Mobile: iOS SDK

Integration guide

1. Get the API key

During your onboarding process, your dedicated integration manager will provide you with your unique API key. If at any time you lose your API key, please reach out to your dedicated integration manager to reset your API key.
You can also find the API key in the Partner Portal under the "For developers" section.

2. Build Integration

Now that you have an API key you are able to set up crypto purchases via Widget in several ways. You can build the integration with a quote or without a quote.

Integration without a quote

When you build an integration without a quote it means you don't show your customer a quote for purchase in your UI. Your customer starts his journey without predefined transaction details. It is a simple integration and you can implement it by creating simple CTA (call to action) buttons in your UI. Your customer will be able to select everything (cryptocurrency, fiat currency, amount, etc) in the widget and his journey will start from an exchange form.

To implement this type of integration you will need to:

- Create a *requestID* with an empty quote. You still have an option to add parameters when creating the *requestID*. You can read more about it in Create request.
- After you receive the *requestID* you can initiate the iOS widget with it. Please read the iOS widget installation guide here.

Integration with a quote

You can also build an integration with a quote. It is a bit more complicated than the integration without a quote because you have to use one more API endpoint. This also means you will need to spend more time developing your UI. This integration is useful if you want to allow your customer to see transaction details in your UI. It is also being used when implementing several on-ramp

solutions and allowing your customers to compare how much they get for their money from different providers. When you implement this type of integration your customer will start their journey with predefined transaction values like transaction amount, cryptocurrency, fiat currency, etc.

To implement this type of integration you will need to:

- Create a quote first. Please read more about how to create a quote here.
- Create a *requestID* with the *quoteID* you got from the quote you created. Please read more about creating *requestID* here.
- After you receive *requestID* you can initiate a widget iOS with it. Please read iOS installation guide here.

>
> ### Important Tip!
>
> Tip: when building your UI in order to get the list of available fiat/crypto pairs and their respective min/max limits per transaction use the **Get currency pairs to buy crypto** and **Get currency pairs to sell crypto** endpoints.

Please note the API endpoints for environments:

- Sandbox for testing: - https://widget-api.sandbox.paybis.com/v1/*
- Production: - *https://widget-api.paybis.com/v1/*

3. Setup the Paybis Widget iOS SDK

Add our SDK to your project so you can run the widget's UI and handle its events. More information about the iOS SDK can be found in the installation guide below.

4: Run the widget

With the *requestId* received, you can start the widget using iOS SDK. For that, you need to request the start function. As a result, the widget's UI is displayed over your application and the user can interact with it directly.

Installation guide

Installation via CocoaPods

1. Make sure you have Cocoapods installed. To install Cocoapods, refer to Cocoapods Documentation.

2. If your project doesn't have a Podfile, you can create one and initialize Cocoapods using `pod init` method. It will create a Podfile where you will be able to add and install widget SDK following the instructions below.
   Remember to change `target` name to the one you are using.
3. If you already use Cocoapods in your project, add the Paybis Mobile Widget SDK pod reference to the proper `target` sections.

Ruby

```ruby
source 'https://gitlab.com/techcloud/shared/widget-sdk-ios-spec'
source 'https://github.com/CocoaPods/Specs.git'
source 'https://github.com/SumSubstance/Specs.git'

target 'WidgetExampleApp' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for WidgetExampleApp
  pod 'PaybisMobileWidgetSDK'
end
```

4. Open the Terminal app, go to the root directory of the project, and type `pod install`. The installation will be started, and you will be prompted to enter the username and password (private access token for Specs repo) that's provided by Paybis. Cocoapods will download all dependencies but will throw an error for PaybisMobileWidgetSDK installation. This is where you will need to move to the next step.
   Note: Size of `CocoaPods/Specs.git` repo is >500mb so it might take a while to fetch it.
5. Additionally, you'll need to run `pod install` command for the second time, but now you need to enter the username and password (private access token for Source code repo) that's provided by Paybis. As a result, Cocoapods will download `PaybisMobileWidgetSDK` library and all its dependencies and then create a `.xcworkspace` file, which from now on you should use to open your project (if you're not doing it already).

Installation via Swift Package Manager (SPM)

Swift Package Manager (SPM) is the most modern and streamlined way to integrate your SDK into iOS projects. It is built directly into Xcode, removing the need for third-party tools like CocoaPods or Carthage.

1. In Xcode, go to File → Add Package Dependencies...
2. Enter the package URL: https://gitlab.com/techcloud/shared/widget-sdk-ios.git
3. You'll be prompted with 2 login windows:

a. For the 2nd (focused) window, enter your GitLab username and personal access token for the `WIDGET-SDK-IOS` project.

b. Once authenticated successfully, switch to the 1st window and enter the credentials for `WIDGET-SDK-IOS-SPEC`.

4. Press Add Package and wait for the dependency to finish resolving.

Project setup

After integrating Widget iOS SDK, you need to adjust your app's permissions and capabilities.

SumSub

The widget uses the SumSub library for the Know Your Customer process. For SumSub to work, you have to add app permissions for Camera Usage, Microphone Usage, and Photo Library Usage. You can find out how to do it via SumSub documentation.

Zendesk

The widget uses the Zendesk library for the Contact Support process. For Zendesk to work, you have to add permissions for Camera Usage and Photo Library Usage. You can find out how to do it via Zendesk documentation.

Apple Pay

The widget has optional Apple Pay support. If you skip the steps below widget will still work, but the option to pay with Apple Pay will not show up.

Apple Pay setup can be achieved by a four-step process:

Creating merchant identifier

In your Apple Developer account go to the Certificates, Identifiers & Profiles section, select Identifiers and press Add New. From the radio selection select Merchant IDs and press continue.

Add a description and identifier you will later use in your app and press continue.

Creating payment processing certificate

Once the identifier has been created, additionally you've to create a payment processing certificate linked to the corresponding merchant identifier. To do so follow the steps below:

- Contact Paybis support team to get a certificate signing request `.csr`
- Log in back to your development portal and under the identifier section select the previously created merchant identifier
- Under *Apple Pay Payment Processing Certificate* section press *Create Certificate*
- Select according answer about processing only in China and press continue

- Upload the `.csr` gathered from Paybis support team and press continue
- Download created certificate `.cer` and add it to the folder to be accessible https://yourwebsite.com/.well-known/apple-developer-merchantid-domain-association.

Enabling Apple pay in XCode's capabilities.

Once both the identifier and certificate have been created and sent to Paybis team, you can add Apple Pay capability to your project using the steps below:

- In Xcode project navigator select your project file
- Select according to project's target
- Select *Signing & Capabilities* tab
- In the toolbar press + button to add a new capability
- From the dropdown list select *Apple Pay*
- Lastly, check the previously created merchant identifier to be used within the app

Connecting merchant identifier to SDK

When all previous steps are completed don't forget to pass the merchant identifier to SDK when starting the widget, example below:

Swift

```
import PaybisMobileWidgetSDK
...
PaybisMobileWidget.startWidget(
  ...
        merchantId: MERCHANT_ID
)
```

The guide can also be found in Apple Pay documentation.

APM redirects

To support redirects back to the app after a successful APM flow, defining a custom scheme and passing it to widget is necessary.

Defining app scheme

To define a custom URL scheme for your app do the following:

- In Xcode project navigator select your project file
- Select according to project's target
- Select *Info* tab
- Under URL types add a new entry by pressing on +

- Give your scheme identifier and define the scheme itself

The guide can also be found in Custom Scheme documentation

Passing scheme to widget

Once you've defined your app's custom scheme pass it to the widget, and we will redirect back to it once the APM flow finishes.

Swift

```
import PaybisMobileWidgetSDK
...
PaybisMobileWidget.startWidget(
  ...
  customAppScheme: CUSTOM_SCHEME://,
)
```

Important note: Remember to add *://* after your scheme as it allows the system to understand that it is a custom scheme and open your app

Starting SDK

To launch the widget `PaybisMobileWidget.startWidget` function should be called. The widget's UI will be presented as soon as the start method is called. Example:

Swift

```
import PaybisMobileWidgetSDK
...
PaybisMobileWidget.startWidget(
        environment: ENVIRONMENT,
        requestId: REQUEST_ID,
  customAppScheme: CUSTOM_SCHEME://,
)
```

Additionally, if you wish to skip authentication you can pass optional parameter `oneTimeToken`. Example:

Swift

```
import PaybisMobileWidgetSDK
...
PaybisMobileWidget.startWidget(
        environment: ENVIRONMENT,
        requestId: REQUEST_ID,
  customAppScheme: CUSTOM_SCHEME://,
```

```
        oneTimeToken: OPTIONAL_ONE_TIME_TOKEN
)
```

To get the aforementioned `oneTimeToken` you must set `passwordless` parameter to `true` when creating your request.

Note: If `passwordless` is set to `true` you **must** pass `email` parameter as well.

SDK Methods

Name	Arguments	Details
startWidget	`environment` - one of the widget's available environments (Sandbox / QA / Production) [required] `requestId` - generated unique `requestId` [required] `customAppScheme` - scheme defined in your app's info tab [required] `oneTimeToken` - generated one-time token [optional] `merchantId` - merchant identifier used to support Apple Pay capability [optional]	used to open the widget UI
closeWidget	n/a	used to close the widget UI

SDK Method Usage Examples

The examples below are related to the above table. The following list is not meant to be a guide but simply demonstrates a possible usage of the SDK.

Starts the Paybis widget.

Swift

```swift
import PaybisMobileWidgetSDK
...
// Without one time token
PaybisMobileWidget.startWidget(
  environment: ENVIRONMENT,
  requestId: REQUEST_ID,
  customAppScheme: CUSTOM_SCHEME://
)
// With one time token
PaybisMobileWidget.startWidget(
  environment: ENVIRONMENT,
  requestId: REQUEST_ID,
  customAppScheme: CUSTOM_SCHEME://,
```

```
  oneTimeToken: ONE_TIME_TOKEN
)
// With merchant identifier
PaybisMobileWidget.startWidget(
  environment: ENVIRONMENT,
  requestId: REQUEST_ID,
  customAppScheme: CUSTOM_SCHEME://,
  oneTimeToken: ONE_TIME_TOKEN,
  merchantId: MERCHANT_ID
)
```

Closes the Paybis widget.

Swift

```
PaybisMobileWidget.closeWidget()
```

SDK Events

Name	Arguments	Details
onError	`error` error object describing what went wrong	Issued when an error occurs in the widget
onPaymentInitiated	-	Issued when a user has initiated the payment in the widget (relevant for Frictionless sell crypto flow)

SDK Error Handling Examples

Perform the following when an error occurs with the widget.

Swift

```
PaybisMobileWidget.instance.onError = { [weak self] error in
        print(error)
}
```

Perform the following when the payment-initiated event occurs in the widget.

Swift

```
PaybisMobileWidget.instance.onPaymentInitiated = { [weak self] in
  PaybisMobileWidget.closeWidget()
  // handle necessary payment operation here and reopen the widget with same
```

```
request
}
```

SDK Environment Handling

SDK comes with multiple environments, to switch them you change the parameter you pass to the `.startWidget` function mentioned above (replacing the `ENVIRONMENT` value with one of the values below.

Name	Parameter
Sandbox	`.sandbox`
Production	`.production`

Updated about 2 months ago

← Helpful Tips

Mobile: Android SDK →