Deezer Android SDK - Getting Started

Contents

1 Overview

This document provides the initial information needed to get a sample application up and running. This will provide the basic knowledge of what the SDK is, what it can do for you and how to use it.

After reading this document, you can then read the more complete User Guide to go further with your Application.

2 Features

The Deezer SDK for Android embeds multiple features from Deezer directly inside third-party applications. Mainly, it enables you to :

- Communicate with the Deezer API, getting access to Deezer's complete music catalog, as well as user informations;
- Authenticate users with their Deezer account (using the Oauth open protocole);
- Easily play music from Deezer's catalog.

3 Installing the SDK

The Deezer SDK for Android contains a Java library, packaged as an AAR archive.

Important
Requirements : Deezer SDK for Android is compatible with any application using Android SDK version 8 (a.k.a. Android Froyo, version 2.2) and above. The library combines Java & native components, built and tested for all supported processor architecture (*armeabi*, *armeabi-v7a*, *x86* and *mips*, as well as their 64 bits counterparts), all stored in the libs folder in the downloadable archive.

3.1 Registering an Android Application

Developers who want to build applications on top of Deezer SDK must register themselves and their applications on Deezer for developers web site, and get an Application ID.

1. Open the My Apps section of the Deezer for developers website;
2. If you don't have an application registered already, click on the "Create a new application" link and fill in the form;
3. Select your created application, then click on "Edit Application";
4. At the bottom of the page, in the "Android Application" section, add the Android package name of your application, and click "Save";
5. Remember your Application ID, as you'll need it to use the SDK.

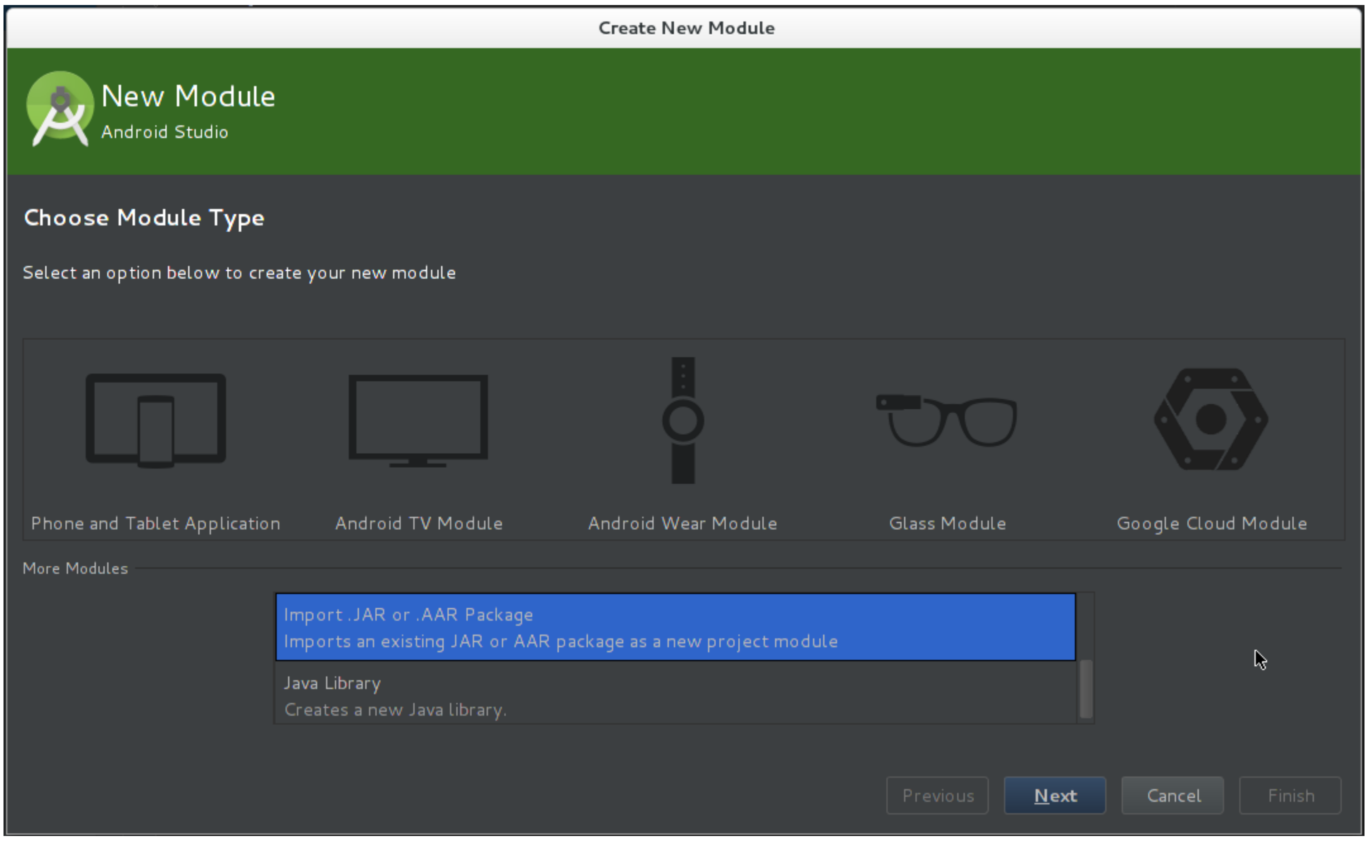
3.2 Adding the library in Eclipse

Warning
Eclipse and the ADT plugin is officially no longer supported by the Android Team. We recommend using Android Studio for any new development, and you can also read Google's guide for Migrating to Android Studio. Including an .AAR dependency is a little bit tricky with Eclipse, but a few solutions can be found online.

3.3 Adding the library in Android Studio

1. Create a new module in your project, selecting "Import .JAR or .AAR package";
2. Browse to find the `deezer-sdk-{sdk_version}.aar` file and choose a name for the module (for instance `deezer-sdk`);
3. Once the module is created, you can add a dependency to the SDK from any `build.gradle` script in your project :

```
dependencies {
    compile project(':deezer-sdk')
}
```

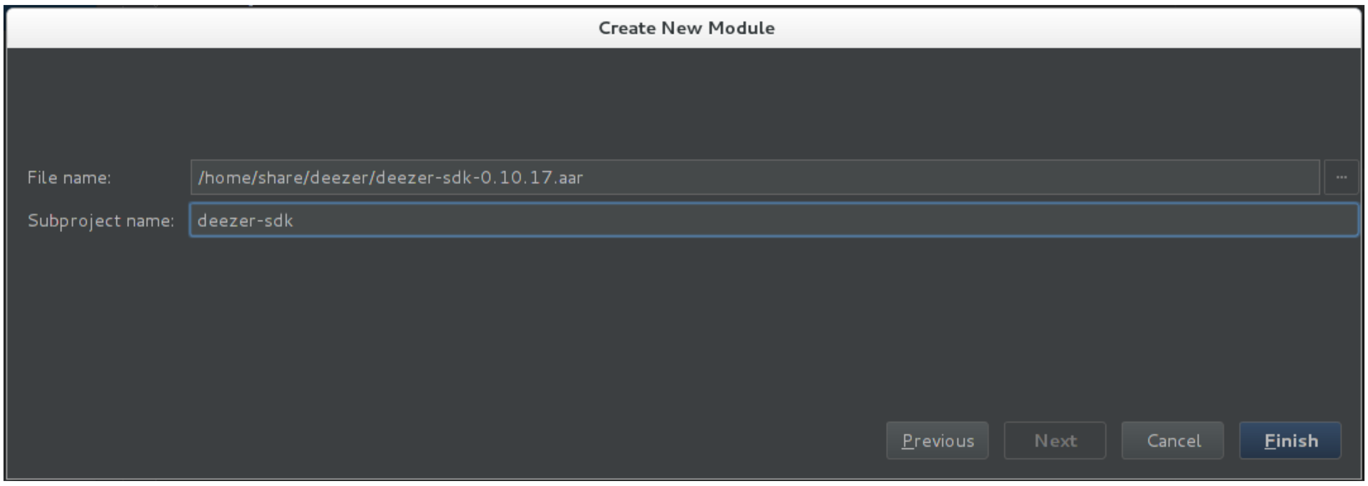


Choose Module Type

Select an option below to create your new module

Import .JAR or .AAR Package
Imports an existing JAR or AAR package as a new project module

Java Library
Creates a new Java library.



File name: /home/share/deezer/deezer-sdk-0.10.17.aar

Subproject name: deezer-sdk

4 Using the SDK

This section provides small code snippets to set you on tracks on using the Deezer SDK for Android. For more exhaustive information, check out the User Guide as well as the SDK Reference.

4.1 Android Permissions

In order for the Deezer SDK to run correctly, you need to add the following permissions to your application's manifest :

```
<uses-permission android:name="android.permission.INTERNET" />             // ❶
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  // ❷
<uses-permission android:name="android.permission.WAKE_LOCK" />            // ❸
```

❶ This permission is needed to download the tracks and send requests to the API

❷ This permission is used to know if we can use the current network to download tracks

❸ This permission is used to prevent the system for going into sleep when music is playing

4.2 Initializing the SDK

The main object required by the Deezer SDK for Android is the DeezerConnect object. To initialize it you need to provide your application ID, and optionnaly your application's context.

```
String applicationID = "123456";                                           // ❶
DeezerConnect deezerConnect = DeezerConnect.forApp(applicationID).
                                            withContext(this).
                                            build();
```

❶ Replace this with your own Application ID

4.3 Authenticating a user

To authenticate a user, you must define a set of Permissions you need for your app, as well as a listener for the authentication events.

```
// The set of Deezer Permissions needed by the app
String[] permissions = new String[] {                                      // ❶
    Permissions.BASIC_ACCESS,
    Permissions.MANAGE_LIBRARY,
    Permissions.LISTENING_HISTORY };

// The listener for authentication events
DialogListener listener = new DialogListener() {

    public void onComplete(Bundle values) {}

    public void onCancel() {}

    public void onException(Exception e) {}
};

// Launches the authentication process
deezerConnect.authorize(activity, permissions, listener);
```

❶ You can find the list of all permissions and what they mean in the Documentation

4.4 Remembering the user session

To avoid showing a pop-up every time a user launches your application, you can use a `SessionStore` to remember the user session between two runs of your application, and share credentials between two activities.

To save a user session, use the following code (for example when the authentication process triggers the `onComplete()` method on your listener).

```
SessionStore sessionStore = new SessionStore();
sessionStore.save(deezerConnect, context);
```

You can then restore the session like this (for example on your activity's `onCreate()` method).

```
SessionStore sessionStore = new SessionStore();
if (sessionStore.restore(deezerConnect, context)) {
    // The restored session is valid, navigate to the Home Activity
    Intent intent = new Intent(context, HomeActivity.class);
    startActivity(intent);
}
```

Important
You must always make sure that every `DeezerConnect` instance you use in your application are coherent. This means that everytime you create a new `DeezerConnect` instance, you should use the restore code above so that it contains a valid token.

4.5 Calling the API

Deezer API offers a feature rich REST API that allows to retrieve various informations about a user's account on Deezer (eg : favorite albums or playlist, listening history). Deezer SDK for Android provides many tools to ease the work of writing request to Deezer API :

- the DeezerRequestFactory class helps you create the correct request for the Deezer API;
- the DeezerConnect object can be used to run the request in background (alternatively, you can use the AsyncDeezerTask class to run a request in background);
- the RequestListener and JSONRequestListener classes can be used to listen for request results / error callbacks.

Below is a complete sample to fetch the albums for a given artist.

```
// the request listener
RequestListener listener = new JsonRequestListener() {

    public void onResult(Object result, Object requestId) {
        List<Album> albums = (List<Album>) result;                    // ❶

        // do something with the albums
    }

    public void onUnparsedResult(String requestResponse, Object requestId) {} // ❷

    public void onException(Exception e, Object requestId) {}             // ❸
}

long artistId = 11472;
DeezerRequest request = DeezerRequestFactory.requestArtistAlbums(artistID);
deezerConnect.requestAsync(request, listener);                            // ❹
```

1. The result object can be a basic model object (Track, Album), or a list. You should cast it depending on the request you're performing.
2. This method can occur when the result JSON was not parse, either because it returns an object that is not yet part of the SDK or because you're using a new API that is not known in your SDK version. The requestResponse contains the JSON response as a String, as returned by the API.
3. This method is called whenever an exception occurs in the request processing. It can be a network exception, an error response (for exemple if the parameters are invalid, or no user is authenticated), or a parse error.
4. The request is performed asynchronously, and the apropriate method of the listener will then be called (on the same thread that started the method).

4.6 Playing Music

The easiest way to play music with the SDK is to use one of the five players provided :

- ArtistPlayer;
- AlbumPlayer;
- PlaylistPlayer;
- RadioPlayer;
- TrackPlayer.

Each of these works in similar ways. Here is a sample use of an AlbumPlayer : you can simply provide the id of an album and it will play all the tracks from the album.

```
// create the player
AlbumPlayer albumPlayer = new AlbumPlayer(application, deezerConnect,
         NetworkStateCheckerFactory.wifiAndMobile());                             // 1

// start playing music
long albumId = 89142;
albumPlayer.playAlbum(albumId);

// ...

// to make sure the player is stopped (for instance when the activity is closed)
albumPlayer.stop();                                                          // 2
albumPlayer.release();
```

1. The `NetworkStateCheckerFactory.xxx` returns a `NetworkStateChecker` to specify when a track is allowed to be downloaded.
2. When you need to release the player, always call the `stop()` method before.

Warning
It's a best practice to allow the user to specify himself which network(s) to use, to fit his own data plan. Some default implementations are available through `NetworkStateCheckerFactory`. You can create your own implementation of `NetworkStateChecker` to get a better on control on this.

5 More Information and Resources

5.1 Sample application

Sample Android applications using Deezer SDK can be found on our Github account :

Deezer SDK Sample
This sample showcases almost all features available in the SDK, from authentication to playing music. Deezer SDK Sample

GraceNote2Deezer
This sample showcases how to use Deezer SDK along with GraceNote features (recognizing a song from a local MP3 file or from the device's microphone. GraceNote2Deezer

5.2 JavaDoc Reference

The Javadoc reference for all public classes included in the SDK is available inside the downloadable archive, in the docs folder.