

PayWithEaseBuzz Payment kit Integration (React Native)

React Native

The PaywithEaseBuzz React Native 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 React Native Integration Guide to get started.

This SDK allows you to integrate payments via PaywithEaseBuzz into your React Native app(Supporting Android/iOS Platforms). It currently supports the following modes of payments:

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

The PaywithEaseBuzz React Native SDK is supported for Android and iOS platforms. The PaywithEaseBuzz React Native 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 React Native SDK>/react-native-easebuzz-kit | tail -1)
```

Example : `npm install $(npm pack HomeDirectory/SDKfolder/react-native-easebuzz-kit | tail -1)`

2. For React Native 0.59 and lower

```
react-native link react-native-easebuzz-kit
```

Android Setup and Configuration

To configure Android setup and configuration into your 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 rules](#) into your proguard rule 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

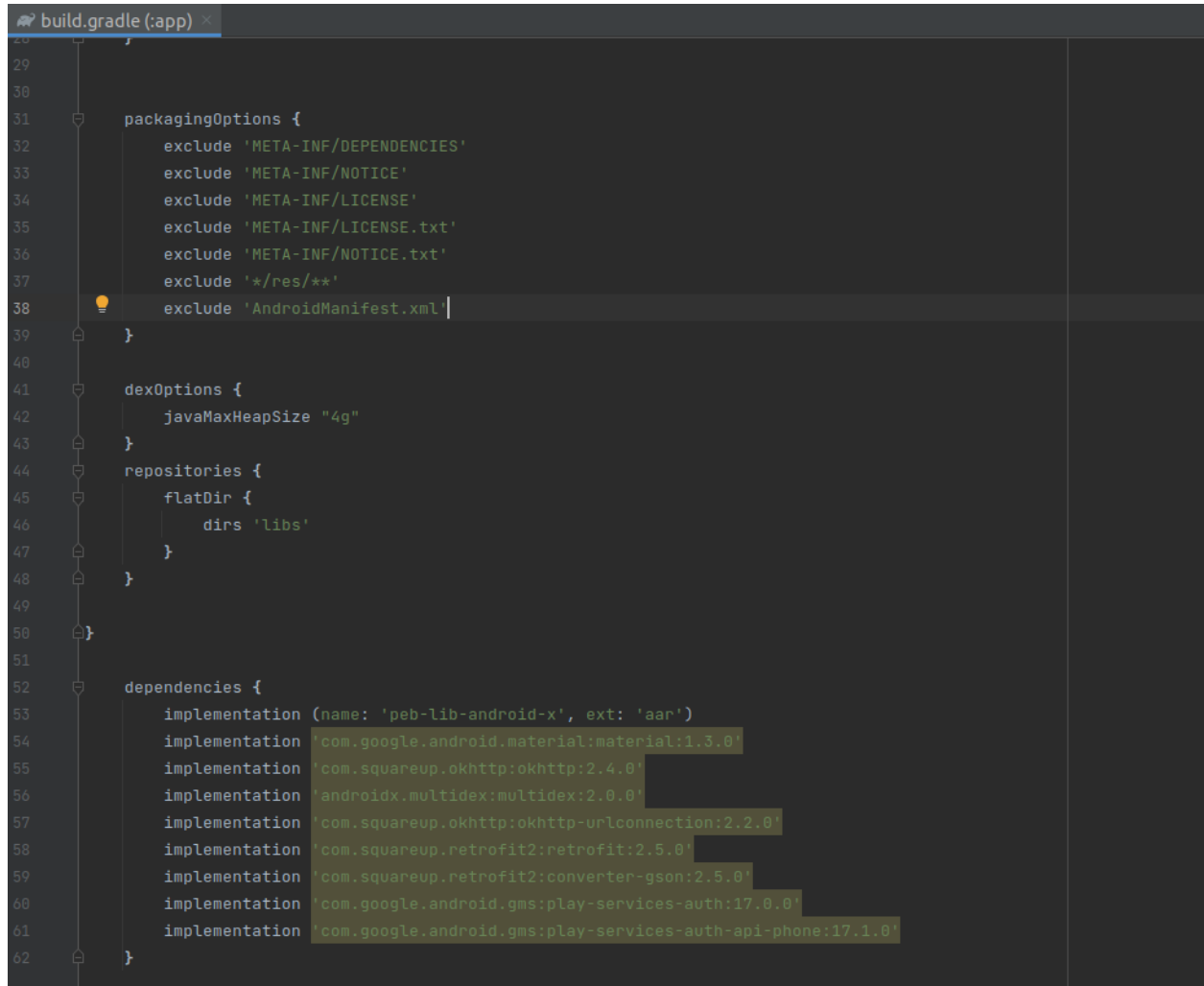
```
repositories {  
    flatDir {  
        dirs 'libs'  
    }  
}
```

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'
```

After complete configuration of [build.gradle](#) (module) it will be like



```
build.gradle (app)
28
29
30
31 packagingOptions {
32     exclude 'META-INF/DEPENDENCIES'
33     exclude 'META-INF/NOTICE'
34     exclude 'META-INF/LICENSE'
35     exclude 'META-INF/LICENSE.txt'
36     exclude 'META-INF/NOTICE.txt'
37     exclude '*/res/**'
38     exclude 'AndroidManifest.xml'
39 }
40
41 dexOptions {
42     javaMaxHeapSize "4g"
43 }
44
45 repositories {
46     flatDir {
47         dirs 'libs'
48     }
49 }
50
51
52 dependencies {
53     implementation (name: 'peb-lib-android-x', ext: 'aar')
54     implementation 'com.google.android.material:material:1.3.0'
55     implementation 'com.squareup.okhttp:okhttp:2.4.0'
56     implementation 'androidx.multidex:multidex:2.0.0'
57     implementation 'com.squareup.okhttp:okhttp-urlconnection:2.2.0'
58     implementation 'com.squareup.retrofit2:retrofit:2.5.0'
59     implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
60     implementation 'com.google.android.gms:play-services-auth:17.0.0'
61     implementation 'com.google.android.gms:play-services-auth-api-phone:17.1.0'
62 }
```

3.6 Change allowBackup value to true in `<application>` tag of [AndroidManifest.xml](#) file as below in your projects android folder.

```
android:allowBackup="true"
```

3.7 To open the upi app, you must add the below lines outside `<application>` into [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>
```

3.8 Add the below code in your launcher activity of the app.

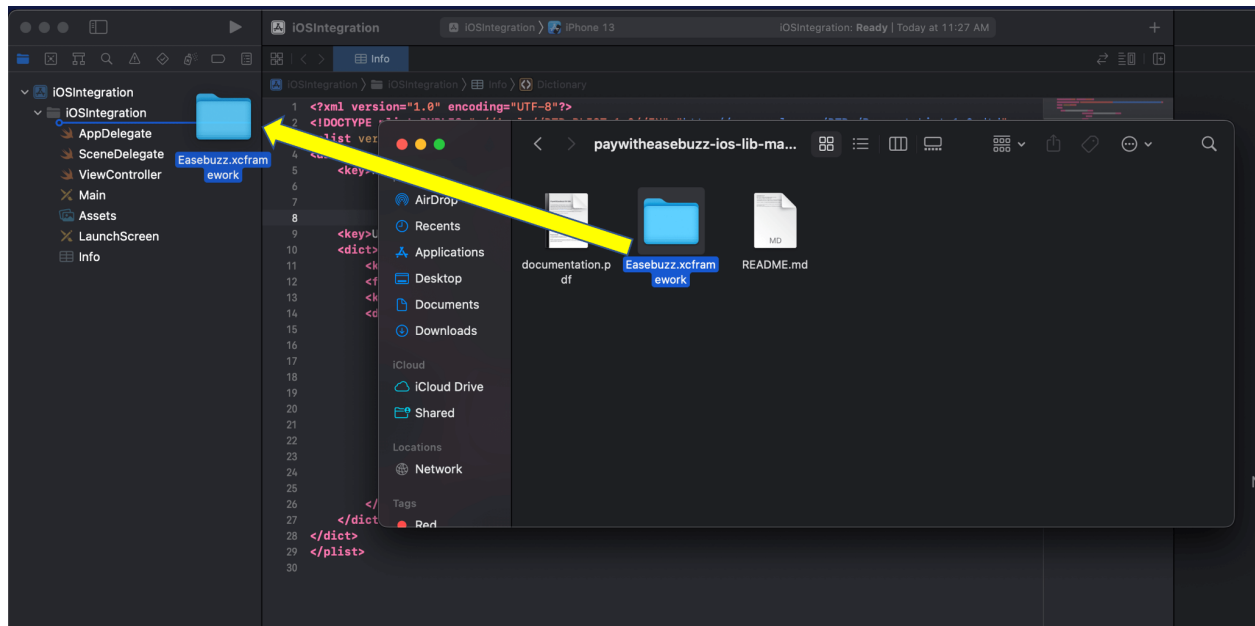
```
import android.content.Intent;

@Override
public void onNewIntent(Intent intent) {
    this.setIntent(intent);
}
```

iOS Setup and Configuration

To configure iOS setup and configuration into your application you have to the below steps:

1. Copy [Easebuzz.xcframework](#) of your application in embedded binaries.



2. Press + and add framework using 'Add other' button.
3. Browse framework: file from your folder and select 'copy items if needed'.
4. Set Always embed & sign from frameworks and Libraries and swift standard libraries to YES from project build settings.
5. As per your react native version run below command

For React Native 0.60+

- open podfile # Change the platform from iOS 9.0 to 10.0
- `cd ios && pod install && cd ..` # CocoaPods on iOS needs this extra step

For React Native 0.59 and lower

- Link the SDK with React Native Project using Xcode. `react-native link react-native-easebuzz-kit`

6. The existing plugin does not work on the 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.
 - a. **Copy file** >> Easebuzz.framework
 - b. **File path** >> "paywitheasebuzz-react-native-lib-master/iOS Frameworks/ios-simulator/Easebuzz.framework
 - c. **Replace with file** >> Easebuzz.framework
 - d. **Replace path** >> paywitheasebuzz-react-native-lib-master/easebuzz-react-native-kit/src/ios
7. 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>
```

Note: If you changed the simulator framework with existing, make sure at the time of uploading the app to APPStore, use the existing framework - iOS-Frameworks/ios-device/Easebuzz.Framework.

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 and also mandatory to integrate the Transaction API.
- Also, it is critical to configure webhooks as this will allow the Easebuzz system to 'push' data to your system whenever any event occurs, ensuring that both systems are always in sync.

React Native Integration code

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

1. Import following components:

```
import {Platform, Button, NativeModules,NativeEventEmitter} from 'react-native';  
import EasebuzzCheckout from 'react-native-easebuzz-kit';
```

2. To start payment you have to fetch the access key into your app which will get in response from the Initiate Payment api and pass that access key to the object.
3. Call EasebuzzCheckout.open method with the payment request parameters as options. This method returns a JS Promise where then part corresponds to a successful payment response or failure response and the catch part corresponds to any sdk failure i.e event failed etc

```
const callPaymentGateway = () => {  
  var options = {  
    access_key: "Access key generated by the Initiate Payment API",  
    pay_mode: "This will either be "test" or "production"  
  }  
  EasebuzzCheckout.open(options).then((data) => {  
    //handle the payment success & failed response here  
    console.log("Payment Response : ",data);  
  }).catch((error) => {  
    //handle sdk failure issue here  
    console.log("SDK Error : ",error);  
  });  
}
```

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.

Handle Payment Response

The payment response should be handled in the callback of `EasebuzzCheckout.open` method from where you started the payment.

1. You have to use `DeviceEventEmitter` for android and `NativeEventEmitter` for iOS to handle the payment result
2. You can retrieve value of result as follow:

```
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"  
"txn_not_allowed"  
"bank_back_pressed"
```

3. 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.
"txn_not_allowed"	It is a string constant and its value is "txn_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.

1. The below is detail response of payment

2.1 Success response json.

```
{
  "name_on_card": "Rizwan",
  "bank_ref_num": "214057035798",
  "udf3": "",
  "hash":
"5cd269f97088bf88ee9aef20864f6c86b738b4450851a7e518b21fc4bb2253ee153e6bd29ddc06a
adbc6a846d08b1452c488aed20f31c05ff6442894b994366a",
  "firstname": "Sagar",
  "net_amount_debit": "2.0",
  "payment_source": "Easebuzz",
  "surl": "http://localhost/paywitheasebuzz-php-lib-master/response.php",
  "error_Message": "Successful Transaction",
  "issuing_bank": "NA",
  "cardCategory": "NA",
  "phone": "8805596828",
  "easepayid": "E220520TZ7VGC2",
  "cardnum": "541919XXXXXX1220",
  "key": "D1K8SLB2XW",
  "udf8": "",
  "unmappedstatus": "NA",
  "PG_TYPE": "NA",
  "addedon": "2022-05-20 07:15:07",
  "cash_back_percentage": "50.0",
  "status": "success",
  "card_type": "Debit Card",
  "merchant_logo": "NA",
  "udf6": "",
  "udf10": "",
  "upi_va": "NA",
  "txnid": "EBZTXN0000357",
  "productinfo": "Test",
  "bank_name": "NA",
  "furl": "http://localhost/paywitheasebuzz-php-lib-master/response.php",
  "udf1": "",
  "amount": "2.0",
  "udf2": "",
  "udf5": "",
  "mode": "DC",
  "udf7": "",
  "error": "Successful Transaction",
  "udf9": "",
  "bankcode": "NA",
  "deduction_percentage": "0.0",
  "email": "sagar.sawant@easebuzz.in",
  "udf4": ""
}
```

2.2 Failure response json.

```
{
    "name_on_card": "Rizwan",
    "bank_ref_num": "",
    "udf3": "",
    "hash":
"bbf14e1a746abc6fba3267020d9d743c7952ad6c0927bacedac20b16eda0971ff734f98b15641805
5668a262f87b9e32b07476bca379992446359fd5bf1ce9d3",
    "firstname": "Sagar",
    "net_amount_debit": "20.0",
    "payment_source": "Easebuzz",
    "surl": "http://localhost/paywitheasebuzz-php-lib-master/response.php",
    "error_Message": "!ERROR!-GV00004-PARes status not sucessful.",
    "issuing_bank": "NA",
    "cardCategory": "NA",
    "phone": "8805596828",
    "easepayid": "E220520SIWNB81",
    "cardnum": "541919XXXXXX1220",
    "key": "D1K8SLB2XW",
    "udf8": "",
    "unmappedstatus": "NA",
    "PG_TYPE": "NA",
    "addledon": "2022-05-20 06:35:40",
    "cash_back_percentage": "50.0",
    "status": "failure",
    "card_type": "Debit Card",
    "merchant_logo": "NA",
    "udf6": "",
    "udf10": "",
    "upi_va": "NA",
    "txnid": "EBZTXN0000351",
    "productinfo": "Test",
    "bank_name": "NA",
    "furl": "http://localhost/paywitheasebuzz-php-lib-master/response.php",
    "udf1": "",
    "amount": "20.0",
    "udf2": "",
    "udf5": "",
    "mode": "DC",
    "udf7": "",
    "error": "!ERROR!-GV00004-PARes status not sucessful.",
    "udf9": "",
    "bankcode": "NA",
    "deduction_percentage": "0.0",
    "email": "sagar.sawant@easebuzz.in",
    "udf4": ""
}
```