ANDROID SDK v1.2.8

Contents

Introduction

Mimopay Android SDK is designed to help you integrate your android application or games to Mimopay's payment gateway. Mimopay Android SDK equipped with a built-in UI as well as quiet mode operation. Quiet mode means that after you initiate execution, the SDK will do the payment process in background instead of popping up its built-in UI. This will allow you to continue with other process that you want to do. All information, whether an error occurs or a successful payment, will be notified via 'onReturn' callback method.

Supported Channels

Topup:

1. Smartfren
2. Sevelin

ATM:

1. BCA
2. Bersama

Airtime (Telkomsel):

1. UPoint

XL:

1. Airtime
2. HRN (Voucher)

Prerequisites

Before you begin, you need to make sure that you are a registerred merchant at Mimopay payment gateway since the SDK will be useless without user ID, merchant code, and secret key that required to pass at initialization

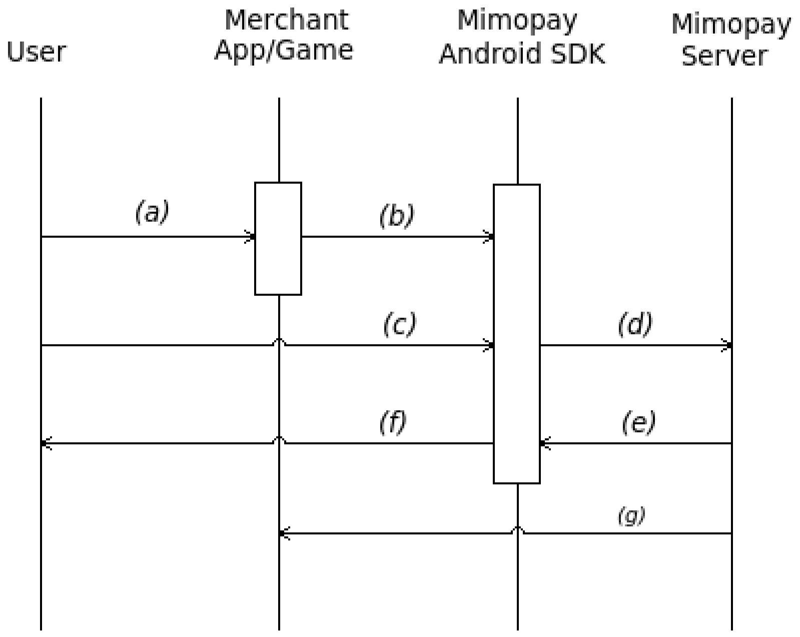
Download the SDK

The SDK is available at https://github.com/mimopay/Mimopay-Android-SDK. You can clone it with whatever git tool you have or use command line as shown below

```
git clone https://github.com/mimopay/Mimopay-Android-SDK.git
```

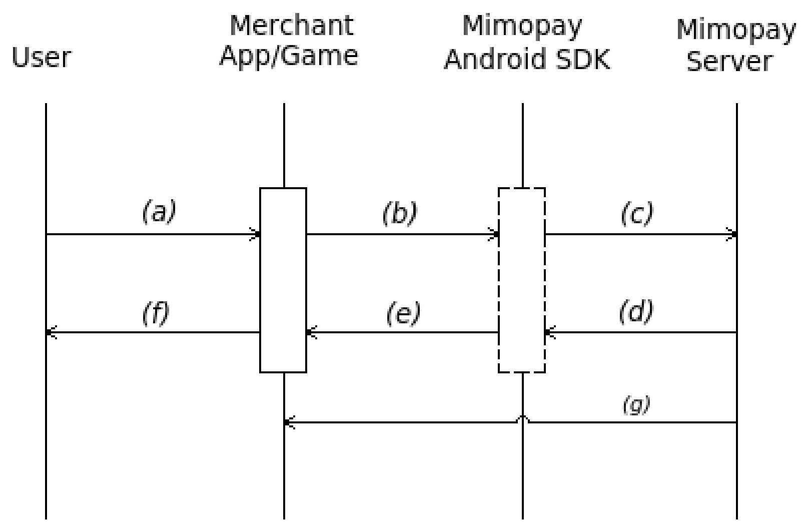
The Flow

UI Mode Flow Illustration



Where:
(a) User choose to do some payment.
(b) Merchant's app to activate SDK's built-in UI.
In this stage, merchant's app is become inactive. All interaction only between user and the SDK. The SDK provides necessary UI controls to simplify user to do the payment
(c) User choose some payment channel and method
(d) The SDK's UI pass all necessary things that user has choosen or input, to mimopay server
(e) Mimopay server replies back the results
(f) The SDK display the results
(g) Mimopay server than apply callback to merchant's server

Quiet Mode Flow Illustration



Where:
(a) User choose to do some payment.
Merchant's app provides some fancy, interactive payment UI to user. In this stage, the SDK running behind its scene.
(b) Merchant's app to execute process according to what payment method that user has choosen.
(c) The SDK pass all values or inputs to mimopay server
(d) Mimopay server replies back the results
(e) The SDK will filtering them, and pass necessary things to merchant's app
(f) Merchant's app display the filterred results to user
(g) Mimopay server than apply callback to merchant's server

Integrating the SDK

Before you start, you need to follow these steps below

1. You need to copy the Mimopay.jar into libs directory of your project.
2. The SDK require several android permissions so you need to add lines of codes below into the AndroidManifest.xml file of your project

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.READ_SMS" />
```

3. The SDK has it own built-in UI that will pops up during execution, so you need to add several line of codes below into the AndroidManifest.xml file of your project

```
<activity
        android:name="com.mimopay.MimopayActivity"
        android:theme="@android:style/Theme.NoTitleBar"
        android:windowSoftInputMode="stateUnspecified|adjustPan"
        >
</activity>
```

4. You also need to add these two line of codes into your java source.

```
import com.mimopay.Mimopay;
import com.mimopay.MimopayInterface;
```

Class and Interface

The following is the description of the SDK class and interface that you have imported in your java source code

```
public class Mimopay {
        public Mimopay(Context context,
                String emailOrUserId, String merchantCode, String productName,
                String transactionId, String secretKey, String currency,
                MimopayInterface mimopayinterface
        )
        public void executeTopup()
        public void executeTopup(String channel)
        public void executeTopup(String channel, String code)
        public void executeATMs()
        public void executeATMs(String channel)
        public void executeATMs(String channel, String amount)
        public void executeUPointAirtime()
        public void executeUPointAirtime(String amount, String phoneNumber,
                boolean autosendsms)
        public String getSdkVersion()
        public void enableGateway(boolean enable)
        public void enableLog(boolean enable)
        public String[] getLastResult()
}

public interface MimopayInterface {
        public void onReturn(String info, ArrayList<String> params);
}
```

Initialization

The SDK initialization require you to fill all your merchant stuffs into its parameters. Please refer to http://staging.mimopay.com/api.

```
Mimopay mimopay = new Mimopay(
        Context context,                // your android application context
        String emailOrUserId,
        String merchantCode,
        String productName,
        String transactionId,
        String secretKeyStaging,
        String secretKeyGateway,
        String currency,
        MimopayInterface mi             // you callback interface
);
```

transactionId may set to empty (transactioID = "") which cause the library to generate a unique value based on unix timestamp.

Note:
In the sample source codes, the secretKeyStaging and secretKeyGateway is managed as string of encrypted keys. Please contact us and we will generate them for you.

Callback

To obtain any information from the SDK during and after execution, you need to use MimopayInterface interface and override its oReturn method as decribed below.

```
        MimopayInterface mimopayinterface = new MimopayInterface()
        {
                public void onReturn(String info, ArrayList<String> params)
                {
                        if(info.equals("Success")) {
                                // do what you want to do
                        } else if(info.equals("Error")) {
                                // do what you want to do
                        } else if(info.equals("FatalError")) {
                                // do what you want to do
                        }
                }
        };
```

Use ‘info’ as indicator whether it is success, error, or fatal error status. Use ‘params’ to obtain the details. Be noticed that ‘params’ may set to **null** so you need to test first whether it equal to null or not. After that, you should obtain its size. Here is the ilustration

```
if(params != null) {
        int j = params.size();
        // ....
}
```

The following table visualize onReturn’s info string

Info	params		Meaning
Error	0	UserCancelled	User was cancelling the transaction
	0	MerchantLoadUrlNull	You pass the channel string other than listed in this document
	0	ErrorConnectingToMimopayServer	Internet connection problem
	0	ErrorValidatingVoucherCode	Occurs when SDK receive other then ‘ok’ status from server.
	0	ErrorInternetConnectionProblem	Internet connection problem
	0	UnsupportedPaymentMethod	Requested payment method does not exist
	0	UnspecifiedChannelRequest	Requested payment channel does not exist
	0	ErrorHTTP404NotFound	Server return 404 error code.
	0	ErrorUnderMaintenance	The current payment channel is under maintenance
Success	0..3	Depends on the channel	Message retrieved from server
Fatal Error	0	Depends on the uncaughtException message	Occurs when the SDK suddenly force closed by the Android OS

Executing Payment

The following table will help you to understand the SDK's methods and how they are categorized.

Pymt Method	Channel	Built-in UI	Quiet

Topup	smartfren
	sevelin
		executeTopup()
		executeTopup(String channel)
			executeTopup(String channel, String code)

ATM	atm_bca
	atm_bersama
		executeATMs()
		executeATMs(String channel)
			executeATMs(String channel, String code)

Airtime	upoint	executeUPointAirtime()	executeUPointAirtime(String amount, String phoneNumber, boolean autosendsms)
XL	xl_airtime xl_hrn	public void executeXL() public void executeXLAirtime() public void executeXLHrn()	- public void executeXLHrn(String code) - public void executeXLAirtime(String amount, String phoneNumber, boolean autosendsms)

On Airtime UPoint, the result that received from mimopay server would be SMS content: up <code> and the destination phone number. The SDK would not go automatically send SMS without user intervention. So for Airtime UPoint, the quiet mode operation will not be completely quiet. The built-in UI will pops up to allow user to type that codes

Parameter explanation:

String channel

channel should be one of the followings

smartfren
sevelin
atm_bca
atm_bersama
upoint
xl_airtime
xl_hrn

String code

A voucher code string for smartfren or sevelin topup

String amount

A mimocard value for atm_bca & atm_bersama

String phoneNumber

A phone number for UPoint & XL

boolean autosendsms

It will send an SMS automatically when it is set to ‘true’.
Please note that due to Indonesia Telco policy, fully automatic sending SMS is not allowed. When it is set to ‘true’, it will redirected to Mimopay’s built-in UI. It then will confirm to user to re-type the SMS code then the SMS will be sent.

Other Important Methods

- For some reason you may want to know the current SDK version. You may call getSdkVersion() to obtain the current SDK version

- By default, the SDK points to staging.mimopay.com during execution. On your production release, you should use gateway.mimopay.com instead of staging. To enable it you call enableGateway(boolean enable) with enable sets to true

- The last successful transaction always be recorded. You can call getLastResult() to obtain those transaction messages.

- Sometimes we need to visualize SDK’s log activities during process. You may enable it by calling enableLog(boolean enable) with enable sets to true

SDK Build

The SDK builds with Android API Level 17

Queries

If you have any queries or issues please email me at jimmy@mimopay.com or skype to jimmybas