



BasisPay Payment Gateway iOS SDK Integration Kit

044 4018 4444 | info@basispay.in | basispay.in

New # 9, Old # 11, 1st Floor, Palayakaran Street, Kalaimagal Nagar, Ekkatuthangal, Chennai, Tamil Nadu 600032

Thanks for choosing BasisPay as your payment facilitator for your business. This document helps business owners or developers to understand how to integrate BasisPay payment gateway in iOS Platform. If you have any queries or clarifications, please email to support@basispay.in or visit www.tychepayment.com

Integration Kit

Steps to integrate the Basispay iOS Standard SDK are given below:

1. Import the Basispay iOS Standard SDK Library
2. Initialize the Basispay iOS Standard SDK and Pass default parameters
3. Pass Payment Options and Display Checkout Form
4. Handle Success and Errors Events
5. Fetch Response

STEP 1: Import the Basispay iOS Standard SDK Library

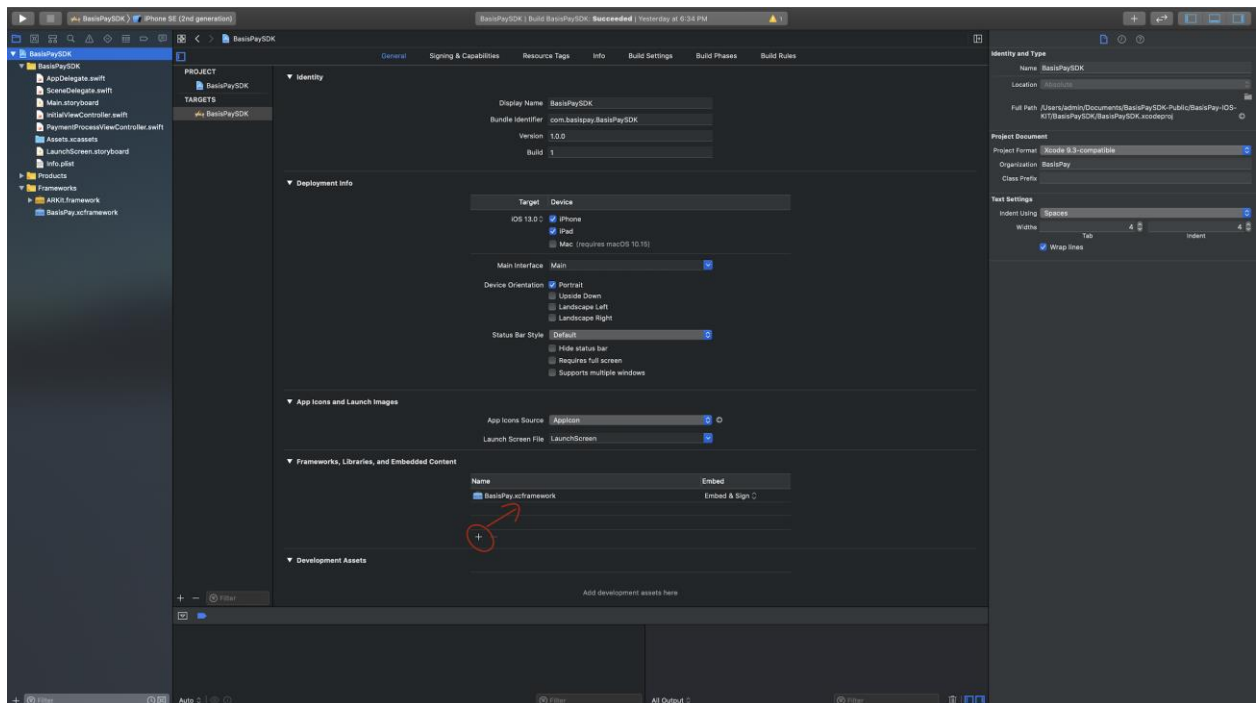
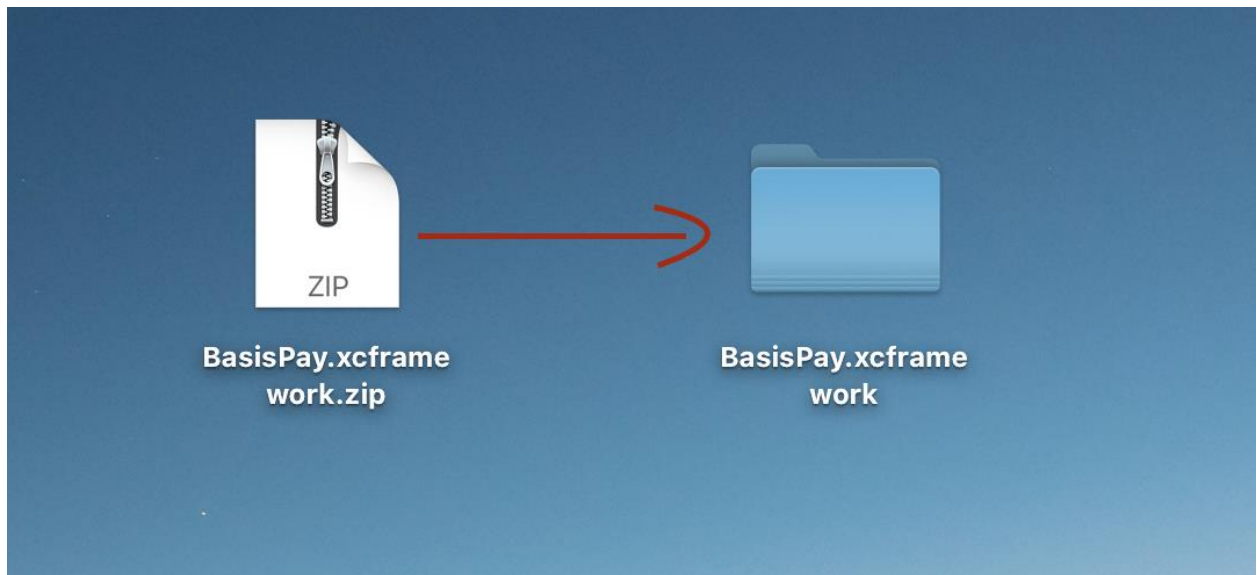
You can import the BasisPay iOS SDK library in two ways. The methods are as follows. If you are familiar with CocoaPod, then you can follow the steps provided in the section called "CocoaPod", after completing it Cocoapod steps you can proceed with step 3 which is given in the document. If you are familiar with Swift framework you can follow steps from this document itself.

CocoaPod:

Checkout our [Cocoapod](#) (bitcode enabled) pod to integrate into your app.

Swift:

1. Download the [SDK](#) and unzip it.
2. Open your project in XCode and go to **Target** and click General **tab** . Go to **Frameworks,libraries, and Embedded Content** and Select **Add other to "yourproject"**.
3. Select **Basispay.xcframework** in the directory you just unzipped.
4. Click **Add**.



STEP 2: Initialize the Basispay iOS Standard SDK

To initialize the Basispay iOS Standard SDK, you need the following parameter keys which can be generated from the BasisPay team:

- API key
- Salt key
- Endpoint

Create a global instance of the `PaymentGatewayViewController` of the framework, and add it as a sub view to your view controller.

```

import UIKit

import BasisPay

class PaymentProcessViewController: UIViewController {

    var paymentGatewayViewController: PaymentGatewayController!

    var amount:String?

    var titleValue:String?

    var descriptionValue:String?

    @IBOutlet weak var viewContainer: UIView!

    override func viewDidLoad() {

        super.viewDidLoad()

        paymentGatewayViewController = PaymentGatewayController()

        viewContainer.addSubview(paymentGatewayViewController.view)

        setDefaults()

        setInputDictionary()

        checkAndGetResponse()

    }

```

```

9  import UIKit
10 import BasisPay
11
12 class PaymentProcessViewController: UIViewController {
13
14     var paymentGatewayViewController: PaymentGatewayController!
15     var amount:String?
16     var titleValue:String?
17     var descriptionValue:String?
18     @IBOutlet weak var viewContainer: UIView!
19     override func viewDidLoad() {
20         super.viewDidLoad()
21         paymentGatewayViewController = PaymentGatewayController()
22         viewContainer.addSubview(paymentGatewayViewController.view)
23         setDefaults()
24         setInputDictionary()
25         checkAndGetResponse()
26     }
27
28     //MARK: Default keys will be provided by the BasisPay support team
29     // API key
30     // Salt key
31     // return url
32     // Endpoint - just provide .Testing or .Live
33     private func setDefaults() {
34         paymentGatewayViewController.paymentDefaults = PaymentDefaults(apiKey: "", saltKey: "", returnUrl: "", endPoint:
35             .Testing)
36         paymentGatewayViewController.delegate = self
37     }
38

```

To provide payment details as inputs to framework, use the following method: Provide the default information using,

```
//MARK: Default keys

// API keys

// Salt keys

// return url

// Endpoint

private func setDefaults() {

    paymentGatewayViewController.paymentDefaults = PaymentDefaults(apiKey: "", saltKey: "", returnUrl:
"", endPoint: .Testing) }
```

| Parameter Name | Description |
|------------------|--|
| apikey | get the “Api Key” from the BasisPay Team and use it here. (Mandatory*) |
| saltkey | Get the “Salt Key” from the BasisPay Team and use it here. (Mandatory*) |
| returnUrl | Create a new empty url and use it here. The response will be updated using this. (Mandatory*) For an example http://yourdomainname.com/paymentresponse |
| endpoint | This is to specify if the project is in a ‘Live’ or ‘Testing’ environment. .live for ‘Live’ and .Testing for ‘Testing’. (Mandatory*) |

STEP 3: Pass Default Parameters

Provide payment input details using,

```
let paymentRequestDictionary: NSDictionary = [
    "orderId" : <Order id of the payment>,
    "amount" : <This is the payment amount>,
    "currency" : <This is the 3-digit currency code (INR)>,
    "description" : <Brief description of product or service that the customer is being charged for>,
    "name" : <Name of customer>,
    "email" : <Customer email address>, "phone" : <Customer phone number>, "addressLine1" : <Customer address>, "addressLine2" : <Customer address 2>, "city" : <Customer city>
    "state" : <Customer State>,
    "country" : <Customer country>,
    "zipCode" : <Customer zip code>,
    "udf1" : <User defined field>,
    "udf2" : <User defined field 2>,
    "udf3" : <User defined field 3>,
    "udf4" : <User defined field 4>,
    "udf5" : <User defined field 5>
]
```

```
private func setInputDictionary() {
    guard let amountVal = amount, let titleVal = titleValue, let descriptionVal = descriptionValue else {
        return
    }

    let paymentRequestDictionary: NSDictionary = [
        "orderId" : "253@98",
        "amount" : amountVal,
        "currency" : "INR",
        "description" : descriptionVal,
        "name" : titleVal,
        "email" : "qwerty@123gmail.com",
        "phone" : "5876986087",
        "addressLine1" : "address_line_1",
        "addressLine2" : "address_line_2",
        "city" : "city",
        "state" : "state",
        "country" : "country",
        "zipCode" : "zip_code",
        "udf1" : "Testing1",
        "udf2" : "Testing2",
        "udf3" : "Testing3",
        "udf4" : "Testing4",
        "udf5" : "Testing5"
    ]
    paymentGatewayViewController.setInputDictionary(inputDictionary: paymentRequestDictionary)
}
```

| Parameter Name | Description |
|---------------------|--|
| orderId | Order id of the payment (Mandatory*) |
| amount | This is the payment amount (Mandatory*) |
| currency | This is the 3-digit currency code (INR) (Mandatory*) |
| description | Brief description of product or service that the customer is being charged for (Mandatory*) |
| name | Name of customer (Mandatory*) |
| email | Customer email address (Mandatory*) |
| phone | Customer phone number (Mandatory*) |
| addressLine1 | Customer address (Mandatory*) |
| addressLine2 | Customer address 2 (Optional) |
| city | Customer city (Mandatory*) |
| state | Customer State (Mandatory*) |
| country | Customer country (Mandatory*) |
| zipCode | Customer zip code (Mandatory*) |
| udf1 | User defined field (Optional) |
| Udf2 | User defined field2 (Optional) |
| Udf3 | User defined field3 (Optional) |
| Udf4 | User defined field4 (Optional) |
| Udf5 | User defined field5 (Optional) |

Note: Use the key names as the same

STEP 4: Handle Success and Errors Events

You can handle success or error events when a payment is completed by implementing `onPaymentSuccess` and `onPaymentFailure` methods of the **PaymentGatewayDelegate**.

```
paymentGatewayViewController.delegate = self
```

```
extension PaymentProcessViewController:PaymentGatewayDelegate {
```

```
    func onPaymentSuccess(orderId: String, description: String) {
```

```

        self.navigationController?.popViewController(animated: true)

        let alertController = UIAlertController(title: "SUCCESS", message: "order Id \ \(orderId)", preferredStyle:
.alert)

        let cancelAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

        alertController.addAction(cancelAction)

        self.view.window?.rootViewController?.present(alertController, animated: true, completion: nil)
    }

```

```

func onPaymentFailure(description: String) {

    self.navigationController?.popViewController(animated: true)

    let alertController = UIAlertController(title: "FAILURE", message: " message - \ \(description)",
preferredStyle: .alert)

    let cancelAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

    alertController.addAction(cancelAction)

    self.view.window?.rootViewController?.present(alertController, animated: true, completion: nil)
}

```

```

extension PaymentProcessViewController:PaymentGatewayDelegate {

    func onPaymentSucess(orderId: String, description: String) {
        self.navigationController?.popViewController(animated: true)
        let alertController = UIAlertController(title: "SUCCESS", message: "order Id \ \(orderId)", preferredStyle: .alert)
        let cancelAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
        alertController.addAction(cancelAction)
        self.view.window?.rootViewController?.present(alertController, animated: true, completion: nil)
    }

    func onPaymentFailure(description: String) {
        self.navigationController?.popViewController(animated: true)
        let alertController = UIAlertController(title: "FAILURE", message: " message - \ \(description)", preferredStyle: .alert)
        let cancelAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
        alertController.addAction(cancelAction)
        self.view.window?.rootViewController?.present(alertController, animated: true, completion: nil)
    }

}

```

STEP 5: Fetch the Response

func getResponseData() -> NSDictionary - This method returns the response Dictionary.

func isResponseAvailable() -> Bool - This method is used to check if the response of payment transaction is available or not. If the response is not available, then do a response check with some delay. If response is available, use it.

```
@objc func checkAndGetPaymentResponse() {  
    if (paymentGatewayController.isResponseAvailable()) {  
        let responseData = paymentGatewayController.getResponseData()  
        processResponseData(responseDictionary: responseData)  
    }  
    else{  
        perform(#selector(checkAndGetPaymentResponse), with: nil, afterDelay: 2)  
    }  
}
```

```
@objc func checkAndGetResponse() {  
    if (paymentGatewayViewController.isResponseAvailable()) {  
        let responseData = paymentGatewayViewController.getResponseData()  
        print(responseData) }  
    else {  
        perform(#selector(checkAndGetResponse), with: nil, afterDelay: 2) }  
}
```

| Parameter Name | Description |
|----------------|---|
| address_line_1 | Customer address |
| address_line_2 | Customer address 2 |
| amount | This is the payment amount |
| cardmasked | This shows the state of card |
| city | Customer city |
| country | Customer country |
| currency | This is the 3-digit currency code |
| description | Brief description of product or service that the customer is being charged. |

| | |
|-------------------------|---|
| email | Customer email address |
| error_desc | Shows the error description. |
| name | Name of customer |
| order_id | Order id of the payment |
| payment_channel | Bank name |
| payment_datetime | Transaction completion time |
| payment_mode | Shows the mode of payment like UPI, Netbanking etc. |
| phone | Customer number |
| response_code | 0 - Transaction successful 1000 - Transaction failed |
| response_message | Shows Transaction status |
| state | Customer state |
| transaction_id | Customer Transaction Id |
| zip_code | Customer zip code |

If you are facing any issues while integration you can email to support@basispay.in