PayWithEaseBuzz Payment kit Integration (Android)

Android

The PaywithEaseBuzz Android SDK makes it quick and easy to build an excellent payment experience in your Android app. We provide powerful UI screens and elements that can be used out-of-the-box to collect your user's payment details. We also expose the low-level API's that power those UI's so that you can build fully custom experiences. See our Android Integration Guide to get started.

This SDK allows you to integrate payments via PaywithEaseBuzz into your native Android app. It currently supports the following modes of payments:

1. Credit / Debit Cards
2. Netbanking
3. Wallets/Cash Cards
4. Debit + ATM
5. UPI
6. Ola-Money
7. EMI

Features

1. **Simplified Security:** With fraud detection and prevention mechanisms, we provide the most protective layer for each transaction.We are also PCI-DSS compliant.
2. **Native UI:** We provide out-of-the-box native screens and elements so that you can get started quickly without having to think about designing the right interfaces.

Requirements

1. The PaywithEaseBuzz Android SDK is compatible with the Android Operating System KitKat and above.

NOTE: According to PCI regulations, payment processing is not allowed on TLS v1 and TLS v1.1. Hence, if the device does not have TLS v1.2, the SDK will throw an error while initiating the payment . You can learn more about TLS versions here.

SDK Configuration

To configure PaywithEaseBuzz Android SDK into your Android application you have to follow the below steps.:

1. Copy peb-lib-android-x.aar file into the app/libs/ folder of your application (If libs folder is not there, Please create it manually).
2. Add the below android proguard rule into your proguard rules file

```
-keepclassmembers class com.easebuzz.payment.kit.**{
    *;
}
```

3. To add the SDK into your app, open the build.gradle (module) and add the following lines with the respective section. If the following section already exists in file then only add lines into their respective section.

   3.1 Add multiDexEnabled into defaultConfig

```
defaultConfig {
    multiDexEnabled true
    }
```

   3.2 Add the following lines to packagingOptions

```
packagingOptions {
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/NOTICE.txt'
        exclude '*/res/**'
        exclude 'AndroidManifest.xml'
}
```

   3.3 Add the following lines to dexOptions

```
dexOptions {
  javaMaxHeapSize "4g"
}
```

3.4 Add the repositories section as follows

```
repositories {
    flatDir {
        dirs 'libs'
    }
 }
```

3.5 Add Dependencies

The following all mentioned dependencies are mandatory for the sdk. If any of the following dependencies you are already using into your app then you do not require to add these particular dependencies.

```
implementation (name: 'peb-lib-android-x', ext: 'aar')
implementation 'com.google.android.material:material:1.3.0'
implementation 'com.squareup.okhttp:okhttp:2.4.0'
implementation 'androidx.multidex:multidex:2.0.0'
implementation 'com.squareup.okhttp:okhttp-urlconnection:2.2.0'
implementation 'com.squareup.retrofit2:retrofit:2.5.0'
implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
implementation 'com.google.android.gms:play-services-auth:17.0.0'
implementation 'com.google.android.gms:play-services-auth-api-phone:17.1.0'
```

3.5 To open the upi app, you must add the below lines to AndroidManifest.xml file.

```
<queries>
   <package android:name="com.google.android.apps.nbu.paisa.user" />
   <package android:name="net.one97.paytm" />
   <package android:name="com.phonepe.app" />
   <package android:name="in.org.npci.upiapp" />
   <package android:name="in.amazon.mShop.android.shopping" />
   <package android:name="com.whatsapp" />
</queries>
```

Your Easebuzz Kit configuration is done.

Initiate Payment

This is a mandatory and important step for initiating the payment. To generate an accesskey you have to integrate the Initiate Payment API at your backend. After you successfully call the Initiate Payment API then you will get an access key which is the value of the data key of the response. You can check the Initiate Payment API Doc Click here.

Note - It is mandatory to integrate the Initiate Payment API at your Backend only.

Integration code

After having successfully set up the sdk configuration and Initiate Payment API at your backend, write the below code to start payment on the click of Pay button from your app.

1. Import below packages into your file
   ```
   import com.easebuzz.payment.kit.PWECouponsActivity;
   import datamodels.PWEStaticDataModel;
   ```

2. Fetch the access key into your app which will get in response of Initiate Payment and pass that access key to payment intent.

3. To start payment from your app you have to start PWECouponsActivity and pass necessary parameters to PWECouponsActivity using Intent. (Use startActivityForResult() method to start the Activity and a response is received on onActivityResult() on the same activity). For start PWECouponsActivity you have to use below code

   Sample Code -

   ```
   Intent intentProceed = new Intent(MerchantAppActivity.this, PWECouponsActivity.class);
                   intentProceed.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);    // This is mandatory flag
                   intentProceed.putExtra("access_key","Access key generated by the Initiate Payment API");
                   intentProceed.putExtra("pay_mode","This will either be “test” or “production");
                   startActivityForResult(intentProceed, PWEStaticDataModel.PWE_REQUEST_CODE);
   ```

   In the above sample code you have to pass the access_key and pay_mode.
   The access key will be like - “555a2b009214573bd833feca997244f1721ac69d7f2b09685911bc943dcf5201” and you will be get it from the initiate payment API. The **pay_mode** parameter will either be "test" or "production". Your key and salt will depend on the pay_mode which you pass.

   Note
   1. Please do not change the name parameter of the putExtra() method mentioned in above code. The datatype of the value can be changed according to the value to be sent.
   2. PWEStaticDataModel.PWE_REQUEST_CODE is integer constant, having value 100.

Handle Payment Response

1. In your app you must override the OnActivityResult() method of the invoking activity class to handle transaction status and transaction response. In the Intent extras, you will receive a set of response parameters which is used to determine if the transaction was successful or not. Also the requestCode must be PWEStaticDataModel.PWE_REQUEST_CODE

2. In intent extras you will get key "result" and you can retrieve value of result as follow
   String result = data.getStringExtra("result");
   The following are the values of the result you can get

   "payment_successfull"
   "payment_failed"
   "txn_session_timeout"
   "back_pressed"
   "user_cancelled"
   "error_server_error"
   "error_noretry"
   "invalid_input_data"
   "retry_fail_error"
   "trxn_not_allowed"
   "bank_back_pressed"

   Note: Description of the result values and detailed response are given at the end of the document.

3. In intent extras you will get the key "payment_response" which is payment detail response and you can payment_responset as follow

   String response = data.getStringExtra("payment_response");

4. You can refer the below sample code how to handle transaction response into the OnActivityResult() method

Note: Handle the response only when the parameter ‘Intent data’ is not null in onActivityResult() method.

Sample Code -

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

        if(data != null )  {
            if (requestCode == PWEStaticDataModel.PWE_REQUEST_CODE) {

                String result = data.getStringExtra("result");
                String payment_response = data.getStringExtra("payment_response");
                try {

                    // Handle response here
                    }catch (Exception e){
                        // Handle exception here
                    }
                }
            }
        }
```

Response Description

1. Payment result values description and equivalent constants

Response	Value
PWEStaticDataModel. TXN_SUCCESS_CODE	PWEStaticDataModel.TXN_SUCCESS_CODE is a string constant and its value is “payment_successfull”. This result contains this value, if the payment transaction is completed successfully.
PWEStaticDataModel. TXN_TIMEOUT_CODE	PWEStaticDataModel.TXN_TIMEOUT_CODE is a string constant and its value is “txn_session_timeout”. This result contains this value, if the payment transaction failed because of the transaction time out.
PWEStaticDataModel. TXN_BACKPRESSED_CODE	PWEStaticDataModel.TXN_BACKPRESSED_CODE is a string constant and its value is “back_pressed”. This result contains this value, if the user pressed the back button on coupons Activity.
PWEStaticDataModel. TXN_USERCANCELLED_CODE	PWEStaticDataModel.TXN_USERCANCELLED_CODE is a string constant and its value is “user_cancelled”. This result contains this value, if the user pressed the cancel button during the payment process.
PWEStaticDataModel. TXN_ERROR_SERVER_ERROR_CODE	PWEStaticDataModel.TXN_ERROR_SERVER_ERROR_CODE is a string constant and its value is “error_server_error” . This result contains this value, if the server side error occurred during the payment process.
PWEStaticDataModel. TXN_ERROR_TXN_NOT_ALLOWED_CODE	PWEStaticDataModel.TXN_ERROR_TXN_NOT_ALLOWED_CODE is a string constant and its value is “trxn_not_allowed”.
PWEStaticDataModel.TXN_BANK_BACK_PRESSED_CODE	PWEStaticDataModel.TXN_BANK_BACK_PRESSED_CODE is a string constant and its value is “bank_back_pressed”. This result contains this value, if the user presses the back button on the bank page.
PWEStaticDataModel. TXN_INVALID_INPUT_DATA_CODE	PWEStaticDataModel.TXN_INVALID_INPUT_DATA_CODE is a string constant and its value is “invalid_input_data’. This result contains this value if payment request input parameters are not valid.
PWEStaticDataModel. TXN_FAILED_CODE	PWEStaticDataModel.TXN_FAILED_CODE is a string constant and its value is “payment_failed” . This result contains this value if payment fails from the bank side.
PWEStaticDataModel. TXN_ERROR_NO_RETRY_CODE	PWEStaticDataModel.TXN_ERROR_NO_RETRY_CODE is a string constant and its value is “error_noretry”. This result can be considered as a failed payment.
PWEStaticDataModel. TXN_ERROR_RETRY_FAILED_CODE	PWEStaticDataModel.TXN_ERROR_NO_RETRY_CODE is a string constant and its value is “retry_fail_error”. This result can be considered as a failed payment.

2. The below is detail response of payment

2.1 Success response json.


```
{
          txnid : '1001',
        firstname : 'John Doe',
        email : 'johndoe@gmail.com',
        phone : '9876543210',
        key : 'DF3252FDSF',
        mode : 'DC',
        status : 'success',
        unmappedstatus : 'failed',
        cardCategory : 'domestic',
        addedon : '2016-07-22 17:17:08',
        payment_source : 'Easebuzz',
        PG_TYPE : 'SBIPG',
        bank_ref_num : '',
        bankcode : 'MAST',
        error : 'E600',
          error_msg : 'Bank denied transaction on card.',
          name_on_card : 'John',
          cardnum : '519620XXXXXX7840',
          issuing_bank : '',
          card_type : '',
          easepayid : 'H5T2RYZKW',
          amount : '100.00',
          net_amount_debit : '100.00',
          cash_back_percentage : '50',
          deduction_percentage : '2.50',
          productinfo : 'Tshirt',
          udf10 : '',
          udf9 : '',
          udf8 : '',
          udf7 : '',
          udf6 : '',=
          udf5 : '',
          udf4 : '',
          udf3 : '',
          udf2 : '',
          udf1 : '',
          hash :
'ce2d0588f8648c62db86475d343d3433d00b87827502c676a093730f04cec5fea2eb0e8bb'
          }
```

2.2 Failure response json.


```
{
        txnid : '1001',
        firstname : 'John Doe',
        email : 'johndoe@gmail.com',
        phone : '7767819428',
        key : 'DF3252FDSF',
        mode : 'DC',
        status : 'failure',
        unmappedstatus : 'failed',
        cardCategory : 'domestic',
        addedon : '2016-07-22 17:17:08',
        payment_source : 'Easebuzz',
        PG_TYPE : 'SBIPG',
        bank_ref_num : '',
        bankcode : 'MAST',
        error : 'E600',
        error_msg : 'Bank denied transaction on card.',
        name_on_card : 'John',
        cardnum : '519620XXXXXX7840',
        issuing_bank : '',
        card_type : '',
        easepayid : 'T5T2RYZKW',
        amount : '100.00',
        net_amount_debit : '100.00',
        cash_back_percentage : '50',
        deduction_percentage : '2.50',
        productinfo : 'Tshirt',
        udf10 : '',
        udf9 : '',
        udf8 : '',
        udf7 : '',
        udf6 : '',
        udf5 : '',
        udf4 : '',
        udf3 : '',
        udf2 : '',
        udf1 : '',
        hash
:'ce2d0588f8648c62db86475d300b87827502c676a093730f04cec5fea2ebb4f47fcdea955f61b6'
        }
```