Article

# Accepting loyalty passes from Wallet

Set up the necessary components so your app can begin using Tap to Pay on iPhone to read and issue loyalty passes.

## Overview

Use ProximityReader to support Tap to Pay on iPhone so your app can read NFC passes in a customer's Wallet as part of a merchant's loyalty program. After configuration, ProximityReader can begin a `PaymentCardReaderSession` so you can:

- Read passes independently of payment, at the same time as payment, or instead of payment if the pass is available.
- Prompt a customer to join a merchant's loyalty program after a successful transaction, with a push notification.

> **Important**
>
> To read loyalty passes from Wallet, the passes must be NFC-enabled. For more information, see Getting Started with Apple Wallet.

For information about how to configure the card reader, see Adding support for Tap to Pay on iPhone to your app.

## Configure which loyalty passes to read

To read loyalty passes in Wallet with Tap to Pay on iPhone, you must have an existing Pass Type Identifier corresponding to a pass. To learn more, see Building a Pass.

You can also optionally specify a URL based on the web service you use to update passes. If no pass is found in the customer's Wallet that corresponds to the specified Pass Type Identifier, the provided URL will be pushed to the customer's device and can be used to trigger a push notification to add the loyalty pass in their Wallet. To learn more, see Adding a Web Service to Update Passes.

If you support multiple passes, you can create a `VASRequest` with multiple `VASRequest.Merchant` structures, each corresponding to a single Pass Type Identifier.

```
let loyaltyPass1 = VASRequest.Merchant(id: "pass.com.example-company.loyaltypass1",
                                       url: URL(string: "https://example-company.com
                                       localizedName: "Example Company Loyalty Pass

let loyaltyPass2 = VASRequest.Merchant(id: "pass.com.example-company.loyaltypass2",
                                       url: URL(string: "https://example-company.com
                                       localizedName: "Example Company Loyalty Pass

let request = VASRequest(vasMerchants: [loyaltyPass1, loyaltyPass2],
                         localizedVASType: "Example Company Loyalty Program")
```

## Read loyalty passes

Use `readVAS(_:)` to bring up a sheet that prompts the merchant's customers to hold their device near the reader. After a successful read of a loyalty pass, the sheet closes and the session returns a `VASReadResult`. If a read isn't successful, the sheet closes and the session returns a `PaymentCardReaderSession.ReadError`, which may require you or your user to take an action; make sure your app can handle all the different types of errors a read session can return.

After you get a `VASReadResult` you must send the pass content, `customerVASData`, to your loyalty-program provider. It's up to your app to process the returned data and to show the final result to the merchant.

> **Note**
>
> To receive specific details about the encryption and decryption of the NFC payload, you need an NFC Certificate for your pass. For more information, see NFC Certificate Request.

```
public class WrapperClass {

    var reader: PaymentCardReader?
    var session: PaymentCardReaderSession?
```

```
    public func readLoyaltyPass(for passTypeIdentifier: String,
                                merchantLoyaltyProgramName: String,
                                merchantLoyaltyPassName: String,
                                provisioningURL: URL?) async throws {
        let merchant = VASRequest.Merchant(id: passTypeIdentifier,
                                           url: provisioningURL,
                                           localizedName: merchantLoyaltyPassName)
        let request = VASRequest(vasMerchants: [merchant],
                                 localizedVASType: merchantLoyaltyProgramName)
        guard let reader = self.reader else {
            return
        }
        let events = reader.events
        do {
            Task {
                for await event in events {
                    // Handle events that happen while the sheet is up.
                }
            }
            let result = try await session?.readVAS(request)
            // Send result.entries[0].customerVASData to your loyalty program provid
        } catch {
            // Handle any errors that occur during read
            // (see PaymentCardReaderSession.ReadError).
        }
    }
}
```

## Combine the read of loyalty passes with payment

You can read both a payment card and multiple loyalty passes in one tap, the flow is the same as simply reading a payment card. Use `readPaymentCard(_:vasRequest:stopOnVASResult:)` to bring up the sheet prompting the merchant's customers to hold their device near the reader. After a successful read, the sheet closes and the session returns a tuple containing both `PaymentCardReadResult` and `VASReadResult` as a result. If a read isn't successful, the sheet closes and the session returns a `PaymentCardReaderSession.ReadError`, which may require you or your user to take an action; make sure that your app can handle all the different types of errors that a read session can return.

After you get a `VASReadResult` you must send the pass content, `customerVASData`, to your loyalty-program provider. It's up to your app to process the returned data and to show the final result to the merchant.

> **Note**
>
> For a list of all possible combinations, see readPaymentCard(_:vasRequest:stopOnVASResult:).

```
public class WrapperClass {

    var reader: PaymentCardReader?
    var session: PaymentCardReaderSession?

    public func readPaymentAndLoyaltyPass(for amount: Decimal,
                                          passTypeIdentifier: String,
                                          merchantLoyaltyProgramName: String,
                                          merchantLoyaltyPassName: String,
                                          provisioningURL: URL?) async throws {
        let merchant = VASRequest.Merchant(id: passTypeIdentifier,
                                           url: provisioningURL,
                                           localizedName: merchantLoyaltyPassName)
        let vasRequest = VASRequest(vasMerchants: [merchant],
                                    localizedVASType: merchantLoyaltyProgramName)

        let paymentRequest = PaymentCardTransactionRequest(amount: amount,
                                                           currencyCode: "USD",
                                                           for: .purchase)
        guard let reader = self.reader else {
            return
        }
        let events = reader.events
        do {
            Task {
                for await event in events {
                    // Handle events that happen while the sheet is up.
                }
            }
            let result = try await session?.readPaymentCard(paymentRequest,
                                                            vasRequest: vasRequest,
                                                            stopOnVASResult: false)
            // Send result.0?.paymentCardData to your payment service provider.
            // Send result.1?.entries[0].customerVASData to your loyalty program pro
        } catch {
            // Handle any errors that occur during read
            // (see PaymentCardReaderSession.ReadError).
```

```
        }
    }
}
```

## Prompt a customer to join a merchant's loyalty program

Use `readPaymentCard(_:vasRequest:stopOnVASResult:)` to prompt a customer to join a merchant's loyalty program, after a successful transaction, with a push notification. To do so, you only need to specify a `url` when trying to read a loyalty pass. If no pass corresponding to the specified Pass Type Identifier is found in the customer's Wallet, the reader session pushes the provided URL, through NFC, to the customer's device where the loyalty-program provider may trigger a push notification to add the loyalty pass in their Wallet.

# See Also

## Loyalty card requests

`class VASRequest`

A request to read a contactless loyalty card and retrieve loyalty program identifiers for the person.

`struct VASReadResult`

The result of a request to read loyalty card information.