Lenddo SDK

for Android

version 1.9

VERSION HISTORY

Date	Remarks	Author
10/20/2014	Version 1.0	Joseph Emmanuel Dayo
12/12/2014	Version 1.1 (Map Widget)	Joseph Emmanuel Dayo
1/21/2014	Version 1.2 (Bug Fix)	Joseph Emmanuel Dayo
1/29/2014	Version 1.3 (Eclipse compatibility)	Joseph Emmanuel Dayo
2/10/2014	Version 1.4 (DataSDK bundle)	Joseph Emmanuel Dayo
3/24/2015	Version 1.5 (FbRedirect)	Joseph Emmanuel Dayo
6/11/2015	Version 1.5.2 (No Changes, Bug fixes)	Joseph Emmanuel Dayo
7/23/2015	Version 1.6 (Removed Map widget)	Joseph Emmanuel Dayo
8/18/2015	Version 1.7 (Allow facebook token to be set)	Joseph Emmanuel Dayo
9/29/2015	Version 1.8 (No Changes, Bug fixes)	Joseph Emmanuel Dayo
10/02/2015	Version 1.9 (Allow verification to be called without the onboarding flow)	Joseph Emmanuel Dayo

SECTION 1: INTRODUCTION

This is the Lenddo SDK for Android based devices, if you are developing for other platforms like IOS and web, please refer to the online documentation: https://www.lenddo.com/documentation

The Lenddo SDK for Android allows you to integrate the Lenddo Verification and/or Scoring workflow seamlessly into your Android app.

SECTION 2: AN OVERVIEW OF THE LENDDO PROCESS

1. User fills up a form in your app.
2. User clicks on the **Lenddo Button** provided by the Lenddo SDK
3. A popup webview showing the Lenddo Authorize site is shown.
4. User completes the Lenddo Authorize process.
5. A callback to your app is initiated and your app will consume the results.
6. User is sent to the next phase in your app.

SECTION 3: GETTING STARTED

1. Make sure you have met all the requirements (See the Requirements section below)
2. Successfully run the SDK in a simple loan app (See the Installation Section below)
3. Add the Lenddo SDK libraries to your own application.
4. Test and deploy

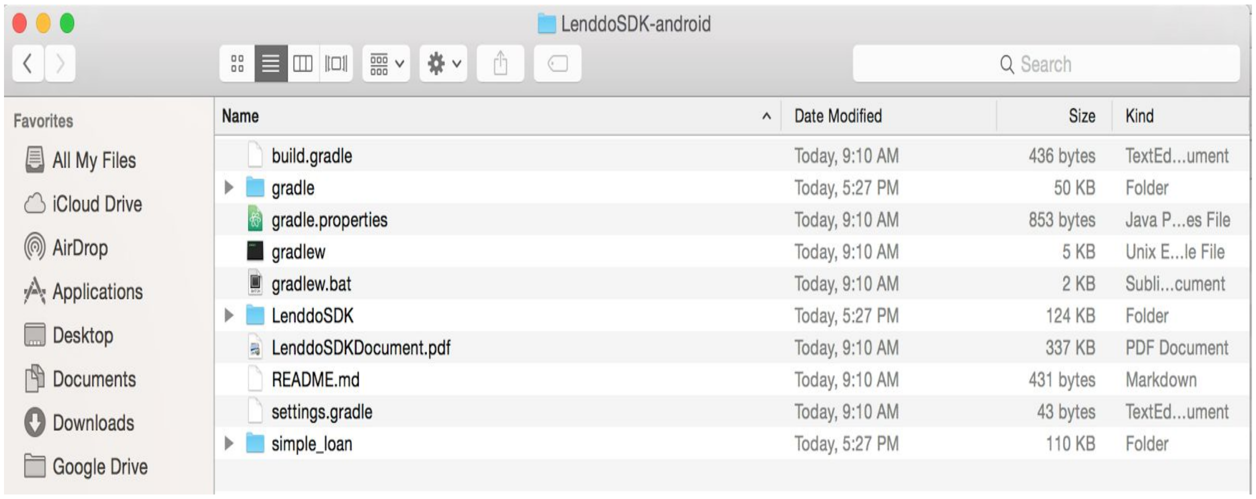
Requirements

Before you start on integrating the Lenddo SDK, please make sure you have the following:

a. Properly installed latest version of **Android Studio**. (You may refer to the Google Developer Docs on how to set this up https://developer.android.com/sdk/index.html)
b. A valid Lenddo **Partner Script Id**
c. Basic knowledge on setting up Android Libraries. (This document will explain the specific steps for the Lenddo SDK only).
d. Download the LenddoSDK onto your hard drive

Installation

Download the Lenddo SDK and extract the archive into your local drive. After extracting the archive, the Lenddo SDK folder structure should look like this:



Name	Date Modified	Size	Kind
build.gradle	Today, 9:10 AM	436 bytes	TextEd...ument
gradle	Today, 5:27 PM	50 KB	Folder
gradle.properties	Today, 9:10 AM	853 bytes	Java P...es File
gradlew	Today, 9:10 AM	5 KB	Unix E...le File
gradlew.bat	Today, 9:10 AM	2 KB	Subli...cument
LenddoSDK	Today, 5:27 PM	124 KB	Folder
LenddoSDKDocument.pdf	Today, 9:10 AM	337 KB	PDF Document
README.md	Today, 9:10 AM	431 bytes	Markdown
settings.gradle	Today, 9:10 AM	43 bytes	TextEd...ument
simple_loan	Today, 5:27 PM	110 KB	Folder

The **LenddoSDK** folder contains the actual Lenddo SDK library project that you can include in your app. The **simple_loan** folder contains the sample app called **Simple Loan** which illustrates how to integrate the **Lenddo Button** into your existing app.

SECTION 4: SETTING UP THE SAMPLE LOAN APP.

1. Using Android Studio, click on **Select File -> Open** and **choose** the folder LenddoSDK-android which was created when you extracted the Lenddo SDK.zip. Android Studio will automatically set up the project for you.

2. The sample app is already configured to use the LenddoSDK, all you need to do is to fill in your **partner-script-id**. Edit the **simple_loan/src/main/res/values/config.xml** and replace the words "PLACE YOUR PARTNER SCRIPT ID HERE" with the **partner_script_id** key provided to you. (See image below)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <!-- Lenddo Partner Script ID -->
<string
name="lenddo_app_id">PLACE_YOUR_PARTNER_SCRIPT_ID_HERE</string>
</resources>
```

3. Now, build and run the sample app *(make sure you have your emulator running or you have an Android device connected and configured for development. If you need more information on how to do this, please refer to the Android Studio documentation to learn more)*.

4. When the sample app successfully launches, you will see a sample form with a Lenddo Button at the bottom and q client ID field above it. This Client ID field corresponds to a user id or application id that is created by your app (for testing purposes you can enter a sample ID).

5. Click on the Verify with Lenddo button to complete the authorize process.

If you would like more information on how this works you can view The file **SampleActivity.java** in the simple_loan/src/main/java/lenddo.com.lenddoconnect folder

SECTION 5: ADDING THE LENDDO LIBRARY TO YOUR EXISTING PROJECT

Inside the extracted directory, copy the **LenddoSDK** subfolder and place it inside the root of your Application's Android Studio project folder. If you encounter an error copy the **LenddoSDK** subfolder in the Folder with the name of your Application in your computer.

Edit the **settings.gradle** file and add the following:

```
include ':LenddoSDK'
```

Open and edit the **build.gradle** file of your app (not the one in the project root but in the app folder), you should see a section for dependencies below is an example on how it looks:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

Add "compile project(':LenddoSDK')" so that it looks like the following:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile project(':LenddoSDK')
}
```

Android Studio should tell you to resync, the SDK classes should now be available after that.

Permissions

The required permissions are already defined in the Lenddo SDK and should automatically be incorporated to your app, however the permissions below are required:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Notes on Backwards Compatibility and Issues

There is a compatibility issue with the latest Chrome-based WebView on Android 4.4 (kitkat) and above. It is recommended to set your **targetSdkVersion** to **18** or below when using the Lenddo SDK.

The SDK has been tested on Android ICS 4.0 and above. Compatibility is not guaranteed on lower versions.

SECTION 6: INTEGRATION

Adding the Lenddo workflow to your app

- Edit your apps' AndroidManifest.xml (located in your src/main folder for gradle projects), and then add the following meta tag under the application tag:

```
    <meta-data android:name="partnerScriptId"
android:value="partner_script_id" />
```

where **partner_script_id** is the partner script id provided to you by Lenddo.

Add the Lenddo Button to your form

The Lenddo button greatly simplifies integrating the Lenddo workflow to your app.

1. Create your form (if you don't have an existing one already)

The Lenddo verification process requires at the minimum, the following fields;

- Primary Address
- Email
- Last Name
- Middle Name
- First Name
- Date of Birth
- Home Phone Number
- Mobile Phone Number
- University
- Employer

However the exact fields that is required for your App may be different depending on your requirements or use cases, please talk to your Lenddo Representative regarding this.

2. Open up your Forms' layout xml and add the following to include the Lenddo Button onto your Layout:

```
<com.lenddo.sdk.widget.LenddoButton
   android:id="@+id/verifyButton"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_gravity="center_horizontal"
   android:gravity="center" />
```

Note that you can also use your own custom button. (See section on customization for more information)

3. Create an Instance of the UIHelper class inside the onCreate block of your activity. Note that the class constructor requires a **LenddoEventListener**. For the sample app, it implements the current activity as a **LenddoEventListener**.

```
        private UIHelper helper;

        protected void onCreate(Bundle savedInstanceState) {
            ....
            helper = new UIHelper(this, this);
        }
```

4. Setup your activity to implement the **LenddoEventListener** in your class or you can define your own class:

```
     public class SampleActivity extends Activity implements
LenddoEventListener {

            ....

            private UIHelper helper;

            protected void onCreate(Bundle savedInstanceState) {
                ....
                helper = new UIHelper(this, this);

            }

            @Override
            public boolean onButtonClicked(FormDataCollector collector) {
                return true;
            }
            @Override
            public void onAuthorizeComplete(FormDataCollector collector) {
            }

            @Override
            public void onAuthorizeCanceled(FormDataCollector collector) {
            }

            ....
        }
```

Note: These methods allow you to hook into the Lenddo Authorize process.

5. Still on the onCreate method, Link the button to the UIHelper:

```
    LenddoButton button = (LenddoButton) findViewById(R.id.verifyButton);
        button.setUiHelper(helper);
```

6. Pass the content of the form using the form collector. On the onButtonClicked method, you can set the required information using the formData object passed to you. You can also send additional custom fields (To be discussed with your Lenddo representative)
and also the facebook token (See section on Passing the facebook token)

```
      @Override
         public boolean onButtonClicked(FormDataCollector formData) {

             //place partner defined user identifier
             formData.setUserId("123456789");
             formData.setLastName(lastName.getText().toString());
             formData.setFirstName(firstName.getText().toString());
             formData.setEmail(email.getText().toString());
             formData.setDateOfBirth(dateOfBirth);

             //send custom fields
             formData.putField("Loan_Amount",
loanAmmount.getText().toString());

             formData.validate();
             return true;
         }
```

Important Note: It is important here that you must pass a unique identifier to formData.setUserId, this will be used if you want to match your transaction records later on.

7. Clicking on the Lenddo Button should trigger the Lenddo Authorization/Verification process and your app will be notified via onAuthorizeComplete when the user id done.

8. Depending on your requirements a score may be available, in this case this is available through our REST APIs. *(Please check here for details https://www.lenddo.com/documentation/rest_api)*

Customizing the Lenddo Button

You may customize the Look and Feel of the Lenddo Button in a couple of ways:

1.) Style are available at the Lenddo SDK res/drawables where you can change various button attributes.

2.) You may create or use any of your existing Button. However you need to manually handle the onClick event with **UIHelper.showAuthorize** like this:

```
helper = new UIHelper(this, this);

Button sampleButton = (Button) findViewById(R.id.sample_button);

sampleButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
       UIHelper.showAuthorize(SampleActivity.this, helper);
   }
});
```

Passing a Facebook Token (Optional)

The lenddo SDK allows you to reuse the existing facebook token that you have in your app that was obtained using the Facebook SDK. This will allow the Lenddo onboarding process to skip its own facebook login and use your token instead. Do note that certain facebook permissions must be approved in your Facebook API account in order for the Lenddo verification process to work properly, below is the list:

email,public_profile,user_birthday,user_work_history,user_education_history,user_friends,user_likes

The above facebook permissions must be requested by your app when the user logs in.

Some of these permissions would require the you go through a verification process that is done by facebook. Make sure that you already have these permissions when you proceed. Any missing permission would result to problems or limitations in process.

Note that this is optional, if no facebook token is passed to the SDK, the facebook login will be shown during the onboarding process.

Setting up your App

Additional settings will need to be added to your app namely the **api secret.** This will be provided to you by your Lenddo representative.

1. Add the api secret. In your AndroidManifest.xml, make sure to add the following meta:

```
<application ....>

<meta-data android:name="partnerApiSecret" android:value="api_secret_here" />

</application>
```

2. Pass the actual facebook token during the onboarding process using the **setFacebookToken** method, the actual token and expiration date are passed (see below for an example) :

```
@Override
public boolean onButtonClicked(FormDataCollector formData) {
  //auto-collect (optional)
  formData .collect(SampleActivity.this, R.id.formContainer);

  AccessToken accessToken = AccessToken.getCurrentAccessToken();
  formData.setFacebookToken(accessToken.getToken().toString(),
accessToken.getExpires().getTime());

  Address primaryAddress = new Address();

  primaryAddress.setHouseNumber(houseNumber.getText().toString());
  primaryAddress.setStreet(street.getText().toString());
  primaryAddress.setBarangay(barangay.getText().toString());
  primaryAddress.setProvince(province.getText().toString());
  primaryAddress.setCity(city.getText().toString());
  primaryAddress.setPostalCode(postalCode.getText().toString());

. . . .
```

Directly Passing Data for Verification (Requires Facebook token)

If you have a facebook token and only verification is needed by your app, you may skip the authorize flow web process and submit your application directly to the Lenddo backend for processing. No popup will be shown and the whole process is done by code.

In order to do this the following are needed:

- A valid facebook access token. Please refer to the section Passing a Facebook Token (Optional) on the required permissions.
- **api secret.** This is provided by your Lenddo contact or obtained through your Dashboard. If you have done this before in Setting up your App then this is already setup. Otherwise perform step 1 of the process in the section Setting up your App.

Passing the data is similar to how it is done when using the Lenddo Button:

1. Create an instance of the UIHelper

```
helper = new UIHelper(activity, lenddoEventListener);
```

2. Implement the LenddoEventListener. Passing the required data is done on the onButtonClicked() method.

```
@Override
public boolean onButtonClicked(FormDataCollector formData) {
   formData.setClientId(customerId.getText().toString());
   formData.setLastName(lastName.getText().toString());
   formData.setMiddleName(middleName.getText().toString());
   formData.setHomePhone(homePhone.getText().toString());
   formData.setFirstName(firstName.getText().toString());
   formData.setEmail(email.getText().toString());
   formData.setEmployerName(nameOfEmployer.getText().toString());
   formData.setMobilePhone(mobilePhone.getText().toString());

   formData.validate();
   return true;
}
```

Note that you have to pass the fields that you want to verify here as well as the client id. For the client ID, if you intend to do the complete onboarding flow later, you will need to provide a different one for this.

3. Call the **UIHelper.startVerificationUsingFacebookToken()** to start the verificaton process passing along the UIHelper and facebook token details (See below for an example):

```
AccessToken accessToken = AccessToken.getCurrentAccessToken();

UIHelper.startVerificationUsingFacebookToken(activity,
accessToken.getToken().toString(), accessToken.getExpires().getTime(),
uiHelper);
```

The **onAuthorizeComplete()** callback method will be called once the process is complete. Lenddo's servers will begin to verify the passed information asynchronously. To obtain the verification results, please refer to the REST api documentation.

Using the auto-collector (Optional)

This is optional. If you don't what to use .getText() on settings the field values, the auto-collector allows you to send tagged fields in your layout by simply using formData.collect(). however, you do need to tell the auto-collector which fields need to be sent. This is done by adding a tag field to your layout xml like below:

```
<LinearLayout style="@style/fieldContainer">
                <TextView
                    style="@style/formLabel"
                    android:text="@string/house_number"/>
                <EditText
                    android:tag="house_number"
                    android:id="@+id/editTextHouseNumber"
                    android:hint="ex. 123"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content" />
            </LinearLayout>
            <LinearLayout style="@style/fieldContainer">
                <TextView
                    style="@style/formLabel"
                    android:text="@string/street_name"/>

                <EditText
                    android:tag="street_name"
                    android:id="@+id/editTextStreetName"
                    android:layout_width="match_parent"
                    android:hint="ex. makati ave"
                    android:layout_height="wrap_content" />
            </LinearLayout>
```

The **android:tag** name will be used by the auto-collector in determining which fields need to be sent and what the attribute name to use.