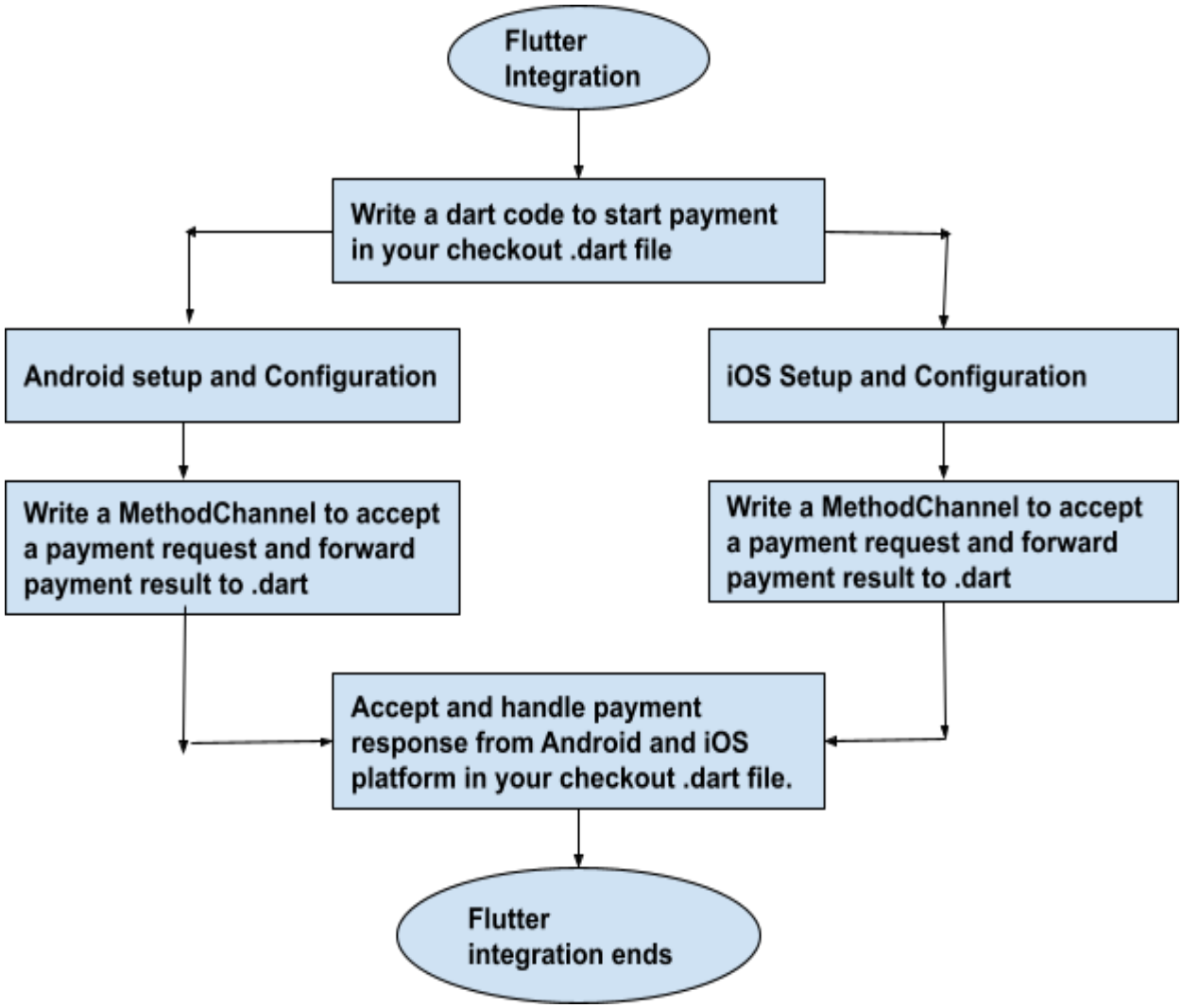
PayWithEaseBuzz Payment kit Integration (Flutter)



- **Initiate Payment :** Generate access key using the Initiate Payment API at your backend.
  (It is mandatory to integrate the Initiate Payment API at Backend only)
  Initiate Payment API Doc : https://docs.easebuzz.in/api/initiate-payment

- **Dart Setup:**
  1. Write the below **.dart** code in your checkout file (On click of pay button)
     1.1: Write MethodChannel declaration as below.

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

     1.2: Write below code to start payment.

```
async {
        String access_key = "Access key generated by the initiate payment API";
        String pay_mode="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 result of payment.
```

- **Android Setup:**
  1. Copy **peb-lib.aar** file into android/app/libs/ folder of your flutter application.
  2. Proguard Rules configurations:
     Add below line to your proguard rules.

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

  3. Build.gradle(app) modifications.
     Add this line to build.gradle (app)

```
defaultConfig {
        multiDexEnabled true
    }
```

     Add the following lines to packagingOptions,

```
exclude 'META-INF/DEPENDENCIES'
exclude 'META-INF/NOTICE'
exclude 'META-INF/LICENSE'
exclude 'META-INF/LICENSE.txt'
exclude 'META-INF/NOTICE.txt'
```

     Add the following line to dexOptions.

```
javaMaxHeapSize "4g"
```

Add repositories section as follows.

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

Add the following dependencies

1.compile(name: 'peb-lib', ext: 'aar')
2. compile 'com.android.support:appcompat-v7:28.0.0'
3. compile 'com.android.support:design:28.0.0'
8. compile 'com.squareup.okhttp:okhttp:2.4.0'
9. compile 'com.squareup.okhttp:okhttp-urlconnection:2.2.0'
10. compile 'com.squareup.retrofit2:retrofit:2.3.0'
11. compile 'com.squareup.retrofit2:converter-gson:2.3.0'

3. Add **JsonConverter.java** in the directory where MainActivity.java is located in the android directory.

4. Modify your MainActivity.java located in the android directory as below.

4.1. Declare the below variables.

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

4.2. Write below code and Set the MethodChannel handler in onCreate() of MainActivity.java as below.

```
start_payment = true;
new MethodChannel(getFlutterView(), 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);
                 }
             }
            }
});
```

4.2. Define startPayment() method which is called in above onCreate method()

```
private void startPayment(Object arguments) {
  try {
       Gson gson = new Gson();
       JSONObject parameters = new JSONObject(gson.toJson(arguments));
       Intent intentProceed = new Intent(getBaseContext(), PWECouponsActivity.class);
       intentProceed.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
       intentProceed.putExtra("access_key",parameters.getString("access_key"));
       intentProceed.putExtra("pay_mode",parameters.getString("pay_mode"));
       startActivityForResult(intentProceed, PWEStaticDataModel.PWE_REQUEST_CODE);
   } 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);
    }
}
```

4.3. Write below code to catch payment result and forward to the flutter.

```
@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));
                channel_result.success(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);
        }
    }
}
```

5. And your Android Setup is done.

- ## iOS Setup :

The PaywithEaseBuzz iOS SDK is compatible with apps supporting iOS 9 and above
And requires Xcode 9 to build from source.

a. Copy easebuzz.framework of your application in embedded binaries.
b. Press + and add framework using 'Add other' button.
c. Browse framework: file from your folder and select 'copy items if needed'.
d. Set Always embed swift standard libraries to YES from project build settings

ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES = YES

e. 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>
```

➢ **Initiate Payment Request**

1. Import Easebuzz module in your AppDelegate/ ViewController
2. Set Delegate to your AppDelegate/ ViewController as PayWithEasebuzzCallback and Confirm the delegate.
3. On clicking the Pay button from your app, you need to call initiatePaymentAction method.

Refer below code for calling 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)
  }
  // Initialise 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 <Flutter/Flutter.h>

@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<NSString *,id> * _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
```

Remove unused architectures -

1. Easebuzz is a custom universal framework and for in on production, we need to remove unused architectures. Because Apple doesn't allow the application with unused architectures to the App Store.
2. Select the Project, Choose Target → Project Name → Select Build Phases → Press "+" → New Run Script Phase → Name the Script as "Run Script".
3. Always this script should be placed below "Embed Frameworks".
4. Always build the project for both simulator and generic device build before start the archives.
5. Run the below script to remove the unused simulator architectures at the time of pushing the App to App Store.

```
APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"

# This script loops through the frameworks embedded in the application and
# removes unused architectures.
find "$APP_PATH" -name 'Easebuzz.framework' -type d | while read -r FRAMEWORK
do
FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist"
    CFBundleExecutable)
FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"

EXTRACTED_ARCHS=()
```

```
for ARCH in $ARCHS
do
echo "Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH" -o
      "$FRAMEWORK_EXECUTABLE_PATH-$ARCH"
EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
done

echo "Merging extracted architectures: ${ARCHS}"
lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged" -create "${EXTRACTED_ARCHS[@]}"
rm "${EXTRACTED_ARCHS[@]}"

echo "Replacing original executable with thinned version"
rm "$FRAMEWORK_EXECUTABLE_PATH"
mv "$FRAMEWORK_EXECUTABLE_PATH-merged" "$FRAMEWORK_EXECUTABLE_PATH"

done
```