# **iOS SDK Flutter Brief recommendation Guide**

Summary

* IOS SDK is developed in Swift version 4.2 for native apps built in Swift or Objective C using Objective C bridging header
* Flutter apps can use the iOS SDK using some tweaks you need to make sure you apply
* This document is just a reference, made by the iOS developer without Flutter skills, based in documentation from docs.flutter.dev

Recommendation for Flutter apps:

Please visit the flutter documentation for using native code: <https://docs.flutter.dev/development/platform-integration/platform-channels#separate>

Please see the SDK documentation:

<https://github.com/GeideaSolutions/iOS-SDK/blob/main/Payment%20Gateway%20IOS%20SDK%202.0%20Integration%20guide%20docx.docx>

1. You need to have a MacBook with Xcode installed
2. You need to download the iOSSDK XCFramework, by right clik and save file as option found here: <https://github.com/GeideaSolutions/iOS-SDK/tree/main/GeideaPaymentSDK>
3. Add the framework to your project:

To connect iOS framework to your iOS application you should:

1. Drag GeideaPaymentSDK.XCframework to your Frameworks folder (Contains both devices and simulators) – Preferred

OR Drag GeideaPaymentSDK.framework to your Frameworks folder (need devices and Fat framework)

1. Add it your target: General -> Frameworks, Libraries and and Embedded Content.
2. Choose “Embed & Sign” option on Embed tab

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. On your FlutterAppDelegate You need to import GeideaPaymentSDK as follow

Import GeideaPaymentSDK

import GeideaPaymentSDK

@UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

GeneratedPluginRegistrant.register(with: self)

return super.application(application, didFinishLaunchingWithOptions: launchOptions)

}

1. On your FlutterAppDelegate You need to connect to our GeideaPaymentSDK by setting the credentials you have been provided by Geidea. The credentials will be saved in the device KeyChain so we recommend to set it only once. If you need to change the credentials you need just to comment the check for isCredentialsAvailable. You can also set the language for English or Arabic. Default is English

import GeideaPaymentSDK

@UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

override func application(

\_ application: UIApplication,

didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

if !GeideaPaymentAPI.isCredentialsAvailable() {

GeideaPaymentAPI.setCredentials(withMerchantKey: “your Key”

andPassword: “Your Password”)

}

GeideaPaymentAPI.setLanguage(Language.arabic)

GeneratedPluginRegistrant.register(with: self)

return super.application(application, didFinishLaunchingWithOptions: launchOptions)

}

}

1. On Your FlutterAppDelegate you need to get your merchantConfig object (GDConfigResponse) and save it in AppDelegate instance

@UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

var merchantConfig: GDConfigResponse?

override func application(

\_ application: UIApplication,

didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

if !GeideaPaymentAPI.isCredentialsAvailable() {

GeideaPaymentAPI.setCredentials(withMerchantKey: “your Key”

andPassword: “Your Password”)

}

GeideaPaymentAPI.setLanguage(Language.arabic)

GeideaPaymentAPI.getMerchantConfig(completion: response,error in {

guard let config = response else { return}

merchantConfig = config

})

GeneratedPluginRegistrant.register(with: self)

return super.application(application, didFinishLaunchingWithOptions: launchOptions)

}

}

1. At this step you have the setup for the SDK connection, and you got your merchantConfig needed for most features of the SDK. Now you need to have a channel for communication between your Flutter code and iOS SDK and also an UIViewController instance from flutter needed for all payment features. We need this viewController because we use your context to present over your app screen our UI.

@UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

var merchantConfig: GDConfigResponse?

override func application(

\_ application: UIApplication,

didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

if !GeideaPaymentAPI.isCredentialsAvailable() {

GeideaPaymentAPI.setCredentials(withMerchantKey: “your Key”

andPassword: “Your Password”)

}

GeideaPaymentAPI.setLanguage(Language.arabic)

GeideaPaymentAPI.getMerchantConfig(completion: response,error in {

guard let config = response else { return}

merchantConfig = config

})

let controller: FlutterViewController = window?.rootViewController as! FlutterViewController

let channel = FlutterMethodChannel(name: “native”, binaryMessenger: controller.binaryMessenger)

channel.setMethodCallHandler({

[weak self] (call: FlutterMethodCall, result: FlutterResult) -> Void in

)}

GeneratedPluginRegistrant.register(with: self)

return super.application(application, didFinishLaunchingWithOptions: launchOptions)

}

}

UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

1. Now you have everything you need to just use the SDK functions by calling SDK APIs provided. Example paying with GeideaForm:

This sample is just for referencing made by iOS developer without Flutter skills. Please fill the gaps by having user input details from your Flutter app. Also check the paranthesis because the snippet was not compiled to check.

@UIApplicationMain

@objc class AppDelegate: FlutterAppDelegate {

var merchantConfig: GDConfigResponse?

override func application(

\_ application: UIApplication,

didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

if !GeideaPaymentAPI.isCredentialsAvailable() {

GeideaPaymentAPI.setCredentials(withMerchantKey: “your Key”

andPassword: “Your Password”)

}

GeideaPaymentAPI.setLanguage(Language.arabic)

GeideaPaymentAPI.getMerchantConfig(completion: response,error in {

guard let config = response else { return}

merchantConfig = config

})

let controller: FlutterViewController = window?.rootViewController as! FlutterViewController

let channel = FlutterMethodChannel(name: “native”, binaryMessenger: controller.binaryMessenger)

channel.setMethodCallHandler({

[weak self] (call: FlutterMethodCall, result: FlutterResult) -> Void in

case: “payment” {

let amount = GDAmount(amount:4.40, currency: “SAR”)

let cardDetails = GDCardDetails(withCardHolderName: “User Input card name”, andCardNumber: “User card number”, andCVV: “User Input CVV”, andExpiryMonth: “User Input expiry month”, andExpiryYear: “User Input expiry year” )

GeideaPaymentAPI.payWithGeideaForm(theAmount: amount, showAddress:false, showEmail: false, showReceipt: true, customerDetails: nil, config: self.merchantConfig, viewController: controller, completion: response, error in {

if let err = error {

// payment failed. Please check error fields GDErrorResponse for relevan fields

} else {

// payment success please check GDOrderResponse for relevant fields

}

)}

}

)}

GeneratedPluginRegistrant.register(with: self)

return super.application(application, didFinishLaunchingWithOptions: launchOptions)

}

}

1. If you want to use Apple Pay features please be sure to configure setupApplePay with your controller from FlutterViewController and make sure your button meets Apple design guidelines, to increase chances of accepting the app, here:

<https://developer.apple.com/design/human-interface-guidelines/apple-pay/overview/buttons-and-marks/>

!Important You need to setup your app for applepay by enabling applePay described here:

To pay the amount with **Apple Pay** you need to have the apple pay marchantId configured on your apple account and to enable apple pay capabilities. Please follow this for setup:

Geidea will give you a CSR file for cerificate request

<https://developer.apple.com/documentation/passkit/apple_pay/setting_up_apple_pay_requirements>

Example:

let amount = GDAmount(amount:4.40, currency: “SAR”)

let applePayDetails = GDApplePayDetails (in viewController, andButtonIn: nil, forMerchantIdentifier: “merchant.your.merchant”, paymentMethods: nil”)

GeideaPaymentApi.setupApplePay(forApplePayDetails: applePayDetails, with: amount, config: merchantConfig, completion: response, error in {

})