# iOS Sandbox SDK

After installing the CoinBridge iOS sandbox SDK, you gain the ability to simulate the correct utilization of our SDK within your sandbox environment.

While the majority of our methods are controlled by CoinBridge SDK, it's important to note that two specific methods fall under Apple's control and won't function within the sandbox environment.

To ensure the seamless operation of these methods, we've developed a tool that facilitates the following processes:

- **Simulating Provisioning**: This allows you to add the card to Apple Wallet without utilizing the Apple Provisioning process.
- **Simulating the Full Payment Cycle**: You can conduct transactions using Apple Pay without the need for actual Apple Pay functionality.

## Simulate "Add Card to Apple Wallet"

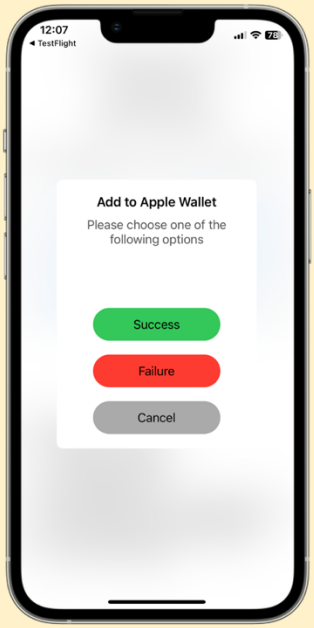
The purpose of this tool is to enable a suitable testing environment for our iOS SDK method, "Add Card to ApplePay." This method initiates the provisioning process of CB Virtual Cards to Apple Wallet.

In practice, when invoking this method, CoinBridge SDK will trigger an action that is controlled by Apple, and we can only determine the outcome of this action once the flow concludes.

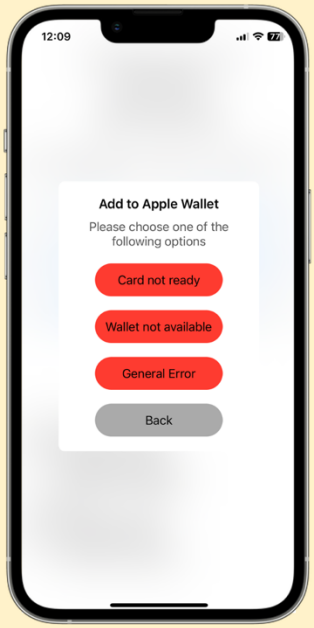
To test the SDK and its responses effectively, we developed of a tool that can simulate the provisioning flow and its results. This will allow you to ensure:

1. The proper functioning of our SDK.
2. The application's ability to handle various outcomes resulting from the provisioning flow.

To simulate various Provisioning statuses within the SDK Demo Application, Launch your Application and Progress through the application until you reach the point where a card is issued to the end-user, but it's not provisioned yet (*IsProvisioned=false*).




Provision Status




Provision "Failure" Reasons

Once you will reach this point you can click on "add To Wallet" button (within your iOS sandbox app), a pop-up will show the various Provisioning statuses possible.

- Clicking on **Success** will result
  - "onSuccess" callback (on SuccessHandler)
  - func onPaymentMethodStatusChange(status: CBPaymentStatus) will return with "isProvisioned = True" (on CBPaymentStatus)
  - "Pay with Apple Pay" is now available
- Clicking on **Failure** will open a pop-up to select the Failure reason:
  - **Card not ready**
    - By clicking on this failure reason SuccessHandler will receive a callback onError(CBError<paymentMethodNotActive>)
  - **Wallet not available**
    - By clicking on this failure reason SuccessHandler will receive a callback onError(CBError<walletLocked>)
  - **General Error**
  - By clicking on this failure reason SuccessHandler will receive a callback onError(CBError<generalError>)
- Clicking on **Cancel** will result
  - Cancelling the action, and not the provisioning flow (meaning no error and no change)

## Simulate “Pay with Apple Pay”

The purpose of this tool is to enable a suitable testing environment for our iOS SDK method, "Pay with ApplePay."

In practice, when invoking this method, CoinBridge SDK triggers an action that is controlled by Apple, and we can only determine the outcome of this action once the flow concludes.

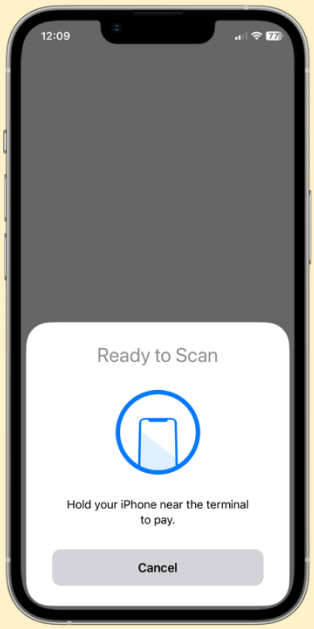
To test the SDK and its responses effectively, we developed of a tool that can simulate the payment flow and its results.

This will allow us to ensure:

1. The proper functioning of our SDK.
2. The application's ability to handle various outcomes resulting from the payment flow.
3. Simulate a full payment flow (and its influence on the loyalty points/balance).

After a successful provisioning on the sandbox environment, there isn’t an actual card in the Apple Wallet – and you will not be able to simulate a payment Tap using Apple Wallet.

So instead of opening the Wallet to pay, once you click on “Prepare to pay” and CoinBridge SDK receives onReadyToPay – our Sandbox SDK will open In-app tag reading, which will allow you to tap the device to a virtual terminal and send information from our SDK to the virtual Terminal.




Device is ready for Tap using NFC