FLASHiZ Android Lib documentation

Technical documentation

for FLASHiZ integration

in Android applications

I. Introduction

The objective of this document is to describe the technical integration and usage of the “**FLASHiZ library module for Android**” (also called android_lib).

This product is a module in native Java for Android and can be used as a plugin by developers.

It targets any merchants that want to integrate in-app payments into their own Android application(s). This library is provided and maintained by FLASHiZ as an easy solution to integrate payment in third party Android Apps.

Once integrated in an app, this module is the sequence of the customer authentication and payment acceptation for an invoice presented by the merchant.

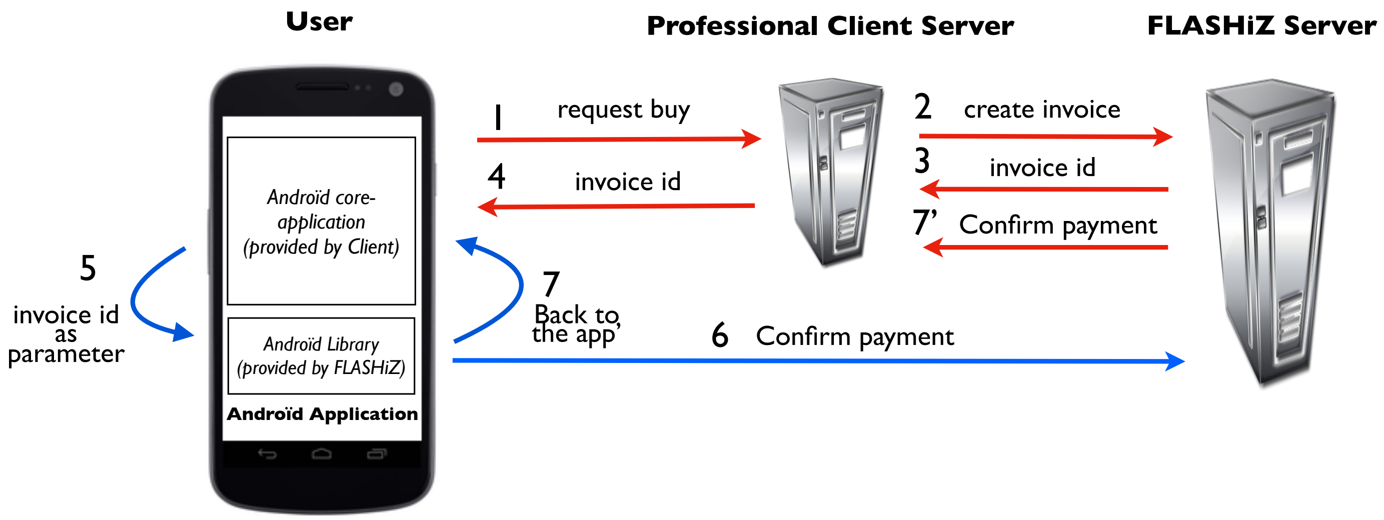
The «Android_lib» module has been developed as an intermediary between the client Android application and FLASHiZ’ servers.

This document presents FLASHiZ Java Android framework, specifically

- Framework installation
- API usage

II. Important notice

The process is carried out according to the following:



Red arrows in the diagram above are the ones concerned in the standard API module (detailed in another document). The blue arrows are managed by «Android library» module that is treated this document.

1.The client Android application send to the professional client server an invoice creation request.

2.The professional client server transmits request to the FLASHiZ server to initiate the transaction (standard API).

3.The FLASHiZ server returns an invoice id to the professional client server

4.The professional client server transmits the invoice id to the client Android application

5.The client Android application makes use of the invoice id as a parameter to FLASHiZ « Android library»

6.The FLASHiZ « Android library» creates the client process acceptance (user's authentication and confirmation)

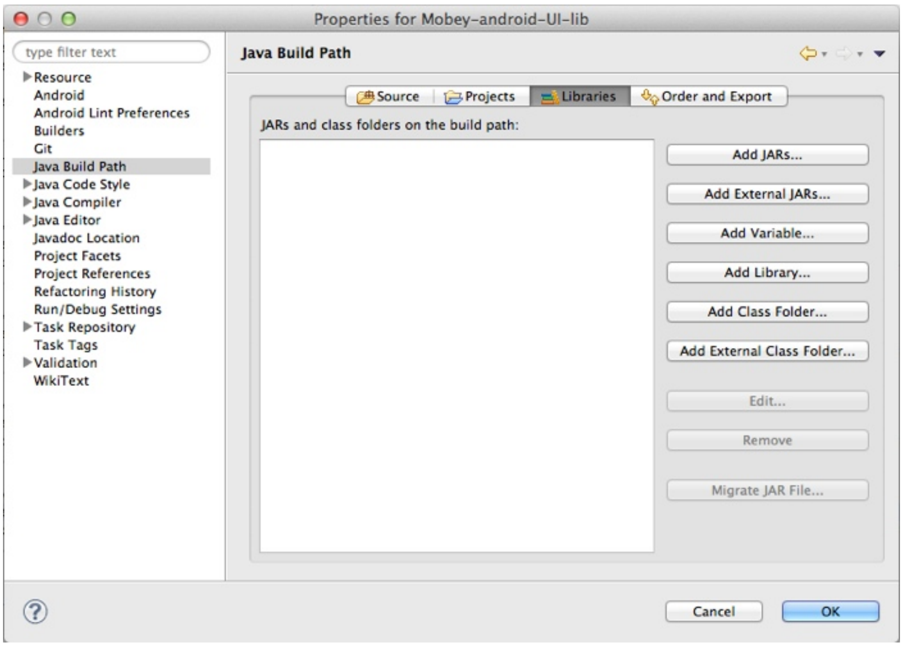
7.The FLASHiZ « Android library» adverts the application of the success or cancellation of the payment

7'. The FLASHiZ server confirms the payment the professional client server (standard API)

III. Framework installation

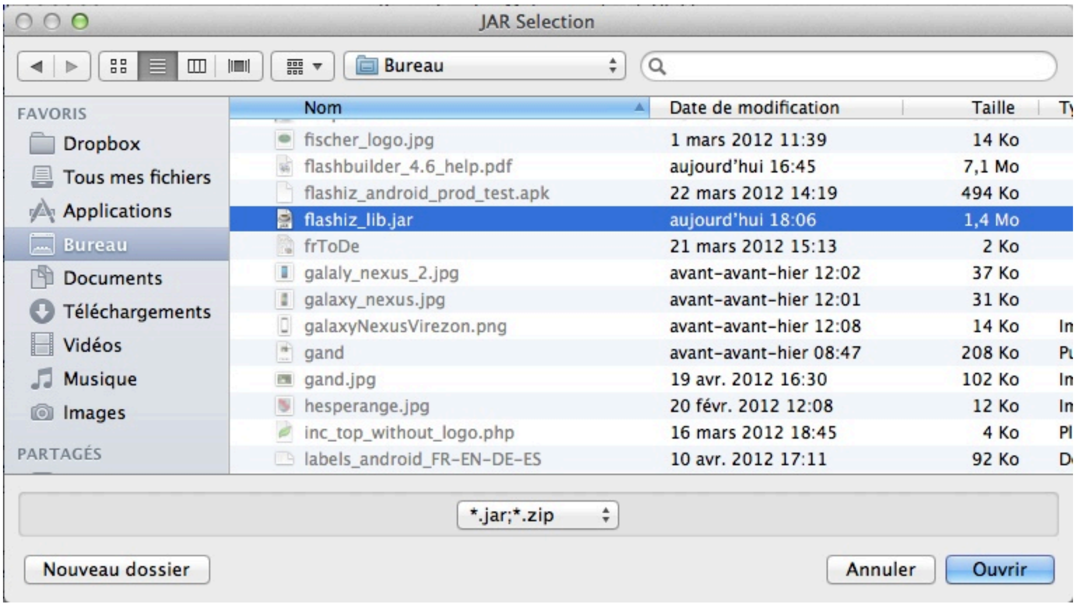
1. Setup Build Path

Right click on your Android project and select Build Path. The following screen should appear.



2. Add the flashiz_lib

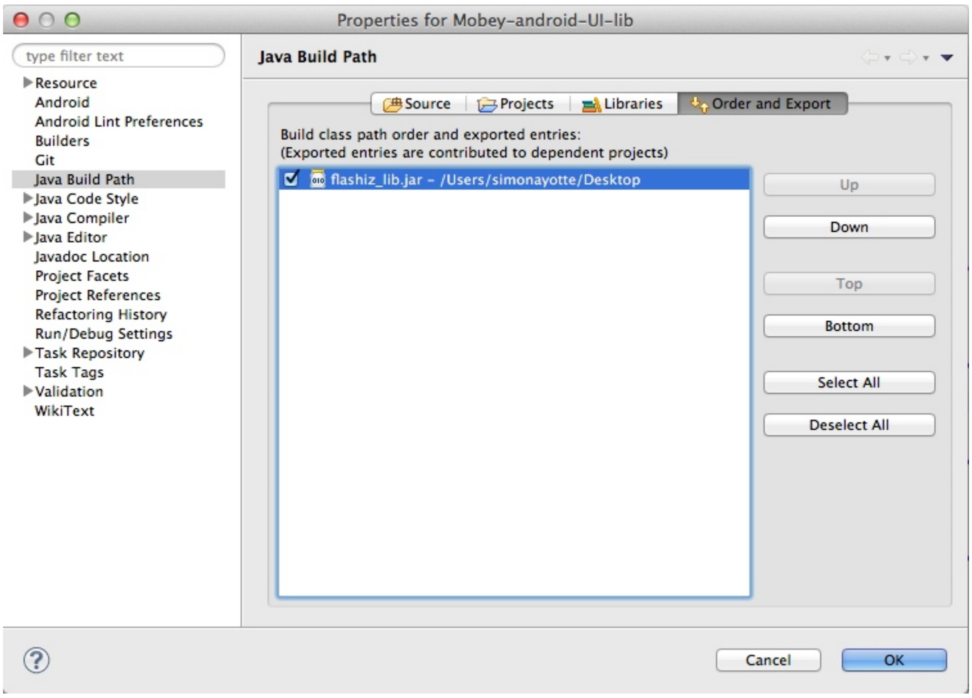
Then, click on Add External JARs button and select flashiz_lib jar file provided by FLASHiZ.



Nom	Date de modification	Taille	Ty
fischer_logo.jpg	1 mars 2012 11:39	14 Ko	
flashbuilder_4.6_help.pdf	aujourd'hui 16:45	7,1 Mo	
flashiz_android_prod_test.apk	22 mars 2012 14:19	494 Ko	
flashiz_lib.jar	aujourd'hui 18:06	1,4 Mo	
frToDe	21 mars 2012 15:13	2 Ko	
galaly_nexus_2.jpg	avant-avant-hier 12:02	37 Ko	
galaxy_nexus.jpg	avant-avant-hier 12:01	31 Ko	
galaxyNexusVirezon.png	avant-avant-hier 12:08	14 Ko	In
gand	avant-avant-hier 08:47	208 Ko	Pu
gand.jpg	19 avr. 2012 16:30	102 Ko	In
hesperange.jpg	20 févr. 2012 12:08	12 Ko	In
inc_top_without_logo.php	16 mars 2012 18:45	4 Ko	Pl
labels_android_FR-EN-DE-ES	10 avr. 2012 17:11	92 Ko	D

3. Order and Export the flashiz_lib

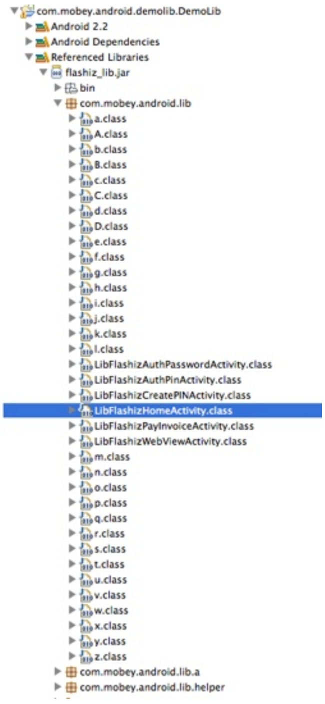
To display this library in the directories of your project, go to Order and Export and select flashiz_lib as below.



Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

☑ flashiz_lib.jar – /Users/simonayotte/Desktop

Remark : The flashiz_lib should appear in the directories of your project as below.The class selected will be the main class used by the payment process :



```
▼ com.mobey.android.demolib.DemoLib
  ▶ Android 2.2
  ▶ Android Dependencies
  ▼ Referenced Libraries
    ▼ flashiz_lib.jar
      ▶ bin
      ▼ com.mobey.android.lib
        ▶ a.class
        ▶ A.class
        ▶ b.class
        ▶ B.class
        ▶ c.class
        ▶ C.class
        ▶ d.class
        ▶ D.class
        ▶ e.class
        ▶ f.class
        ▶ g.class
        ▶ h.class
        ▶ i.class
        ▶ j.class
        ▶ k.class
        ▶ l.class
        ▶ LibFlashizAuthPasswordActivity.class
        ▶ LibFlashizAuthPinActivity.class
        ▶ LibFlashizCreatePINActivity.class
        ▶ LibFlashizHomeActivity.class
        ▶ LibFlashizPayInvoiceActivity.class
        ▶ LibFlashizWebViewActivity.class
        ▶ m.class
        ▶ n.class
        ▶ o.class
        ▶ p.class
        ▶ q.class
        ▶ r.class
        ▶ s.class
        ▶ t.class
        ▶ u.class
        ▶ v.class
        ▶ w.class
        ▶ x.class
        ▶ y.class
        ▶ z.class
      ▶ com.mobey.android.lib.a
      ▶ com.mobey.android.lib.helper
```

IV.API usage

FLASHiZ API allows to integrate FLASHiZ credit system functionalities in a third-party application or web site (external to FLASHiZ system).

Credit process is very similar to the payment process except *createCredit* replaces *createInvoice*.

1. introduction

In order to correctly understand implementation of FLASHiZ library, we have simulated his usage with an Activity called LibFlashizHomeActivity. This activity picks up the invoiceId and lauches the payment process.

Here is the definition of LibFlashizHomeActivity in your Activity.

```
Intent intent = new Intent(YourAcitivity.this, com.mobey.android.lib.LibFlashizHomeActivity.class);
intent.putExtra("invoiceId", invoiceId);
startActivity(intent);
```

2. Implementation in your activity

A configuration of library can be basic or completed with many parameters

```
// ----------------------------> instantiate the FLASHiZ activity
Intent intent = new Intent(UsingLIbActivity.this,com.mobey.android.lib.LibFlashizHomeActivity.class);
```

```
// ----------------------------> Invoice Id as input parameter
// Method for getting invoiceId directly or from a remote
String invoiceId =  "XXXXXXXXXXXX";
```

```
// ----------------------------> sequence activity for acceptation or refusal
// # create your list and fill your end activities
HashMap<String, Class<?>> configActivity = new HashMap<String, Class<?>>();
configActivity.put("activityBackAccepted", AcceptedActivity.class);
configActivity.put("activityBackCanceled", CanceledActivity.class);
```

```
// ----------------------------> create your Environment setup for language and server
//  # create your list
HashMap<String, String> configEnv = new HashMap<String, String>();
// ----------------------------> fill the language ( en , fr )
//  # Optional by default in English (en) here forced in French (fr)
configEnv.put("forceLang", "fr");
// ----------------------------> fill the Payment Engine Environment ( prod , integration )
//  # Optional default is production here forced in test
configEnv.put("FlashizEnvironment", "FE_TEST");
//configEnv.put("FlashizEnvironment", "FE_PRODUCTION");
```

```
// ----------------------------> fill all parameters to intent
intent.putExtra("invoiceId", invoiceId);
intent.putExtra("configActivity", configActivity);
intent.putExtra("configEnv", configEnv);
```

```
/**************************
for lauching the Flashiz activities there are 2 options:
option 1 : when the FLASHiZ activity ends, a new activity is lauched on the app (paramtered in the configActivity list)
option 2 : when the FLASHiZ activity ends, it comes back to the initial activity with response parameters
 **************************/
```

```
// ----------------------------> Option 1
startActivity(intent);
```

```
// ----------------------------> Option 2
com.mobey.android.lib.helper.FlashizApplication.getInstance().setFinish(false);
startActivityForResult(intent,com.mobey.android.lib.helper.Constants.MESSAGE_ACTIVITY_RESULT_LIB);
// in the initial activity, the app must have the onActivityResult function implemented in order for get result
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(requestCode == com.mobey.android.lib.helper.Constants.MESSAGE_ACTIVITY_RESULT_LIB) {
                switch (resultCode) {
                case com.mobey.android.lib.helper.Constants.MESSAGE_ACTIVITY_RESULT_ACCEPTED:
                                Log.d("result success", data.getStringExtra("invoiceId"));
                        break;
                case com.mobey.android.lib.helper.Constants.MESSAGE_ACTIVITY_RESULT_CANCEL:
                                Log.d("result error", data.getStringExtra("invoiceId"));
                        break;
                }
        }
        super.onActivityResult(requestCode, resultCode, data);
}
```

3. Update your Manifest.xml

Your AndroidManifest.xml should be completed with the following entries :

```
<activity
   android:name="com.mobey.android.lib.LibFlashizHomeActivity"
   android:noHistory="false"
   android:screenOrientation="portrait" />
<activity
   android:name="com.mobey.android.lib.LibFlashizAuthPasswordActivity"
   android:noHistory="true"
   android:screenOrientation="portrait" />
<activity
   android:name="com.mobey.android.lib.LibFlashizAuthPinActivity"
   android:noHistory="true"
   android:screenOrientation="portrait" />
<activity
   android:name="com.mobey.android.lib.LibFlashizCreatePINActivity"
   android:noHistory="true"
   android:screenOrientation="portrait" />
<activity
   android:name="com.mobey.android.lib.LibFlashizPayInvoiceActivity"
   android:noHistory="true"
   android:screenOrientation="portrait" />
<activity
   android:name="com.mobey.android.lib.LibFlashizWebViewActivity"
   android:noHistory="true"
   android:screenOrientation="portrait" />
```

V. Appendix

1. Documentation versions

V1.0	Initial version in EN
V1.1	Fixes and clarifications
V1.2	Adjustment on createCredit
V1.3	Add SEAL in call back URL
V1.4	New documentation structure and layout

2. FLASHiZ Environments

TEST environment (Sandbox):

Sandbox is dedicated to partners integration and test phases.

URL: http://test.mobey.net/mobey

Connecting to this environment requires user/psw provided by FLASHiZ (contactpro@flashiz.com)

Production environment:

To be used for Production Deployment

URL: https://my.flashiz.com/account/

Download Apps:

Test Apps can be downloaded here:

http://test.mobey.net/iMobey