Integration with Android Pay

Version: 1.8.0

1. Introduction

This document is an aid to the integration with the *payment-library-android-pay*. The library is a separate payment method (supported exclusively by Android) and, for the time being, it is not integrated with the client payment flows (classic integration). The client must have access to 'transparent API' (e.g. API 2.1) in order to properly call a payment flow.

2. Requirements

- concerning the device
    - Device with Android version 4.4 (KitKat) or later and NFC
    - Google Play Services installed on the phone
    - Android Pay App installed on the phone

      https://play.google.com/store/apps/details?id=com.google.android.apps.walletnfcrel
    - At least one card added to the Android Pay app
    - Server-side OAuth2 token retrieval
    - The device should not be rooted
- other:
    - In order to implement the new payment method, the client must contact Google about using Android Pay in a mobile app.

3. Android Pay history

The antecedent of Android Pay was Google Wallet, which enabled adding a card to your Gmail account and making online payments. However, that payment method did not catch on. Google decided to revisit payments after iOS Apple Pay was released. Android Pay enables storing your real cards in the virtual world. Based on a card recorded in the Google app, a virtual card is generated for a fixed amount to be used to make one specific payment. Tokens to be utilized for mobile payments are generated in the cloud. If you have lost your device, all stored cards can be easily removed using an integrated Google service.

4. Supported currencies and languages

- Languages:
    - Polish
    - English
    - German
    - Czech

- Currencies:
  - PLN
  - EUR
  - CZK

5. How to add the Android Pay library

Add remote repository

Maven (pom.xml)

```
<repositories>
    <repository>
        <id>payu-mvn-repo</id>
        <url>https://raw.github.com/PayU/paytouch-android/mvn-repo/</url>
    </repository>
</repositories>
```

Gradle (build.gradle)

```
repositories {
    /* PayU repository on GitHub */
    maven { url 'https://raw.github.com/PayU/paytouch-android/mvn-repo/' }
    mavenCentral()
}
```

Add required dependencies

In order to run properly, the library needs Google Play Services to communicate with the Android Pay app.

Maven (pom.xml)

```
<dependency>
    <groupId>com.payu.android.sdk</groupId>
    <artifactId>payment-library-android-pay</artifactId>
    <version>1.8.0</version>
    <type>aar</type>
    <exclusions>
        <exclusion>
            <groupId>com.android.support</groupId>
            <artifactId>support-v4</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.jetbrains</groupId>
            <artifactId>annotations</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.android.support</groupId>
            <artifactId>appcompat-v7</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>com.payu.android.sdk</groupId>
    <artifactId>payment-library-core-full</artifactId>
    <version>1.8.0</version>
    <type>jar</type>
    <exclusions>
        <exclusion>
            <groupId>com.android.support</groupId>
            <artifactId>support-v4</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.jetbrains</groupId>
            <artifactId>annotations</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>com.google.android.gms</groupId>
    <artifactId>play-services-wallet</artifactId>
    <version>9.6.1</version>
    <scope>provided</scope>
    <type>aar</type>
</dependency>
```

Gradle (build.gradle)

```
compile('com.payu.android.sdk:payment-library-android-pay:1.8.0') {
    exclude group: 'com.android.support', module: 'support-v4'
    exclude group: 'org.jetbrains', module: 'annotations'
    exclude group: 'com.android.support', module: 'appcompat-v7'
}
compile('com.payu.android.sdk:payment-library-core-full:1.8.0')
        {
            exclude group: 'com.android.support', module: 'support-v4'
            exclude group: 'org.jetbrains', module: 'annotations'
        }

provided('com.google.android.gms:play-services-wallet:9.6.1')
```

If any issues arise with dependencies in the library, it is recommended to use *compile* on the *play-services-wallet* library. Additional excludes depends on your other libraries.

AndroidManifest.xml

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

<application
<meta-data android:name="com.google.android.gms.wallet.api.enabled"
        android:value="true" />

    <meta-data android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
<service

android:name="com.payu.android.sdk.payment.service.PaymentEntrypointService"
    android:exported="false" />

</application>
```

You may want to keep track of the Google documentation containing up-to-date information on Android Pay integration:
https://developers.google.com/android-pay/get-started

6. Service

For payments to work properly, a service for passing the token to the library must be implemented on the mobile side. For more information, please refer to the documentation provided at:

https://github.com/PayU/paytouch-android/blob/master/docs/PayTouch_Android_v.1.8.0.pdf

Section: *Token Provider Service*

Service scope: **mobile.pay**

7. Environment

In case of integration test please use sandbox environment. This environment will call test environment on Android Pay application. When moving to production you need to contact PayU and Google.

For more information about the production environment, please refer to the documentation provided at:

https://github.com/PayU/paytouch-android/blob/master/docs/PayTouch_Android_v.1.8.0.pdf

Section *Configuration file & instantiation.*

8. Use

Basing on the Google documentation available at:

http://developers.payu.com/pl/restapi.html

each client should create their own payment process in the app and the library concerned can only enhance the said process by enabling Android Pay payments.

Payment Selection Screen

The client will use this template to retrieve all payments available for a given POS device and configured by PayU. A part of the *retrieve pay method* query should return:

```
{
    "value": "ap",
    "name": "Android Pay",
    "brandImageUrl": "https://androidpay_logo.png",
    "status": "ENABLED"
  }
```

The above responsei shows that the Android Pay payment is supported on a given POS device.

The next step is to verify if the device supports that payment method. To do so, the client should implement Listener: *AndroidPayVerificationListener* on their side.

Calling verification if payment can be made:

```
AndroidPayVerifier androidPayVerifier =
AndroidPayVerifierService.createInstance();
androidPayVerifier.isAndroidPaySupported(mContext,
androidPayVerificationListener);
```

Creating a new order

To create a new order (the create order method; for more information, please refer to the documentation provided at: http://developers.payu.com/pl/restapi.html) the payment method must be set to Android Pay (PBL).

```
...
"payMethods":{
        "payMethod":{
       "type":"PBL",
       "value":"ap"
  }
```

An Order Id will be returned with information to continue with the payment through Android Pay:

```
{
  "orderId": "PUBLIC_ORDER_ID_HASH",
  "status": {
    "statusCode": "WARNING_CONTINUE_ANDROID_PAY"
  }
}
```

The Order Id received will be used to pay for the order made.

Payment Screen

The final part refers to making an order through Android Pay:

```
AndroidPayService  androidPayService =
SdkAndroidPayService.createInstance();

AndroidPayPaymentData androidPayPaymentData = new
AndroidPayPaymentData("WspanialySklep", orderId);
androidPayService.pay(mContext, androidPayPaymentData);
```

The *pay* method can be called only after an *order* is created on the AndroidPay dedicated payment server. The *AndroidPayPaymentsData* object has two fields: merchant name, e.g. SuperStore, and *orderId.*

Handling requests in AndroidPay

The AndroidPay app passes information to *onActivityResult()* of the activity calling it and, consequently, the *handlePayPayment()* method must be called in *onActivityResult()*. This method returns a *boolean* field with information if the request has been handled on the SDK side (two request codes: *1000* and *1001,* are used in communication with AndroidPay).

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    boolean isHandled = androidPayService.handleAndroidPayPayment(requestCode,
resultCode, data);
    //other actions
}
```

9. Progress notification

For more information about notifications, please refer to the documentation provided at:

https://github.com/PayU/paytouch-android/blob/master/docs/PayTouch_Android_v.1.8.0.pdf

Section: *Event subscription*

.

10. ProGuard

For more information, how to use ProGuard, please refer to the documentation provided at:

https://github.com/PayU/paytouch-android/blob/master/docs/PayTouch_Android_v.1.8.0.pdf

Section: *Proguard*

11. Payment flow (conclusion)

- A user wishing to make a purchase chooses Android Pay from the list of available payment methods on their device.
- The user goes to the summary screen and then selects the Pay button.
- After the Pay button is pressed, a selection of cards stored in the Android Pay app is displayed.
- After selecting the card and pressing Next, the payment process is initiated.
- The merchant is notified of the transaction status in the app through the EventBus.

12. App release

For the app release please contact Google following the instructions available at: https://developers.google.com/android-pay/deployment

---

i