# Push provisioning

To start Apple Wallet push provisioning, use `PKAddPaymentPassViewController` with `PKAddPaymentPassRequestConfiguration`.

The configuration should use the `.ECC_V2` encryption scheme.

```
import PassKit

let configuration = PKAddPaymentPassRequestConfiguration(encryptionScheme: .ECC_V2)!

configuration.cardholderName = "John Doe"
configuration.primaryAccountSuffix = "1234" // Last 4 digits of the card number

// Required when provisioning to Apple Watch.
// Use deviceAccountIdentifier from the phone PassItem.
configuration.primaryAccountIdentifier = phonePassItem.deviceAccountIdentifier

let viewController = PKAddPaymentPassViewController(
    requestConfiguration: configuration,
    delegate: delegate
)
```

If the card is being added to Apple Watch, `primaryAccountIdentifier` should be set using `deviceAccountIdentifier` from the phone `PassItem`.

## Delegate Implementation

The application must implement `PKAddPaymentPassViewControllerDelegate`.

In `generateRequestWithCertificateChain`, call the TMP API endpoint:

```
POST /issuer/push-provisioning/signed-cards
```

For Apple Pay, the request requires `certificate`, `nonce`, and `nonceSignature`. The TMP response returns `activationData`, `ephemeralPublicKey`, and `encryptedData`, which are required to create `PKAddPaymentPassRequest`.

**Swift Example**

```
import PassKit

final class AddPaymentPassDelegate: NSObject, PKAddPaymentPassViewControllerDelegate {

    func addPaymentPassViewController(
        _ controller: PKAddPaymentPassViewController,
        generateRequestWithCertificateChain certificates: [Data],
        nonce: Data,
        nonceSignature: Data,
        completionHandler handler: @escaping (PKAddPaymentPassRequest) -> Void
    ) {
        Task {
            do {
                let signedCard = try await signCard(
                    certificates: certificates,
                    nonce: nonce,
                    nonceSignature: nonceSignature
                )

                let request = PKAddPaymentPassRequest()
                request.activationData = Data(base64Encoded: signedCard.activationData)
                request.ephemeralPublicKey = Data(base64Encoded:
signedCard.ephemeralPublicKey)
                request.encryptedPassData = Data(base64Encoded: signedCard.encryptedData)

                handler(request)
            } catch {
                controller.dismiss(animated: true)
            }
        }
    }

    func addPaymentPassViewController(
        _ controller: PKAddPaymentPassViewController,
        didFinishAdding pass: PKPaymentPass?,
        error: Error?
    ) {
        controller.dismiss(animated: true)

        if let error {
```

```
            // Handle provisioning error
            print("Apple Wallet provisioning failed: \\(error)")
            return
        }

        // Provisioning finished successfully
    }
}
```

## Notes

- The code above is simplified and should be adapted to the existing TMP integration layer.
- Sensitive card data should not be handled directly in the mobile application unless explicitly required by the approved architecture.

---

Revision #1
Created 19 May 2026 07:34:35 by Bartłomiej Jończy
Updated 19 May 2026 07:35:02 by Bartłomiej Jończy