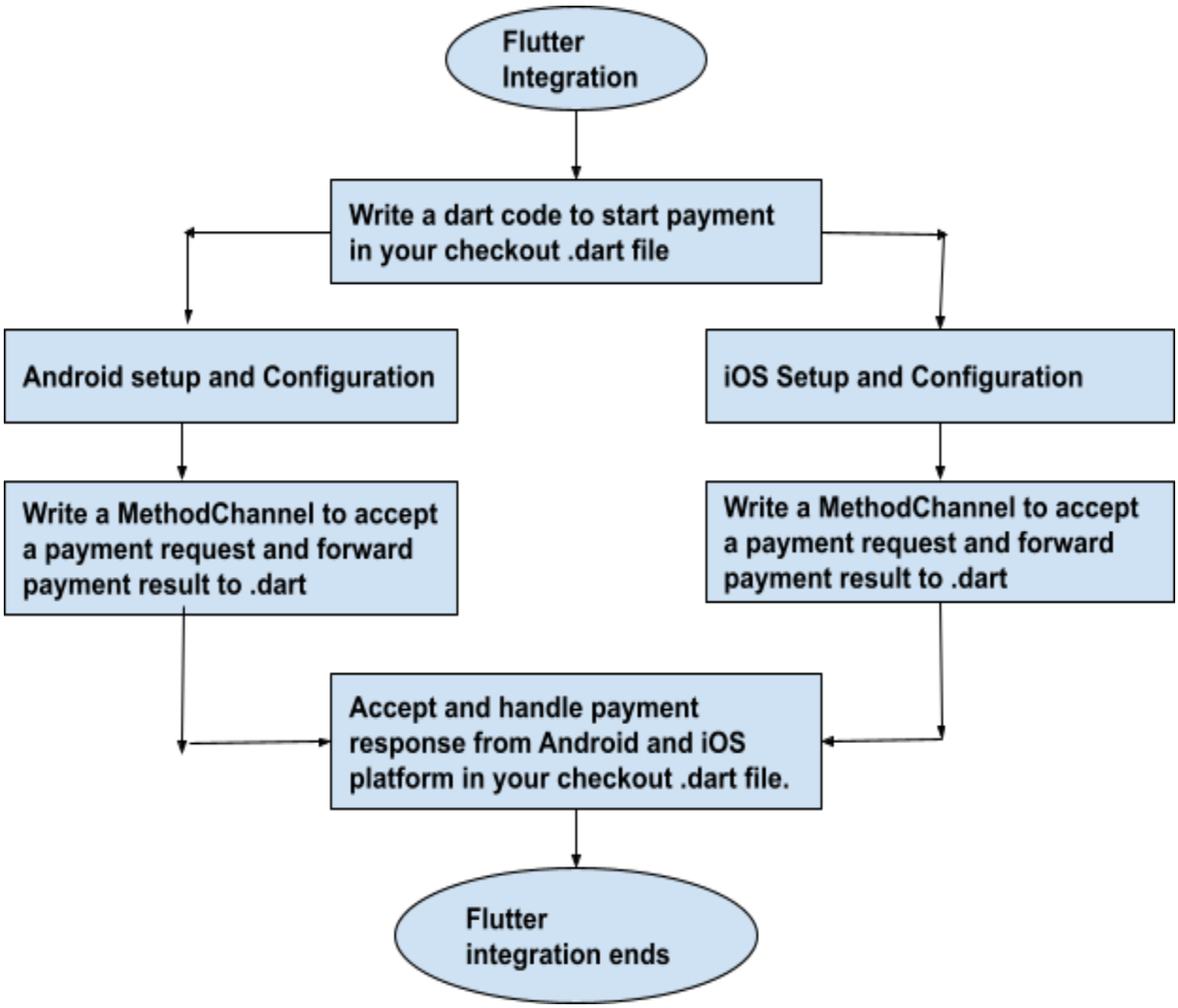
PayWithEaseBuzz Payment kit Integration (Flutter)



Flutter

The PaywithEaseBuzz Flutter 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 Flutter Integration Guide to get started.

This SDK allows you to integrate payments via PaywithEaseBuzz into your Flutter 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. EM

Requirements

1. The PaywithEaseBuzz Flutter SDK is supported for Android and iOS platforms.The PaywithEaseBuzz Flutter 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.

Android Setup

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

1. Copy peb-lib-android-x.aar file into app/libs/ folder of flutter 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 the 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"
}
```

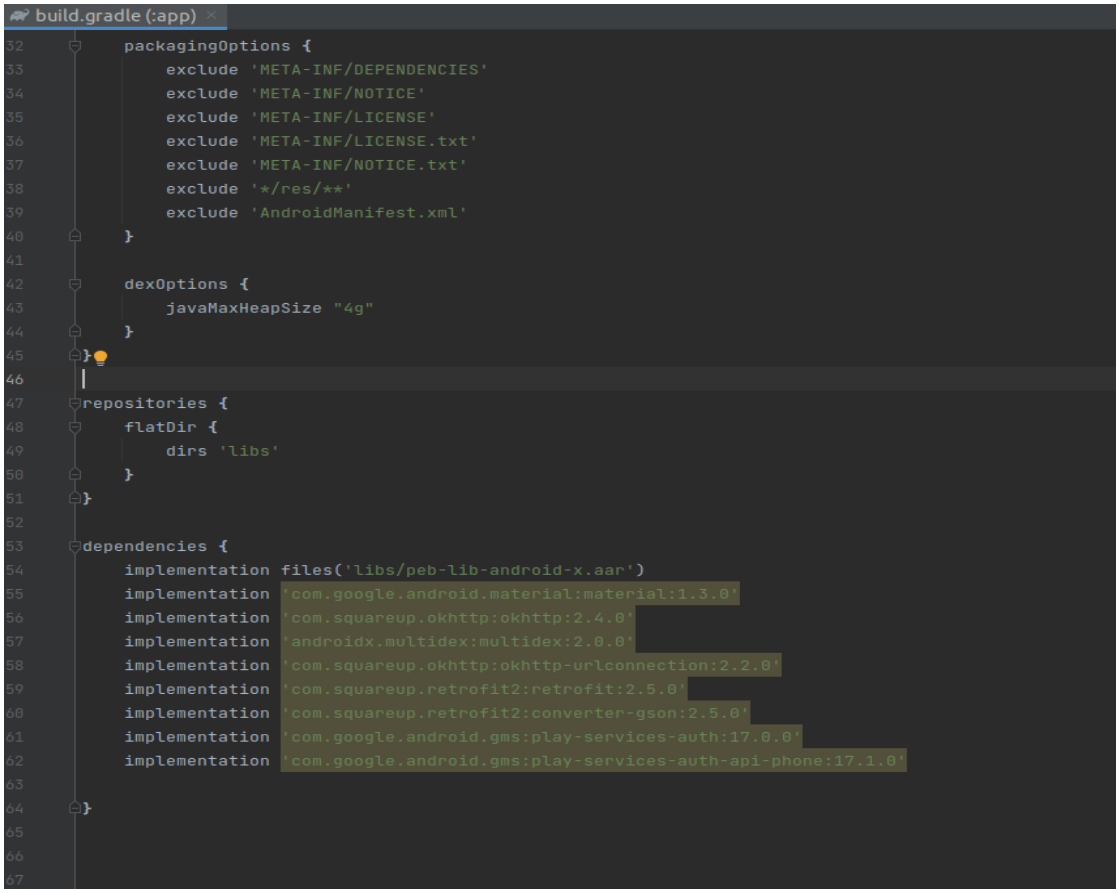
3.4 Add the repositories section as follows

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

3.5 Add the following Dependencies

```
implementation files('libs/peb-lib-android-x.aar')
implementation 'androidx.appcompat:appcompat:1.3.1'
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



3.6 To open the upi app, you must add the below lines outside <application> into AndroidManifest.xmlfile.

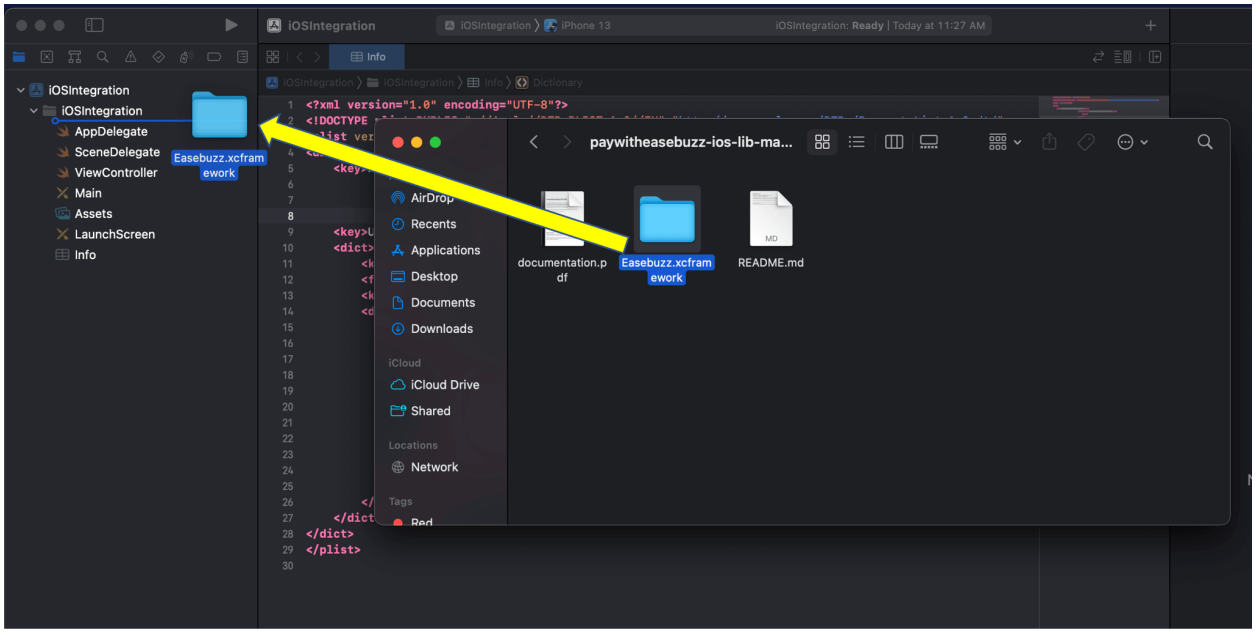
```
<queries>
   <package android:name="com.google.android.apps.nbu.paisa.user" />
   <package android:name="net.one97.paytm" />
   <package android:name="com.phonepe.app" />
   <package android:name="in.org.npci.upiapp" />
   <package android:name="in.amazon.mShop.android.shopping" />
   <package android:name="com.whatsapp"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

4. As per your respective native coding language add JsonConverter file into the directory where MainActivity file is located inside the android directory.

   Your android setup is done.

iOS Setup

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 *Swift Standard Libraries* to YES from project **Build Settings.**

```
ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES = YES
```

5. To simply disable ATS, you can follow this steps by open Info.plist, and add the following lines:

```
<key>NSAppTransportSecurity</key>
<dict> <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

6. To open UPI 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>
```

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.

Dart Code

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

1. To call the SDK inside your application, add the MethodChannel inside Class where the onPress() method is called.

```
static MethodChannel _channel = MethodChannel('easebuzz');
```

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 as below code.

```
async {
        String access_key = "Access key generated by the Initiate Payment API";
        String pay_mode = “This will either be “test” or “production";

        Object parameters = {
          "access_key":access_key,
          "pay_mode":pay_mode
        };
        final payment_response = await _channel.invokeMethod("payWithEasebuzz",
parameters)
        /* payment_response is the HashMap containing the response of the payment.
          You can parse it accordingly to handle 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 the code as per your native language file located in the android directory as below

   1.1 Declare the CHANNEL and MethodChannel result in MainActivity.java as below.

   Java (MainActivity.java) -

   ```
   private static final String CHANNEL = "easebuzz";
   MethodChannel.Result channel_result;
   private boolean start_payment = true;
   ```

   Kotlin (MainActivity.kt) -

   ```
   var channel_result: MethodChannel.Result? = null
   private var start_payment = true
   ```

   1.2 Set the MethodChannel handler in the onCreate() method of MainActivity.java as below.

   Java (MainActivity.java)

   ```
   start_payment = true;
   new MethodChannel(getFlutterEngine().getDartExecutor().getBinaryMessenger(),
   CHANNEL).setMethodCallHandler(
       new MethodChannel.MethodCallHandler() {
          @Override
          public void onMethodCall(MethodCall call, MethodChannel.Result result) {
             channel_result = result;
             if (call.method.equals("payWithEasebuzz")) {
                if (start_payment) {
                   start_payment = false;
                   startPayment(call.arguments);
                }
             }
          }
       });
   ```

   Kotlin (MainActivity.kt)

   ```
   start_payment = true
   MethodChannel(flutterEngine!!.dartExecutor.binaryMessenger,
   CHANNEL).setMethodCallHandler { call, result ->
     channel_result = result
     if (call.method == "payWithEasebuzz") {
        if (start_payment) {
           start_payment = false
           startPayment(call.arguments)
        }
   }
   }
   ```

1.3 Define startPayment() method which is declared inside MethodChannel as above step.

Java (MainActivity.java) -

```
private void startPayment(Object arguments) {
      try {
         Gson gson = new Gson();
         JSONObject parameters = new JSONObject(gson.toJson(arguments));
        try{
  Intent intentProceed = new Intent(getActivity(), PWECouponsActivity.class);
  Iterator<?> keys = parameters.keys();
  while(keys.hasNext() ) {
     String value = "";
     String key = (String) keys.next();
     value = parameters.optString(key);
          if (key.equals("amount")){
              Double amount = new Double(parameters.optString("amount"));
              intentProceed.putExtra(key,amount);
           } else {
              intentProceed.putExtra(key,value);
           }

  }

  startActivityForResult(intentProceed,PWEStaticDataModel.PWE_REQUEST_CODE );
} catch (Exception e){
  Toast.makeText(getActivity(), e.getMessage(), Toast.LENGTH_LONG).show();
}
      }catch (Exception e) {
         start_payment=true;
         Map<String, Object> error_map = new HashMap<>();
         Map<String, Object> error_desc_map = new HashMap<>();
         String error_desc = "exception occured:"+e.getMessage();
         error_desc_map.put("error","Exception");
         error_desc_map.put("error_msg",error_desc);
         error_map.put("result",PWEStaticDataModel.TXN_FAILED_CODE);
         error_map.put("payment_response",error_desc_map);
         channel_result.success(error_map);
      }
   }
```

Kotlin (MainActivity.kt)

```
private fun startPayment(arguments: Any) {
  try {
     val gson = Gson()
     val parameters = JSONObject(gson.toJson(arguments))
    try {
                val intentProceed = Intent(activity, PWECouponsActivity::class.java)
                val keys: Iterator<*> = parameters.keys()
                while (keys.hasNext()) {
                        var value: String? = ""
                         val key = keys.next() as String
                         value = parameters.optString(key)
                        if (key == "amount") {
                                val amount: Double = parameters.optDouble("amount")
                                intentProceed.putExtra(key, amount)
                         } else {
                                 intentProceed.putExtra(key, value)
                        }
                }
        startActivityForResult(intentProceed, PWEStaticDataModel.PWE_REQUEST_CODE)
     } catch (e: java.lang.Exception) {
        Toast.makeText(activity, e.message, Toast.LENGTH_LONG).show()
     }

  } catch (e: Exception) {
     start_payment = true
     val error_map: MutableMap<String, Any> = HashMap()
     val error_desc_map: MutableMap<String, Any> = HashMap()
     val error_desc = "exception occured:" + e.message
     error_desc_map["error"] = "Exception"
     error_desc_map["error_msg"] = error_desc
     error_map["result"] = PWEStaticDataModel.TXN_FAILED_CODE
     error_map["payment_response"] = error_desc_map
     channel_result!!.success(error_map)
  }
}
```

1.4 Write below code to fetch payment result and response which is to be forwarded to the flutter..

Java (MainActivity.java)

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(data != null ) {
        if(requestCode==PWEStaticDataModel.PWE_REQUEST_CODE)
        {
            start_payment=true;
            JSONObject response = new JSONObject();
            Map<String, Object> error_map = new HashMap<>();
            if(data != null ) {
                String result = data.getStringExtra("result");
                String payment_response = data.getStringExtra("payment_response");
                try {
                    JSONObject obj = new JSONObject(payment_response);
                    response.put("result", result);
                    response.put("payment_response", obj);
                    channel_result.success(JsonConverter.convertToMap(response));
                }catch (Exception e){
                    Map<String, Object> error_desc_map = new HashMap<>();
                    error_desc_map.put("error",result);
                    error_desc_map.put("error_msg",payment_response);
                    error_map.put("result",result);
                    error_map.put("payment_response",error_desc_map);
                    channel_result.success(error_map);
                }
            }else{
                Map<String, Object> error_desc_map = new HashMap<>();
                String error_desc = "Empty payment response";
                error_desc_map.put("error","Empty error");
                error_desc_map.put("error_msg",error_desc);
                error_map.put("result","payment_failed");
                error_map.put("payment_response",error_desc_map);
                channel_result.success(error_map);
            }
        }else
        {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
}
```

Kotlin (MainActivity.kt)

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent) {
  if (data != null) {
    if (requestCode == PWEStaticDataModel.PWE_REQUEST_CODE) {
      start_payment = true
      val response = JSONObject()
      val error_map: MutableMap<String, Any> = HashMap()
      if (data != null) {
        val result = data.getStringExtra("result")
        val payment_response = data.getStringExtra("payment_response")
        try {
          val obj = JSONObject(payment_response)
          response.put("result", result)
          response.put("payment_response", obj)
          channel_result!!.success(JsonConverter.convertToMap(response))
        } catch (e: Exception) {
          val error_desc_map: MutableMap<String, Any> = HashMap()
          /* Used the below code For target 30 api*/
          error_desc_map["error"] = result.toString()
          error_desc_map["error_msg"] = payment_response.toString()
          error_map["result"] = result.toString()
          /* End code For target 30 api*/
          error_map["payment_response"] = error_desc_map
          channel_result!!.success(error_map)
        }
      } else {
        val error_desc_map: MutableMap<String, Any> = HashMap()
        val error_desc = "Empty payment response"
        error_desc_map["error"] = "Empty error"
        error_desc_map["error_msg"] = error_desc
        error_map["result"] = "payment_failed"
        error_map["payment_response"] = error_desc_map
        channel_result!!.success(error_map)
      }
    } else {
      super.onActivityResult(requestCode, resultCode, data)
    }
  }
}
```

iOS Code

1. Initiate Payment Request.

   1.1 Import Easebuzz module in your AppDelegate/ ViewController.

   1.2 Set Delegate to your AppDelegate/ ViewController as PayWithEasebuzzCallback and Confirm the delegate.

   1.3 On clicking the Pay button from your app, you need to call the initiatePaymentAction method.

2. Refer below code for start payment gateway.

SWIFT : **copy below code and paste in AppDelegate.swift file. Please do not change Flutter method channel name and flutter method call name.**

```
import UIKit
import Flutter
import Easebuzz

@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate,PayWithEasebuzzCallback {
   var payResult:FlutterResult!
   override func application(
      _ application: UIApplication,
      didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey:
Any]?
      ) -> Bool {
      self.initializeFlutterChannelMethod()
      return super.application(application, didFinishLaunchingWithOptions: launchOptions)
   }
   // Initialize flutter channel
   func initializeFlutterChannelMethod()  {
      GeneratedPluginRegistrant.register(with: self)
      guard let controller = window?.rootViewController as? FlutterViewController else {
         fatalError("rootViewController is not type FlutterViewController")
      }

      let methodChannel = FlutterMethodChannel(name: "easebuzz",
                                               binaryMessenger: controller)
      methodChannel.setMethodCallHandler({
         [weak self] (call: FlutterMethodCall, result: @escaping FlutterResult) -> Void in
         guard call.method == "payWithEasebuzz" else {
            result(FlutterMethodNotImplemented)
            return
         }
         self?.payResult = result;
         self?.initiatePaymentAction(call: call);
```

```
            })
        }

        // Initiate payment action and call payment gateway
        func initiatePaymentAction(call:FlutterMethodCall) {
            if let orderDetails = call.arguments as? [String:String]{
                let payment = Payment.init(customerData: orderDetails)
                let paymentValid = payment.isValid().validity
                if !paymentValid {
                    print("Invalid records")
                } else{
                    PayWithEasebuzz.setUp(pebCallback: self )
                    PayWithEasebuzz.invokePaymentOptionsView(paymentObj: payment, isFrom:
self)
                }
            }else{
                // handle error
                let dict =  self.setErrorResponseDictError("Empty error", errorMessage: "Invalid
validation", result: "Invalid request")
                self.payResult(dict)
            }
        }

        // payment call callback and handle response
        func PEBCallback(data: [String : AnyObject]) {
            if data.count > 0 {
                self.payResult(data)
            }else{
                let dict =  self.setErrorResponseDictError("Empty error", errorMessage: "Empty
payment response", result: "payment_failed")
                self.payResult(dict)
            }
        }

        // Create error response dictionary that the time of something went  wrong
        func setErrorResponseDictError(_ error: String?, errorMessage: String?, result: String?)
-> [AnyHashable : Any]? {
            var dict: [AnyHashable : Any] = [:]
            var dictChild: [AnyHashable : Any] = [:]
            dictChild["error"] = "\(error ?? "")"
            dictChild["error_msg"] = "\(errorMessage ?? "")"
            dict["result"] = "\(result ?? "")"
            dict["payment_response"] = dictChild
            return dict
        }
    }
```

Objective C : copy below code and paste in AppDelegate.m file

```
#include "AppDelegate.h"
#include "GeneratedPluginRegistrant.h"
#import

@implementation AppDelegate
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [self initialisePaywithEasebuzz];
    return [super application:application didFinishLaunchingWithOptions:launchOptions];
}
// Initiate method
-(void)initialisePaywithEasebuzz{
    [GeneratedPluginRegistrant registerWithRegistry:self];
    FlutterViewController* controller =
    (FlutterViewController*)self.window.rootViewController;
    FlutterMethodChannel* methodChannel = [FlutterMethodChannel
                                    methodChannelWithName:@"easebuzz"
                                    binaryMessenger:controller];
    __weak typeof(self) weakSelf = self;
    [methodChannel setMethodCallHandler:^(FlutterMethodCall* call,
                                    FlutterResult result) {
        NSLog(@"call kit = %@",call.method);
        self.payResult = result;
        if ([@"payWithEasebuzz" isEqualToString:call.method]) {
            [weakSelf initiatePaymentAction:call];
        } else {
            result(FlutterMethodNotImplemented);
        }
    }];
}

// Initialize payment gateway
-(void)initiatePaymentAction:(FlutterMethodCall*)call {
    NSDictionary *orderDetails1 = [NSDictionary dictionaryWithDictionary:call.arguments];
    NSLog(@"%@",orderDetails1);
    self.payment = [[Payment alloc]initWithCustomerData:orderDetails1];
    BOOL paymentValid = _payment.isValid;
    if (!paymentValid) {
        NSDictionary *dict = [self setErrorResponseDictError:@"Empty error"
errorMessage:@"Invalid validation" result:@"Invalid request"];
        if (dict != nil) {
            self.payResult(dict);
        }
    } else {
        [PayWithEasebuzz setUpWithPebCallback:self];
        [PayWithEasebuzz invokePaymentOptionsViewWithPaymentObj:_payment
isFrom:self];
```

```
    }
}

// Call back delegate from the paywitheasebuzz gateway
- (void)PEBCallbackWithData:(NSDictionary * _Nonnull)data {
    @try {
        if (data != nil) {
            self.payResult(data);
        }else{
            NSDictionary *dict = [self setErrorResponseDictError:@"Empty error"
errorMessage:@"Empty payment response" result:@"payment_failed"];
            if (dict != nil) {
                self.payResult(dict);
            }
        }
    }
    @catch (NSException *exception) {
        NSString *str = [NSString stringWithFormat:@"exception
occured:%@",exception.reason];
        NSDictionary *dict = [self setErrorResponseDictError:@"Exception" errorMessage:str
result:@"payment_failed"];
        if (dict != nil) {
            self.payResult(dict);
        }
    }
    @finally {
    }
}

// Create error response dictionary that the time of something went  wrong
-(NSDictionary *)setErrorResponseDictError:(NSString *)error
errorMessage:(NSString*)errorMessage result:(NSString*)result{
    NSMutableDictionary *dict = [[NSMutableDictionary alloc]init];
    NSMutableDictionary *dictChild = [[NSMutableDictionary alloc]init];
    dictChild[@"error"] = [NSString stringWithFormat:@"%@",error];
    dictChild[@"error_msg"] = [NSString stringWithFormat:@"%@",errorMessage];
    dict[@"result"] = [NSString stringWithFormat:@"%@",result];
    dict[@"payment_response"] = dictChild;
    return dict;
}
@end
```

Handle Payment Response

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

1. Below method you have used to start the payment and fetch response in .dart file:

   final Map response = await _channel.invokeMethod("payWithEasebuzz", parameters);

   In the above code Map response is the HashMap contains the response of payment.

2. You can retrieve value of result as follow

   String result = response['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"

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

   String detailed_response = response['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 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",
        "addedon": "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": ""
 }
```