PayWithEaseBuzz Payment kit Integration (Capacitorjs)

Capacitorjs

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

This SDK allows you to integrate payments via PaywithEaseBuzz into your capacitor app(Supporting Android/iOS 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

Requirements

1. The PaywithEaseBuzz Capacitorjs SDK is supported for Android and iOS platforms.The PaywithEaseBuzz Capacitorjs SDK is compatible with apps supporting iOS 10 and above, requires Xcode 9.2 to build from source and Android 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.

Installation

1. Execute the below command to install the sdk into your app.

   ```
   npm install $(npm pack  <Path of Capacitor SDK>/easebuzz-capacitorjs-sdk | tail -1)
   ```

   Example : npm install $(npm pack HomeDirectory/SDKfolder/easebuzz-capacitorjs-sdk | tail -1)

Android Setup and Configuration

1. Copy peb-lib-android-x.aar file into android/app/libs/ folder of your capacitor JS application (If libs folder is not there, Please create it manually).
2. 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.

Capacitor Code

1. Write below TypeScript Code to Start Payment using Easebuzz Payment Gateway.
   1.1 Import the below packages

```
import { EasebuzzCheckoutPlugin, EasebuzzCheckout } from 'easebuzz-capacitorjs-sdk';
```

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

1.3 Call EasebuzzCheckout.proceedToPayment method with the payment request parameters as options. This method returns data and receives the result and response of the payment process.

```
async load_paymentgateway() {
        const option = {
        access_key: ' Access key generated by the Initiate Payment API',
        pay_mode : ' This will either be "test" or "production '  }
        let data = await EasebuzzCheckout.proceedToPayment({ option });
        console.log(data['result']);
        console.log(data['payment_response']);
}
```

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.

Android Code

1. Modify MainActivity.java located in the android directory as below.

1.1 Import the below package in MainActivity.java as below.

```
import com.easebuzz.capacitorjs.EasebuzzCheckoutPlugin;
```

1.2 Write below code in MainActivity.java.

```
@Override
  public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            this.init(savedInstanceState, new ArrayList<Class<? extends Plugin>>() {{
            add(EasebuzzCheckoutPlugin.class);
      }});
  }
```

Note for iOS -

1. To open upi psp app, you must make the following changes in your iOS app's Info.plist file

```
<key>LSApplicationQueriesSchemes</key>
  <array>
  <string>tez</string>
  <string>phonepe</string>
  <string>paytm</string>
  <string>gpay</string>
  </array>
```

2. The existing plugin does not work on a simulator. The iOS framework is shipped with simulator architectures , so you have to replace simulator Easebuzz.framework (from iOS Frameworks folder) with device Easebuzz.framework. Please find the below folder path for replacing the framework.
   replace path: easebuzz-capacitorjs-sdk -> ios -> Pods -> Easebuzz -> Easebuzz.framework

3. Make sure at the time of uploading the app to APPStore, use the existing framework - iOS device framework.

Handle Payment Response

The payment response should be handled in your checkout.js file from where you started the payment.

1. The data is the Key Value Pair contains the response of payment so In the app you can get the detailed result by:

   String result = data['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"

2. You will get the key "payment_response" which is payment detail response and you can retrieve payment_response as follow:

   String detailed_response = data['payment_response'];

   This detailed_response is a HashMap and can be parsed to get the details about the ongoing transaction.

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

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 a detailed 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'

}
```