# SoundCloud API

SoundCloud is an open source audio streaming service that allows users to upload, share and promote their audio content to users globally. This documentation provides a brief overview of some of the SoundCloud APIs  and its python SDK.

***Note: This is a sample revised version of the existing SoundCloud API documentation. You will find the sample Python SDK for SoundCloud at the end of the document.***

## Authentication

SoundCloud provides a RESTful API service that allows  your web application to integrate with its platform. To get started, you should register your application with SoundCloud by creating an account. Refer to the instructions in Sign in with SoundCloud.

SoundCloud uses OAuth 2.1 to authorize your application with a stateless transaction where the information regarding the user credentials are not stored. To know more about OAuth setup and authentication refer to Authentication Code Flow.

## SoundCloud APIs

The SoundCloud APIs are accessible through REST-ful syntax from HTTPS calls. The base URL for all requests is:

```
https://api.soundcloud.com
```

| **Service endpoints** | |
|---|---|
| `tracks` | Uploads and manages new tracks or an audio file. |
| | POST/tracks |
| | GET/tracks{track_id} |
| | DELETE/tracks{track_id} |
| | PUT/tracks{track_id} |
| `search` | Finds the required track based on the query |
| | GET/playlists |

| | |
|---|---|
| | GET/users |

# REST Resources

## tracks

## Post tracks

```
POST /tracks
```

Upload a song or audio file or a track. The supported audio formats are `AIFF`, `WAV`, `FLAC`, `OGG`, `MP2`, `MP3`, `AAC`, `AMR` and `WMA`. The maximum size is 500MB per audio file. To upload a track, send a POST request with a multipart/form-data media type to the /tracks endpoint.

### PATH Parameters

There are no path parameters for uploading an audio file.

### Request

| Name | Data Type | Required | Description |
|---|---|---|---|
| multipart/form-data | string | Yes | Content type of the track or audio file including some metadata. |
| track[asset_data] | Binary string | Yes | Audio file |

### Example

```
$ curl -X POST "https://api.soundcloud.com/tracks" \
       -H  "accept: application/json; charset=utf-8" \
       -H "Authorization: OAuth ACCESS_TOKEN" \
       -H "Content-Type: multipart/form-data" \
       -F "track[title]=YOUR_TITLE" \
       -F "track[purchase_url]: null" \
```

```
      -F "track[label_name]: some label" \
      -F "track[streamable]: true" \
      -F "track[asset_data]=@JF5_IDL_DZ_C2_M3.wav;type=audio/wav' \
      -F "downloadable": true
```

## Fields

| Name | Data Type | Required | Description |
|---|---|---|---|
| title | string | Yes | Name of the track |
| purchase_url | string | No | Link where the track was purchased |
| label_name | string | No | Name of the song album |
| streamable | boolean | Yes | Required values:<br>Yes: If the song is streamable<br>No: If the song can't be streamed |
| downloadable | boolean | No | Values:<br>Yes: If the song can be downloaded<br>No: if the song can't be downloaded. |

## JSON representation

```
{
"title": YOUR_TITLE,
"purchase_url": null,
"label_name": "some label",
"streamable": true,
"downloadable": true
}
```

## HTTP Response status

| Code | Message |
| --- | --- |
| 201 | Success(OK) |
| 400 | Bad Request |
| 401 | Unauthorized |
| 422 | Unprocessable Entity |

## Get tracks

```
GET /tracks/{track_id}
```

Returns a track. You should specify the track ID along with a token if you need to fetch your private tracks or playlists.

### PATH Parameters

| Name | Data Type | Required | Description |
| --- | --- | --- | --- |
| track_id | string | Yes | The ID of the track |
| secret_token | string | Optional | A secret token to fetch private playlist or tracks |

### Example

```
$ curl -X GET
"https://api.soundcloud.com/tracks/308946187?secret_token=1234"\
       -H  "accept: application/json; charset=utf-8" \
```

### JSON Representation

```
{
"id": 1234,
"sharing": "public",
"label_name": "some label",
"description": "Your sample description of the track"
}
```

### HTTP Response status

| Code | Message |
|---|---|
| 200 | Success(OK) |
| 404 | Not Found |
| 401 | Unauthorized |

## Delete tracks

```
DELETE /tracks/{track_id}
```

Deletes a track.

### PATH Parameters

| Name | Data Type | Required | Description |
|---|---|---|---|
| track_id | integer | Yes | ID of the audio file. |

### Example

```
$ curl -X 'DELETE' \
  'https://api.soundcloud.com/tracks/308946187' \
  -H 'accept: application/json; charset=utf-8'
```

### JSON Representation

```
{
  "code": 401,
  "message": "",
  "link": "https://developers.soundcloud.com/docs/api/explorer/open-api",
  "status": "401 - Unauthorized",
  "errors": [],
  "error": null
}
```

### HTTP Response status

| Code | Message |
| --- | --- |
| 200 | Success |
| 401 | Unauthorized |
| 404 | Not Found |

## Put tracks

Updates the track's information.

```
PUT /tracks/{track_id}
```

### PATH Parameters

| Name | Data Type | Required | Description |
| --- | --- | --- | --- |
| track_id | integer | Yes | ID of the audio file |

### Example

```
$ curl -X 'PUT' \
  'https://api.soundcloud.com/tracks/308946187' \
  -H 'accept: application/json; charset=utf-8' \
  -H 'Content-Type: multipart/form-data' \
  -F 'track[description]=Taylor Swift' \
  -F 'track[sharing]=public' \
  -F 'track[streamable]=true' \
  -F 'track[commentable]=true' \
  -F 'track[title]=Fortnight'
```

### JSON Representation

```
{
   "commentable": true,
   "description": Taylor Swift,
   "sharing": "public",
   "streamable": true,
   "title": "Fortnight",
   "user": {
    "$ref": "#/components/examples/User/value"
  },
  "user_favorite": true,
  "user_playback_count": 1,
  "waveform_url": "https://wave.sndcdn.com/someString.png",
  "access": "playable"
}
```

## search

## Get playlists

Performs a playlist search based on a query

```
GET /playlists
```

## PATH Parameters

| Name | Data Type | Required | Description |
|---|---|---|---|
| q | string | Yes | The search query |
| access | array[string] | Yes | Filters content by level of access the user has to the track |
| show_tracks | boolean | Yes | A boolean flag to request a playlist with or without tracks. Default is true. |
| limit | integer | Yes | Number of results to return in the collection. |
| linked_partitioning | boolean | Yes | Returns paginated collection of items (recommended, returning a list without pagination is deprecated and should not be used) |

## Example

```
$ curl -X 'GET' \
'https://api.soundcloud.com/playlists?q=hello&access=playable%2Cpreview&sho
w_tracks=true&limit=2&offset=0&linked_partitioning=true' \
  -H 'accept: application/json; charset=utf-8'
```

## JSON Representation

```
{
  "collection": {
    "$ref": "#/components/examples/Playlist/value"
  },
```

```
  "next_href":
"https://api.soundcloud.com/collection?page_size=10&cursor=1234567"
}
```

# Python SDK

SoundCloud SDK provides a set of libraries or tools to integrate SoundCloud features into your application. Some of the primary methods used for integrating include initialization, uploading, embed and audio and play an audio using the Widget API. The documentation provided in the SoundCloud SDK is in Javascript. This guide provides implementation in Python. Refer to Javascript SDK.

## `initialize` method

The `initialize` method is used to set up and configure your application for handling  user authentication. Refer to the Javascript example here.

Example

```
import requests

def initialize_soundcloud(client_id, redirect_uri):
    return {
        "client_id": client_id,
        "redirect_uri": redirect_uri,
        "auth_url":
"https://soundcloud.com/connect?client_id={client_id}&redirect_uri={redirec
t_uri}&response_type=token"
    }
```

## `upload` method

`upload` method is used to upload audio files and recording. This method requires the user's browser to support FormData. Upload takes longer duration dependending on the file size and so this method also returns the progress of upload. Refer to the Javascript example here.

```
import requests

def upload_track_to_soundcloud(file_path, title):
      # Open the audio file in binary mode
    with open(file_path, "rb") as file:
        files = {
            "track[asset_data]": file  # The audio file
        }
        data = {
            "track[title]": title  # The track title
        }
```