Android SDK

⚠ Updated version will be provided end of 2023

1. Overview

This is the documentation for the Axepta Android SDK, which describes how to integrate payments in your Android app.

The integration of the SDK is achieved by following a list of steps:

- Configuration of your merchant account in the Axepta system.
- Making your Android project runnable with gradle and adding the SDK into your project.
- Configuration of the SDK by inserting appropriate data retrieved from Axepta.
- Configuration of preferable payment methods.
- Authentication against the merchant backend.
- Insertion of the payment data.
- Checkout.
- Handling of errors.

List of supported payment methods

The Android SDK currently supports the following payment methods:

- Credit Card
- SEPA Direct Debit
- PayPal
- WeChat

2.Requirements

Requirements for using the SDK:

Basically you need: your Axepta merchant account with according information and correct setting of payment parameter values.

2.1. Preparation

- You need an existing merchant account at Axepta
- You will receive your Merchant ID from Axepta after creation of the merchant account
- You need backend and the belonging URL on merchant side to create and deliver auth token (see chapter 4.2)
- Website and the belonging URL's on merchant side to forward and show the status of a payment process in case of a success, failure or a notify event (see chapter 4.3.2).

2.2. Development

- Using gradle as dependency management in your Android project
- Android min SDK Version is 16

3.Installation

Add jCenter in the main build.gradle:

```
allprojects {
repositories {
jcenter()
}
}
```

Then add the following line to your app build.gradle:

```
implementation 'com.axepta.sdk:lib:<current_version>'
```

and sync gradle.

4.How to implement

You can find a demo on https://github.com/axepta/axepta-online

Try the demo in order to see how it works or use the following step-by-step guideline.

4.1. Configuration

Configure the SDK by importing the Axepta class and inserting the configuration parameters -merchantID, authURL - you receive from Axepta.

Set the provided parameters in the file strings.xml

```
<resources>
<string name="axepta_merchant_id">XXX</string>
<string name="axepta_auth_url">XXX</string>
</resources>
```

4.2. Authentication

One requirement for the Mobile SDK is to insert the respective Merchant's URL in order to be able to receive the auth token. The SDK is responsible for retrieving the token and use it appropriately when executing payment requests.

For more information, see the Axepta documentation.


```
<resources>
<string name="axepta_merchant_id">XXX</string>
<string name="axepta_auth_url">XXX</string>
</resources>
```


4.3. Making a payment

Assume your app user inserted some products into the basket, typed the shipping address and now wants to make the payment - that means you want to offer the user different payment options he can choose from.

For that you need to ask the SDK for supported payment methods and then configure the payment data.

First, initialize an instance of the SDK, activity should be instance of FragmentActivity.

```
Axepta axepta = Axepta.with(getActivity())
```

4.3.1. Available payment methods

To get all the available payment methods you need to call:

```
axepta.requestPaymentMethods()
.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThread())
.subscribe((paymentMethods, throwable) -> {
if (throwable != null) {
// manage error
Log.e(TAG, "PaymentMethods error: ", throwable);
return;
}
// manage the paymentMethods list (List<PaymentMethod>)
});
```

You receive a payment icon and localized description foreach payment method.

4.3.2. Payment data

The next step is to select a payment method, then fill payment data with the proper key and values based on the documentation for your payment method (more details on Payment can be found in the Axepta documentation). You need to set valid parameters for your payment or the transaction will not work.

```
Payment getPayment(@NonNull PaymentMethod paymentMethod) {
Payment payment = paymentMethod.getPayment();
payment.setParamWithKey("TransID", "********");
payment.setParamWithKey("Amount", "100");
payment.setParamWithKey("Currency", "EUR");
payment.setParamWithKey("URLSuccess", "http://*****/api/success");
payment.setParamWithKey("URLNotify", "https://*****/axepta/index.php");
payment.setParamWithKey("URLFailure", "http://*****/api/failure");
payment.setParamWithKey("RefNr", "******");
payment.setParamWithKey("OrderDesc", "Tests");
payment.setParamWithKey("AddrCity", "Berlin");
payment.setParamWithKey("FirstName", "Lorem");
payment.setParamWithKey("LastName", "Ipsum");
payment.setParamWithKey("AddrZip", "10000");
payment.setParamWithKey("AddrStreet", "Berlin");
payment.setParamWithKey("AddrState", "AL");
payment.setParamWithKey("AddrCountryCode", "DE");
payment.setParamWithKey("Phone", "01777777124");
payment.setParamWithKey("LandingPage", "Login");
payment.setParamWithKey("eMail", "******-accounts@****.com");
payment.setParamWithKey("ShopID", "1");
7
payment.setParamWithKey("Subject", "******-accounts@****.com");
payment.setParamWithKey("Language", "en");
payment.setParamWithKey("Channel", "APP");
payment.setParamWithKey("MerchantID", "****");
return payment;
}
```

The method *setParamWithKey*() returns **true** if the value will be set successfully for the specified key.

The values of *Amount* and *Currency* are validated during the checkout process.

You have to ensure that you are using valid data.

The currency is the currency you want to use you for the payment. You have to use three characters for currency coed (DIN/ISO 4217), e.g. “EUR”.

The amount is the lowest unit of the currency you are using. That means if you are using EUR as currency the amount needs to be in cents, e.g. an amount of 100 is 1 EUR.

URLSuccess and *URLFailure* are the URL’s to which the SDK redirect the status of a payment process. These URL’s normally point to a HTML site on your merchant backend to show the status to the user of your app.

For the rest of the parameters you should take a look into the Axepta documentation.

Some of the parameters need to be defined by you and are mandatory.

Now you know the available payment methods and you have configured them. It’s time to show the payment methods to the user.

4.3.2.1. WeChat Pay

When using WeChat Pay as a payment method you also have to add the class *WXPayEntryActivity* which extends from *WeChatPayEntryActivity*.

```
public class WXPayEntryActivity extends WeChatPayEntryActivity {

}
```

This class needs to be placed inside a package called *com.mypackage.wxapi*, where *com.mypackage* is your *applicationId* defined in your app build.gradle.

For example, if your *applicationId* is *com.axepta.android.sdk.example*, the class *WXPayEntryActivity* should be placed inside the package *com.axepta.android.sdk.example.wxapi*

Besides that, you have to define the following Strings in your string.xml.

```
<string name="axepta_wechat_app_id" translatable="false">your wechat app id</string>

<string name=" axepta_wechat_mch_id" translatable="false">your wechat merchant id</string>

<string name=" axepta_wechat_key" translatable="false">your wechat key</string>
```

4.3.2.1. PayPal

For this payment method, the Chrome Custom Tab is used. To get back from the Chrome Custom Tab to your app after payment is done (or canceled), you have to redirect to a custom uri from your merchant backend.

The flow after payment is done / canceled:
Axepta will call your merchant backend using URLSuccess / URLFailure you specified.
On your merchant backend you have to redirect from your URLSuccess / URLFailure to the custom uri to get back to our app, e.g.:

URLSuccess merchant://com.shop.demo.PayPalReturn://return
URLFailure merchant://com.shop.demo.PayPalReturn://cancel

It is important to use the values **return** and **cancel** as path segments for success and failure.

In your app's string.xml you have to define the Strings *axepta_paypal_intent_uri_host* and *axepta_paypal_intent_uri_scheme.*

Following the example above, you must define the two Strings as follows:

```
<string name="axepta_paypal_intent_uri_host"

translatable="false">com.shop.demo.PayPalReturn</string>

<string name="axepta_paypal_intent_uri_scheme" translatable="false">merchant</string>
```

Optional: You can define the color of the Chrome Custom Tab toolbar in your colors.xml as follows:

```
<color name="colorCctToolbar">#3F51B5</color>
```

4.3.3. Make the checkout

Now that you have configured the SDK, selected the payment method and filled the values for your payment data, you are ready to make a payment.

For the Checkout we are using Retrofit underneath.

Start the checkout by instantiating subscription object (Observable). The *Axepta* class is a toplevel class that facilitates the payment procedure. It is responsible for validating payment data and instantiating a *WebView* when a new payment is triggered by passing the respective payment and *paymentMethod*.

```
Disposable disposable = axepta
.withPaymentMethod(method)
.setWebViewListener(() -> {
Log.i(TAG, "[WebViewClient] onPageFinishedLoading");
})
.checkout()
.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThread())
.subscribe(o ->
{
Log.v(TAG, "Payment received");
}, throwable -> {
if (throwable instanceof AxeptaError) {
Log.e(TAG, "Axepta Error " + (throwable));
showErrorAlert((AxeptaError) throwable);
} else if (throwable instanceof InternalError) {
Log.e(TAG, "Internal Error boolean isPaymentCanceled(): " + ((InternalError)
throwable).isPaymentCanceled());
} else {
Log.e(TAG, "Payment error: ", throwable);
}
});
```

Observable<T> subscribeOn(Schedulers.io()) will execute the operation on the background thread. When you subscribe to this observable you can show a loading indicator to the user. To get notified when the Webview is loaded you can set the listener *OnWebViewLoadedListener* and hide the loading indicator.

In case of a successful transaction you will get notified in onNext Consumer.

In case of an error you will receive the response in onError Consumer. You have to handle all types of errors including network errors during the checkout process and give feedback to the user.

4.4. Support screen rotation

If the activity in which you are doing the payment using Axepta SDK (only for payment methods that use WebView) needs to support screen rotation than you have to handle the configuration change.

To do this you have to add this flag in your AndroidManifest.xml file:

```
<activity android:name=".MyActivity"
android:configChanges="orientation|screenSize"
android:label="@string/app_name">
```

In this case you will have to handle the configuration changes for your activity.

For more info please check:

- https://developer.android.com/guide/topics/resources/runtimechanges.html#HandlingTheChange
- https://developer.android.com/reference/android/app/Activity.html#ConfigurationChanges