PayWithEaseBuzz Payment kit Integration (Xamarin)

Xamarin

The PaywithEaseBuzz Xamarin SDK makes it quick and easy to build an excellent payment experience in your Xamarin 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 Xamarin Integration Guide to get started.

This SDK allows you to integrate payments via PaywithEaseBuzz into your Xamarin app(Supporting Xamarin Android Platforms). 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 mechanism, 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 Xamarin SDK is supported for Android. 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.

Android Setup

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

1. Download and add EasebuzzAndroidBinding into the androidx app.
2. To access the EasebuzzAndroidBinding library into your androidx app you have to add as reference into the reference section of the android-x app.
3. Install the following native android-x dependencies into your app using Xamarin Package Manager commands. (Package Manager console can be found in Tools->NuGet Package Manage - > Package Manager Console)

Execute below commands on Package Manager Console

```
Install-Package Xamarin.Google.Android.Material -Version 1.3.0
Install-Package Square.OkHttp -Version 2.7.5.1
Install-Package Xamarin.AndroidX.MultiDex -Version 2.0.1.10
Install-Package Square.OkHttp.UrlConnection -Version 2.7.5.1
Install-Package Xamarin.Android.SquareUp.Retrofit2 -Version 2.9.0
Install-Package Square.Retrofit2.ConverterGson -Version 2.9.0
Install-Package Xamarin.GooglePlayServices.Auth -Version 119.2.0.2
Install-Package Xamarin.GooglePlayServices.Auth.Api.Phone -Version 117.5.1.2
```

4. Add the below permission in AndroidManifest.xml.

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

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>
```

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.

Xamarin code

1. Create an Interface as below in your App.

```
public interface PWEInterface  {

        void processEasebuzzPayment(Dictionary <string, string> customerData);
}
```

2. Write the below cs code in your checkout file (On click of pay button)

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

2.2 To start payment from your app Write below code

```
var customerdata = new Dictionary<string, string> () {
        { "access_key", "Access key generated by the Initiate Payment API"},
        { "pay_mode", "This will either be “test” or “production" }
};

DependencyService.Get().processEasebuzzPayment(customerdata);
```

In the above 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.

2.3 Write the below method to handle payment response :

```
        public static void processPaymentResponse(Dictionary<string, string>
paymentDictionary) {
                foreach (var kvp in paymentDictionary) {
                        string key = kvp.Key;
                        string values = kvp.Value;
                    }
                }
```

In the above method you will get the result as well as details about the ongoing transaction as follows
Get Result : string result = paymentDictionary["result"];
Get transaction id : string txnid= paymentDictionary["txnid"];

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.

Android code

1. Modify MainActivity.cs located in the android directory as below.
   1.1 Implement Interface which you have declared before (Point 1 of xamarin code section).
   1.2 Add below assembly level attribute tag above android namespace.
   Syntax - [assembly: Xamarin.Forms.Dependency(typeof(ClassName))
   Example - [assembly: Xamarin.Forms.Dependency(typeof(MainActivity))
   1.3 Import Datamodels using the below lines.
   using Datamodels;
   1.4 Implement method of interface which you have declared before (Point 1 of xamarin code section) and write the below code in it to start android payment

```
try {
 var clsName = Java.Lang.Class.ForName("com.easebuzz.payment.kit.PWECouponsActivity")
 Intent intentProceed = new Intent(Android.App.Application.Context, clsName)
foreach (var kvp in customerData) {
string key = kvp.Key;
                    string values = kvp.Value
intentProceed.PutExtra(key, values)
}
((Activity)Forms.Context).StartActivityForResult(intentProceed,
PWEStaticDataModel.PweRequestCode)
    }
catch (Exception e) {
}
```

   1.5 Override OnActivityResult() method as below
   You will get a payment response and result in OnActivityResult() method. This response is a json string so you have to parse the response and add it into the Dictionary and pass a dictionary object to the processPaymentResponse() method for access response in xamarin code..

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

```
protected override void OnActivityResult(int requestCode, [GeneratedEnum] Result resultCode,
Intent data) {

base.OnActivityResult(requestCode, resultCode, data);

        if (data != null) {
            if (requestCode == PWEStaticDataModel.PweRequestCode) {
                var dictionary = new Dictionary();
                dictionary.Add("result", data.GetStringExtra("result"))

                JSONObject jSONObject = new
        JSONObject(data.GetStringExtra("payment_response"));

                for (int i = 0; i < jSONObject.Names().Length(); i++) {
                dictionary.Add(jSONObject.Names().GetString(i),
        jSONObject.Get(jSONObject.Names().GetString(i)).ToString());
                }

                MainPage.processPaymentResponse(dictionary);
                }
            }
        }
```

Response Description

1. Payment result values description and equivalent constants

Response	Value
"payment_successfull"	It is a string constant and its value is “payment_successfull.” The result contains this value, if the payment transaction is completed successfully.
“txn_session_timeout”	It is a string constant and its value is “txn_session_timeout”. The result contains this value, if the payment transaction failed because of the transaction time out.
"back_pressed"	It is a string constant and its value is “back_pressed”. The result contains this value, if the user pressed the back button on coupons Activity
“user_cancelled”	It is a string constant and its value is “user_cancelled”. The result contains this value, if the user pressed the cancel button during the payment process.
“error_server_error”	It is a string constant and its value is “error_server_error”. The result contains this value, if the server side error occurred during the payment process.
“trxn_not_allowed”	It is a string constant and its value is “trxn_not_allowed”
“bank_back_pressed”	It is a string constant and its value is “bank_back_pressed”. The result contains this value if the user presses the back button on the bank page.
“invalid_input_data"	It is a string constant and its value is “invalid_input_data". The result contains this value if payment request input parameters are not valid.
“payment_failed”	It is a string constant and its value is “payment_failed” . result contains this value if payment fails from the bank side.
“error_noretry”	It is a string constant and its value is “error_noretry”. This result can be considered as a failed payment.
“retry_fail_error”	It 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 :
        'ce2d0588f8648c62db86475d343d3433d00b87827502c676a093730f04cec5fea2
        eb0e8bb'
}
```

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
    :'ce2d0588f8648c62db86475d300b87827502c676a093730f04cec5fea2ebb4f47fc
    dea955f61b6'
}
```