track, album, app, playlist or other Spotify resource in a Spotify client (which client is determined by the user's device and account settings at play.spotify.com (https://play.spotify.com/).	`http://open.spotify.com/track/6rqhFgbbKwnb9MLmUQDhG6`

Rate Limiting

Rate Limiting enables Web API to share access bandwidth to its resources equally across all users.

Rate limiting is applied as per application based on Client ID, and regardless of the number of users who use the application simultaneously.

To reduce the amount of requests, use endpoints that fetch multiple entities in one request. For example: If you often request single tracks, albums, or artists, use endpoints such as Get Several Tracks (/documentation/web-api/reference/tracks/get-several-tracks/), Get Several Albums (/documentation/web-api/reference/albums/get-several-albums/) or Get Several Artists (/documentation/web-api/reference/artists/get-several-artists/), instead.

Note: If Web API returns **status code 429**, it means that you have sent too many requests. When this happens, check the `Retry-After` header, where you will see a number displayed. This is the number of *seconds that you need to wait*, before you try your request again.

Responses

Web API returns all response data as a JSON object. See the Web API Object Model (/documentation/web-api/reference/object-model/) for a description of all the retrievable objects.

Timestamps

Timestamps are returned in ISO 8601 (http://en.wikipedia.org/wiki/ISO_8601) format as Coordinated Universal Time (UTC) (http://en.wikipedia.org/wiki/Offset_to_Coordinated_Universal_Time) with a zero offset: `YYYY-MM-DDTHH:MM:SSZ`. If the time is imprecise (for example, the date/time of an album release), an additional field indicates the precision; see for example, `release_date` in an album object (/documentation/web-api/reference/object-model/#album-object-full).

Pagination

Some endpoints support a way of paging the dataset, taking an offset and limit as query parameters:

```
$ curl
https://api.spotify.com/v1/artists/1vCWHaC5f2uS3yhpwWbIA6/albums?
album_type=SINGLE&offset=20&limit=10
```

In this example, in a list of 50 (`total`) singles by the specified artist : From the twentieth (`offset`) single, retrieve the next 10 (`limit`) singles.

Note: The offset numbering is zero-based. Omitting the `offset` parameter returns the first X elements. Check the documentation for the specific endpoint and verify the default `limit` value. Requests that return an array of items are automatically paginated if the number of items vary. For example, tracks in a playlist. In such case, the results are returned within a paging object (/documentation/web-api/reference/object-model/#paging-object).

Conditional Requests

Most API responses contain appropriate cache-control headers set to assist in client-side caching:

- If you have cached a response, do not request it again until the response has expired.
- If the response contains an ETag, set the If-None-Match request header to the ETag value.
- If the response has not changed, the Spotify service responds quickly with **304 Not Modified** status, meaning that your cached version is still good and your application should use it.

Note: To target changes to a particular historical playlist version and have those changes rolled through to the latest version, use playlist endpoints that also return a snapshot-id. For further information, see Working With Playlists (/documentation/general/guides/working-with-playlists/).

Response Status Codes

Web API uses the following response status codes, as defined in the RFC 2616 (https://www.ietf.org/rfc/rfc2616.txt) and RFC 6585 (https://www.ietf.org/rfc/rfc6585.txt):

STATUS CODE	DESCRIPTION
200	OK - The request has succeeded. The client can read the result of the request in the body and the headers of the response.

STATUS CODE	DESCRIPTION
201	Created - The request has been fulfilled and resulted in a new resource being created.
202	Accepted - The request has been accepted for processing, but the processing has not been completed.
204	No Content - The request has succeeded but returns no message body.
304	Not Modified. See Conditional requests.
400	Bad Request - The request could not be understood by the server due to malformed syntax. The message body will contain more information; see Response Schema.
401	Unauthorized - The request requires user authentication or, if the request included authorization credentials, authorization has been refused for those credentials.
403	Forbidden - The server understood the request, but is refusing to fulfill it.
404	Not Found - The requested resource could not be found. This error can be due to a temporary or permanent condition.
429	Too Many Requests - Rate limiting has been applied.
500	Internal Server Error. You should never receive this error because our clever coders catch them all ... but if you are unlucky enough to get one, please report it to us through a comment at the bottom of this page.
502	Bad Gateway - The server was acting as a gateway or proxy and received an invalid response from the upstream server.
503	Service Unavailable - The server is currently unable to handle the request due to a temporary condition which will be alleviated after some delay. You can choose to resend the request again.

Response Schema

Web API uses two different formats to describe an error:

- Authentication Error Object
- Regular Error Object

Authentication Error Object

Whenever the application makes requests related to authentication or authorization to Web API, such as retrieving an access token or refreshing an access token, the error response follows RFC 6749 (https://tools.ietf.org/html/rfc6749) on the OAuth 2.0 Authorization Framework.

KEY	VALUE TYPE	VALUE DESCRIPTION
error	string	A high level description of the error as specified in RFC 6749 Section 5.2 (https://tools.ietf.org/html/rfc6749#section-5.2).
error_description	string	A more detailed description of the error as specified in RFC 6749 Section 4.1.2.1 (https://tools.ietf.org/html/rfc6749#section-4.1.2.1).

Here is an example of a failing request to refresh an access token.

```
$ curl -H "Authorization: Basic Yjc...cK" -d grant_type=refresh_token
-d refresh_token=AQD...f0 "https://accounts.spotify.com/api/token"

{
    "error": "invalid_client",
    "error_description": "Invalid client secret"
}
```

Regular Error Object

Apart from the response code, unsuccessful responses return a JSON object containing the following information:

KEY	VALUE TYPE	VALUE DESCRIPTION
status	integer	The HTTP status code that is also returned in the response header. For further information, see Response Status Codes.
message	string	A short description of the cause of the error.

Here, for example is the error that occurs when trying to fetch information for a non-existent track:

```
$ curl -i "https://api.spotify.com/v1/tracks/2KrxsD86ARO5beq7Q0Drfqa"

HTTP/1.1 400 Bad Request
{
    "error": {
        "status": 400,
        "message": "invalid id"
    }
}
```

Authentication

All requests to Web API require authentication. This is achieved by sending a valid OAuth access token in the request header. For more information about these authentication methods, see the Web API Authorization Guide (/documentation/general/guides/authorization-guide/).

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/)