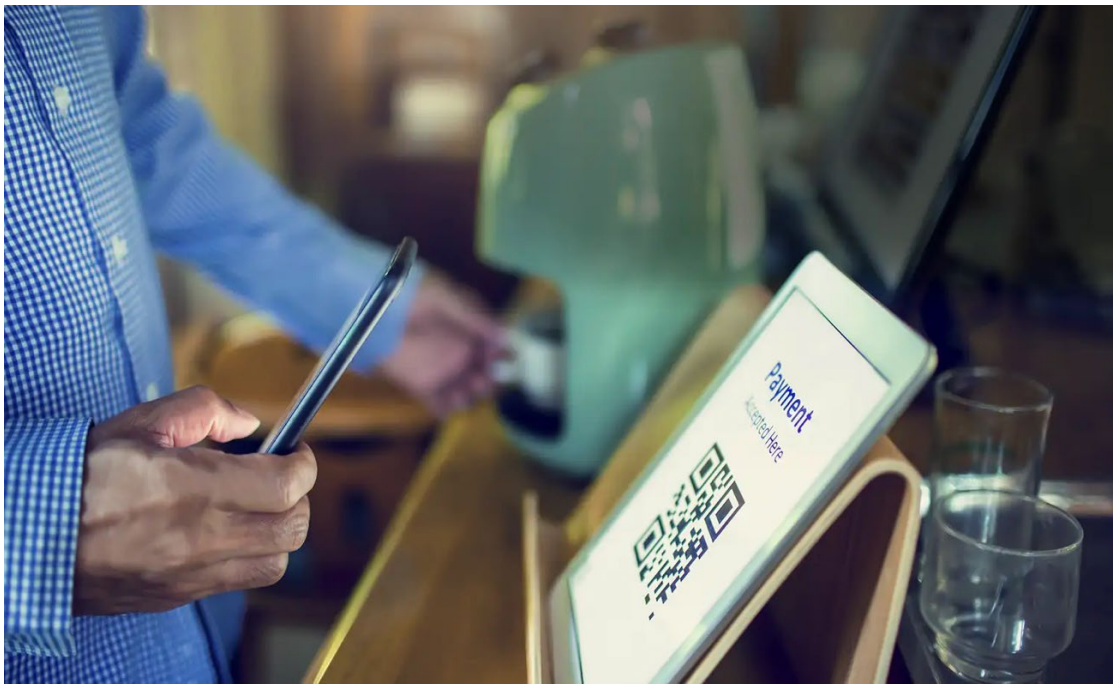
MAGENSA™
A MAGTEK COMPANY

Mobile App Payments

Google™ Pay and Apple Pay®, In-app and In-Web

Magensa Services Developer Program Manual



November 10, 2023
PN D998200516 v.100

MagTek® Inc., 1710 Apollo Court, Seal Beach, CA 90740 | p 562-546-6400 | f 562-546-6301 | www.magtek.com

Prerequisite:

The merchant must be successfully onboarded to MPPGv4/v3 and access to the "ProcessInAppApplePay" operation.

The merchant should possess a CSR (Certificate Signing Request) obtained during the onboarding process. MPPGv4/v3's "ProcessInAppApplePay" operation exclusively supports the "SALE" transaction type for processing InApp ApplePay tokens.

The process of requesting MPPG to handle in-app ApplePay tokens is simple. Please refer to the sample request provided later in this document. However, to make this call, several steps need to be taken as part of building the iOS app and this document will provide the corresponding instructions.

Purpose of the document:

The purpose of this document is to guide merchants and developers in integrating in-app Google Pay and in-app and In-web Apple Pay functionality through Magensa Payment Protection Gateway (MPPG v. 4,3) into an iOS app.

Table of Contents

1 Google Pay In-App Payments

1.1 Introduction

Magensa Web Services provide a wide variety of payment options, including InApp payments, subscriptions, rewards, and loyalty programs.

Google Pay (stylized as **G Pay**; formerly **Android Pay**) is a digital wallet platform and online payment system developed by Google to power InApp, online, and in-person contactless purchases on mobile devices, enabling users to make payments with Android phones, tablets, or watches. In addition to this, the service also supports passes such as coupons, boarding passes, campus ID cards, car keys, event tickets, movie tickets, public transportation tickets, store cards, and loyalty cards.

1.2 How It Works

When a user taps the Google Pay payment button, they see a payment sheet that displays the payment methods saved to their Google Account, as well as optional fields such as a shipping address field. Users can quickly select a payment method, add an optional shipping address, or add new information.

The payment flow is as follows:

1. The user taps the Google Pay payment button and sees a payment sheet with a list of supported payment methods.
2. The user selects a payment method and Google Pay securely returns a payment token for that method to your app.
3. Your app submits the payment token, along with details about the purchase, to its backend.
4. To execute the payment, the backend processes the purchase and sends the payment token to the payment service provider.

1.3 Set Up

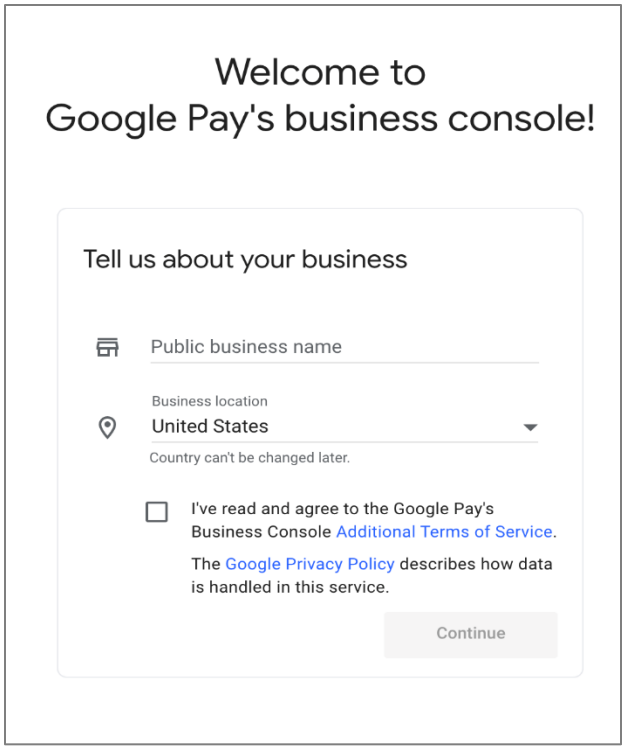
Follow the steps below to Set Up Google Pay.

1.4 Register a Business Profile

When merchants set up a payments profile, Google assigns it a unique numeric code called a Merchant ID. You cannot change this number. If you ever contact Google support, we'll ask you to give your Merchant ID.

To find your Merchant ID follow these steps:

1. Sign in to your payments profile.
2. At the top, click **Settings**.
3. Find 'Public merchant profile', then find your merchant ID.



1.5 Development Steps

- Step 1: Define your Google Pay API version.
- Step 2: Request a payment token for your payment provider.
- Step 3: Define supported payment card networks.
- Step 4: Describe your allowed payment methods.
- Step 5: Create a PaymentsClient instance.
- Step 6: Determine readiness to pay with the Google Pay API.
- Step 7: Create a PaymentDataRequest object.
- Step 8: Register event handler for user gesture.
- Step 9: Handle the response object.

Creating Payment Data Request

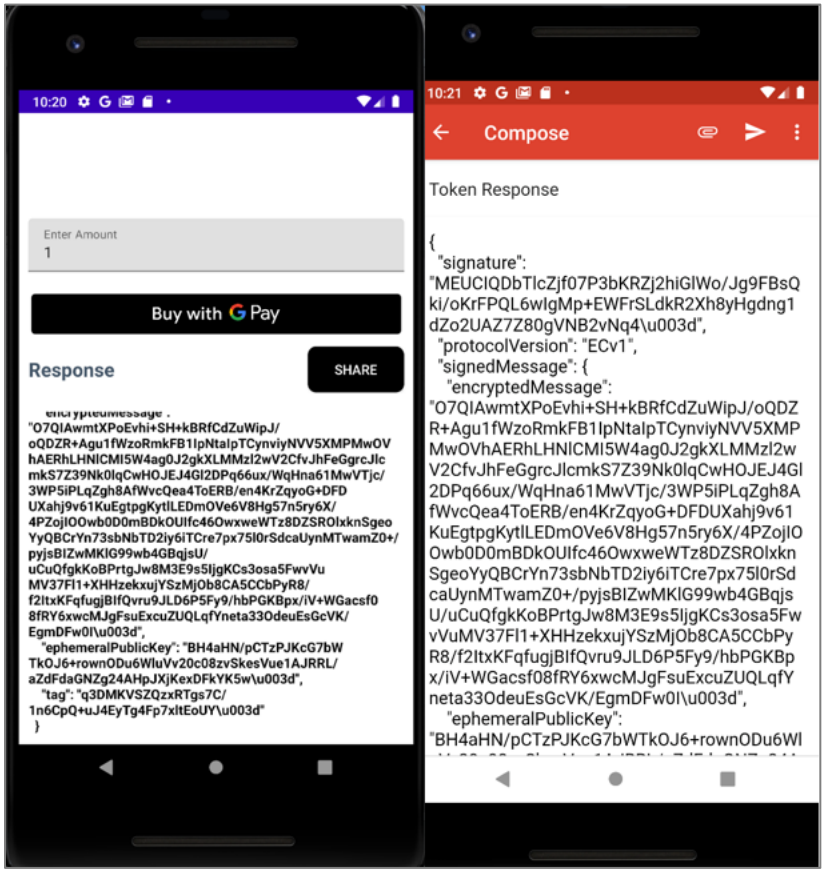
```
fun getPaymentDataRequest(price: String): JSONObject? {
    try {
        return JSONObject(baseRequest.toString()).apply {
            put("allowedPaymentMethods", JSONArray().put(cardPaymentMethod()))
            put("transactionInfo", getTransactionInfo(price))
            put("merchantInfo", merchantInfo)


            val shippingAddressParameters = JSONObject().apply {
                put("phoneNumberRequired", false)
                put("allowedCountryCodes", JSONArray(Constants.SHIPPING_SUPPORTED_COUNTRIES))
            }
            put("shippingAddressRequired", true)
            put("shippingAddressParameters", shippingAddressParameters)
        }
    } catch (e: JSONException) {
        return null
    }

}

private fun cardPaymentMethod(): JSONObject {
    val cardPaymentMethod = baseCardPaymentMethod()
    cardPaymentMethod.put("tokenizationSpecification", directTokenizationSpecification())

    return cardPaymentMethod
}
```

Handling Google payment token response:

```
private fun handlePaymentSuccess(paymentData: PaymentData) {
    val paymentInformation = paymentData.toJson() ?: return

    try {
        // Token will be null if PaymentDataRequest was not constructed using fromJson(String).
        val paymentMethodData = JSONObject(paymentInformation).getJSONObject("paymentMethodData")

        // If the gateway is set to "example", no payment information is returned - instead, the
        // token will only consist of "examplePaymentMethodToken".

        val tokenData = JSONObject(paymentMethodData
            .getJSONObject("tokenizationData")
            .getString("token"))
        val signedObject = JSONObject(tokenData.getString("signedMessage"))

}
```

1.6 Sharing Token Through Email

From Android app sharing gpay token through email.



2 Apple Pay

2.1 Apple Pay In-Web Payments

2.1.1 Introduction

In-web browser-based payments use TokenExchance Connect for manual entry and piggyback on Magensa iFrame technology. Apple Pay in-web requires site validation with TokenExchange Connect setup. Use this guide for Token Exchange Connect Domain Verification.

2.1.2 2.0 Verification Set Up

Simply follow the steps below.

2.1.2.1 Step One – Contact MagTek

Contact MagTek to obtain the Domain Verification File.

2.1.2.2 Step Two – Host the Domain Verification File

Host the domain verification file at the domain where you wish to host your web application:

- https://[DOMAIN_NAME]/.well-known/apple-developer-merchantid-domain-association

Ensure that the file is available at the path listed above.

If you wish to register multiple domains – ensure the domain verification file is hosted at the path specified above for each domain you wish to register.

2.1.2.3 Step Three – Register Domains with MagTek

Contact MagTek to register the domain(s) to your account.

2.2 Apple Pay In-App Payments

2.2.1 Introduction

Magensa Web Services provide a wide variety of payment options, including InApp payments, subscriptions, rewards, and loyalty programs.

Familiarize Yourself with ApplePay APIs: Gain a solid understanding of the ApplePay APIs by following the step-by-step instructions in the tutorial video available at: ApplePay API Configuration Tutorial. This video will guide you through configuring your Apple Developer Account to enable utilization of the ApplePay APIs.

https://developer.apple.com/videos/play/tutorials/configuring-your-developer-account-for-apple-pay/

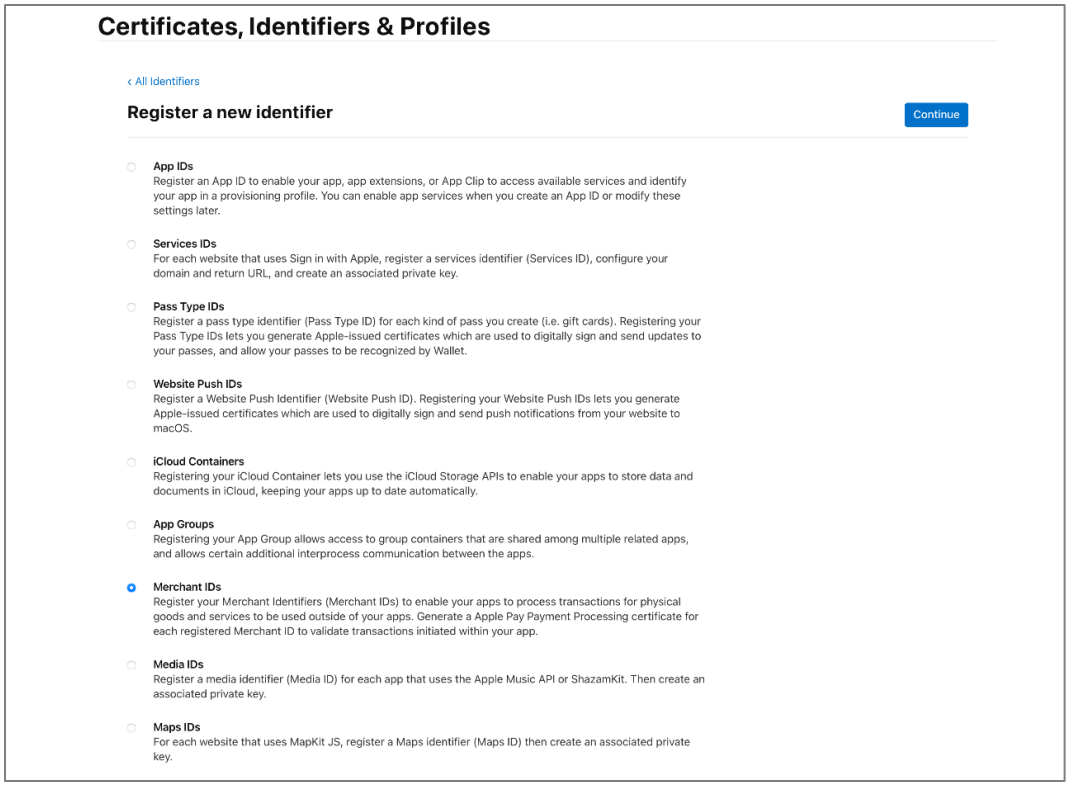
2.2.2 Register an ApplePay Merchant Identifier

To enable your app to use ApplePay, register an identifier with Apple that uniquely identifies your business as a merchant able to accept payments. This ID never expires and can be used in multiple websites and apps.

See Create a merchant identifier for the setup steps.

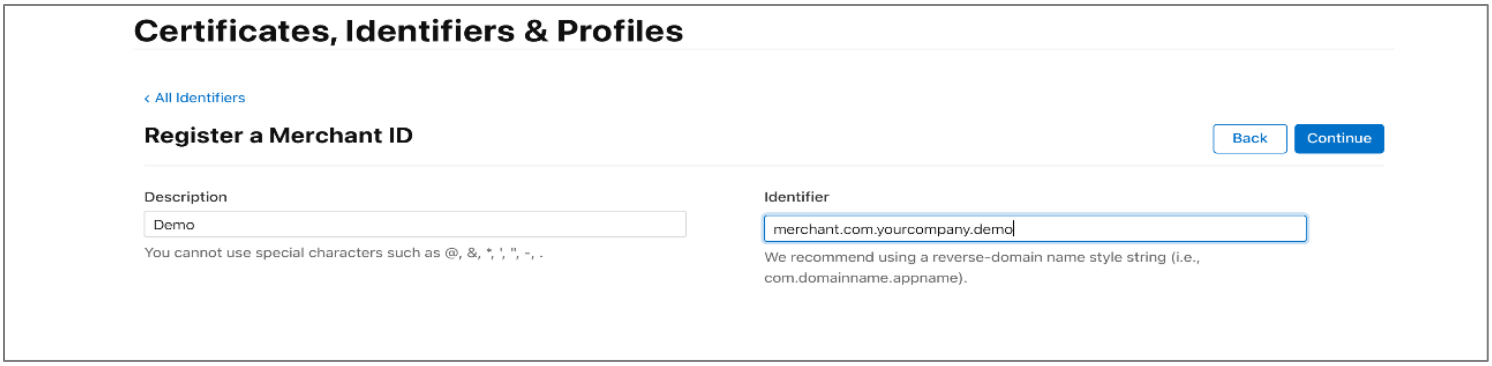
2.2.3 Step One - Register a new identifier

Select Merchant IDs



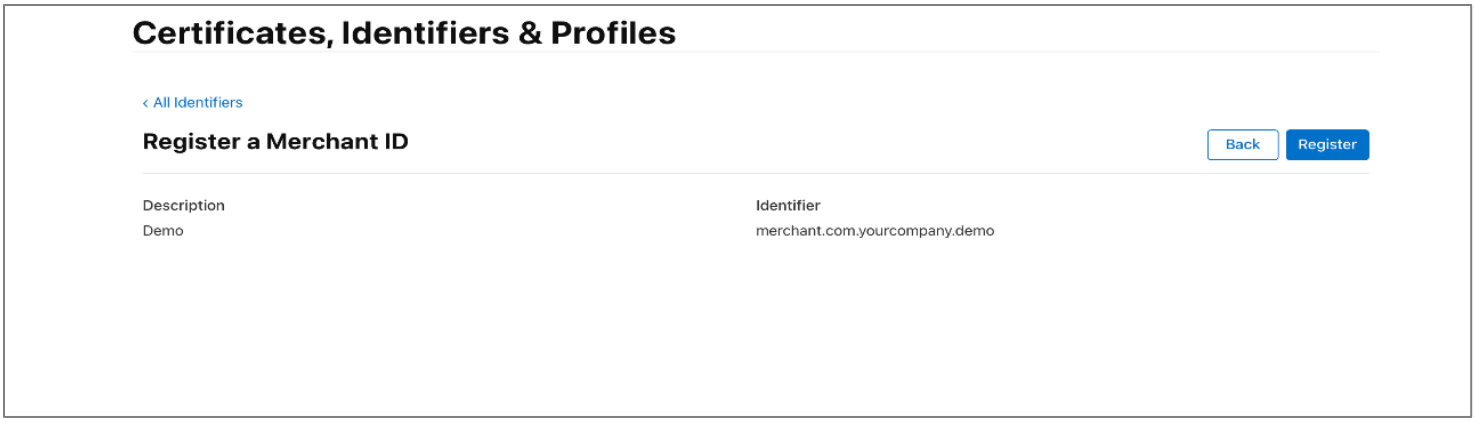
2.2.4 Step Two - Register a Merchant ID

Input description and identifier.



2.2.5 Step Three - Register

Ensure inputted information is accurate and click Register.

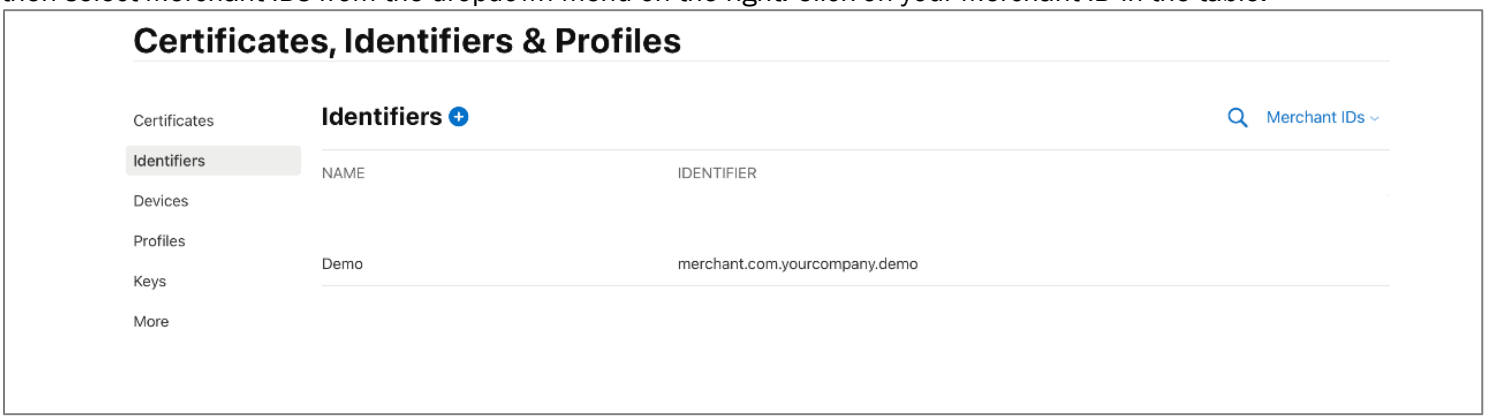


2.2.6 Create a new ApplePay Certificate

A new Certificate Signing Request (CSR) will be created for you at MagTek. Please upload this certificate to your Apple Developer account. This allows your app to encrypt payment data with MagTek's certificate.

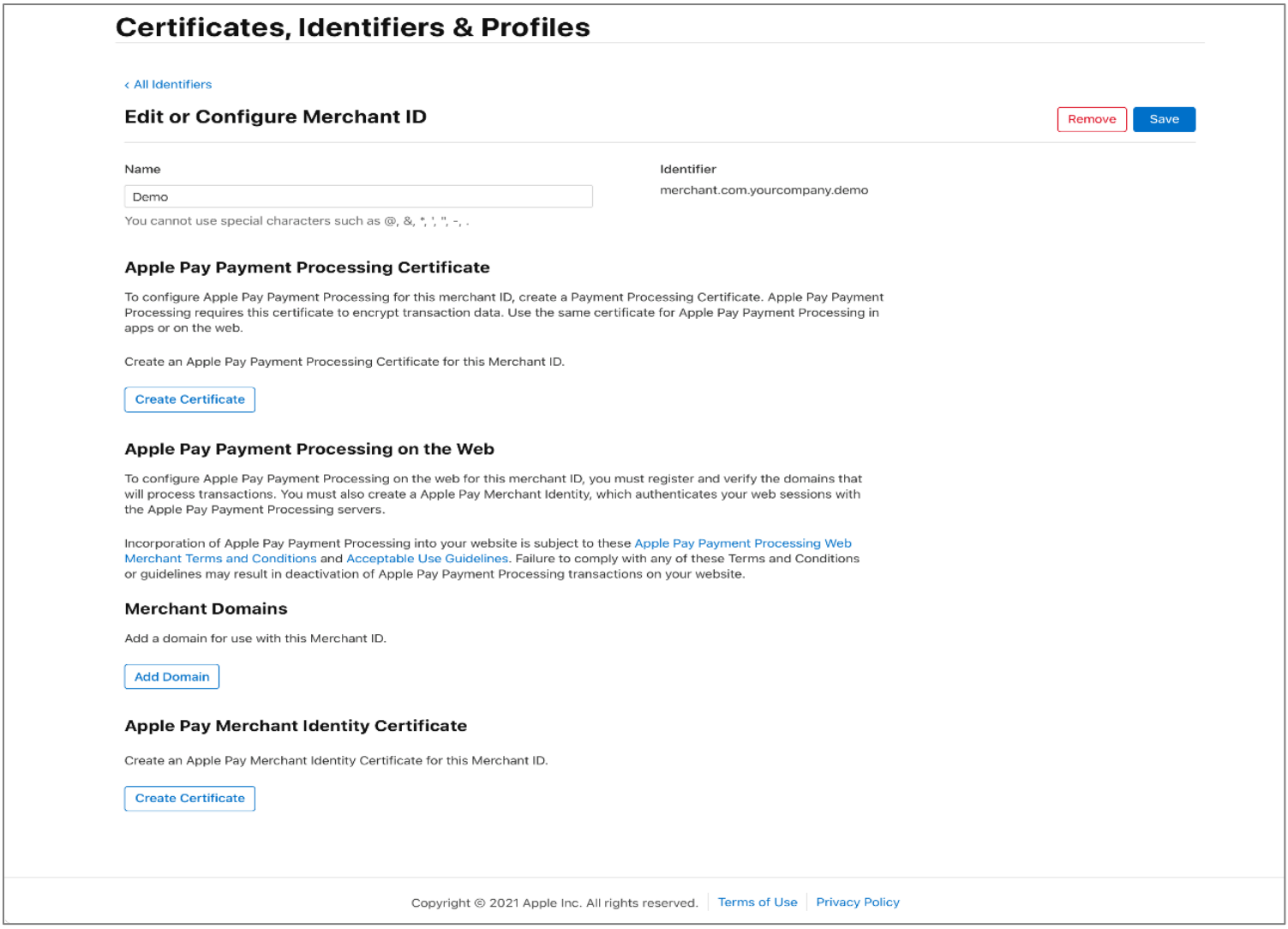
2.2.6.1 Step One – Login to the Apple Developer Website

Login at the Apple Developer website with your Apple ID and Click on Certificates, IDs & Profiles. Click Identifiers and then select Merchant IDs from the dropdown menu on the right. Click on your Merchant ID in the table.



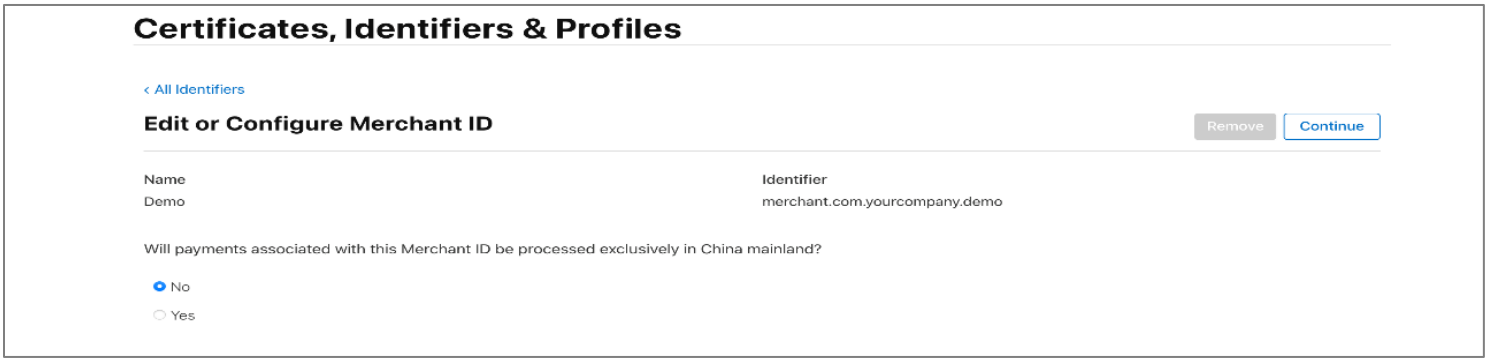
2.2.6.2 Step two – Create Certificate

Under the ApplePay Merchant Identity Certificate heading, click Create Certificate



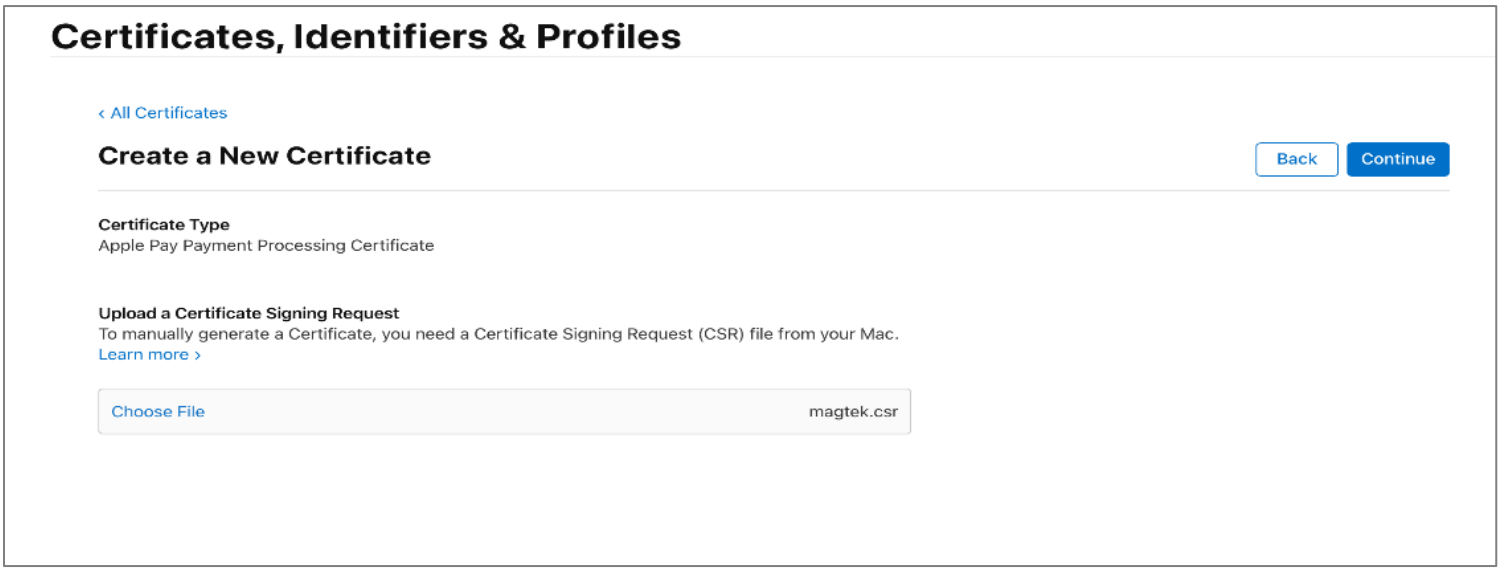
2.2.6.3 Step Three – Payments in China

Select the appropriate option with regards to payments in China and click Continue.



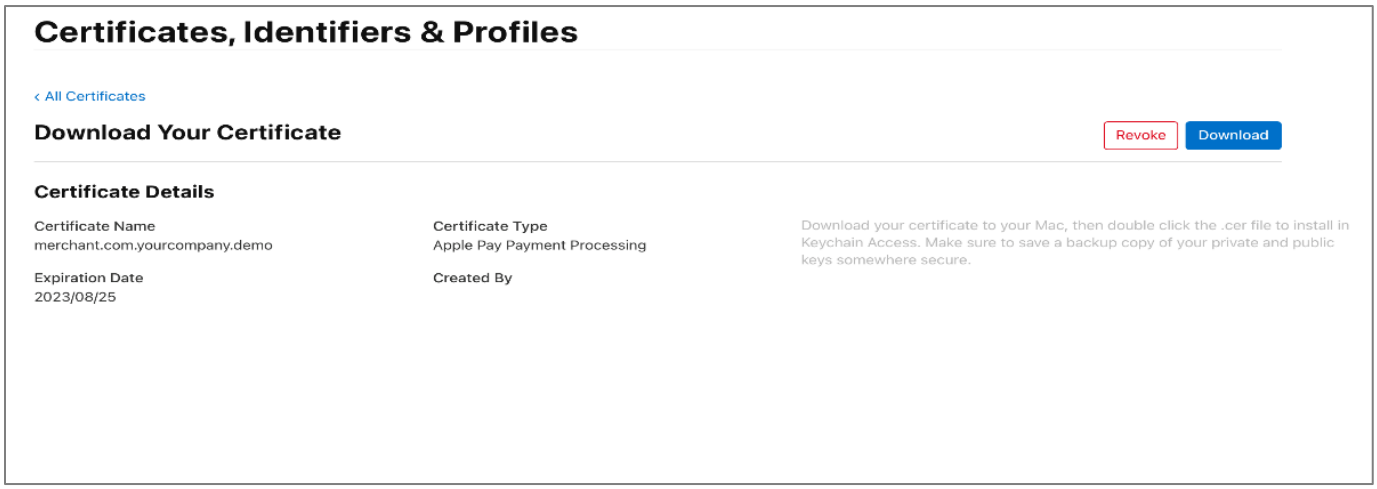
2.2.6.4 Step Four – CSR File Upload

Upload the CSR file and click Continue.



2.2.6.5 Step 5 – Download file

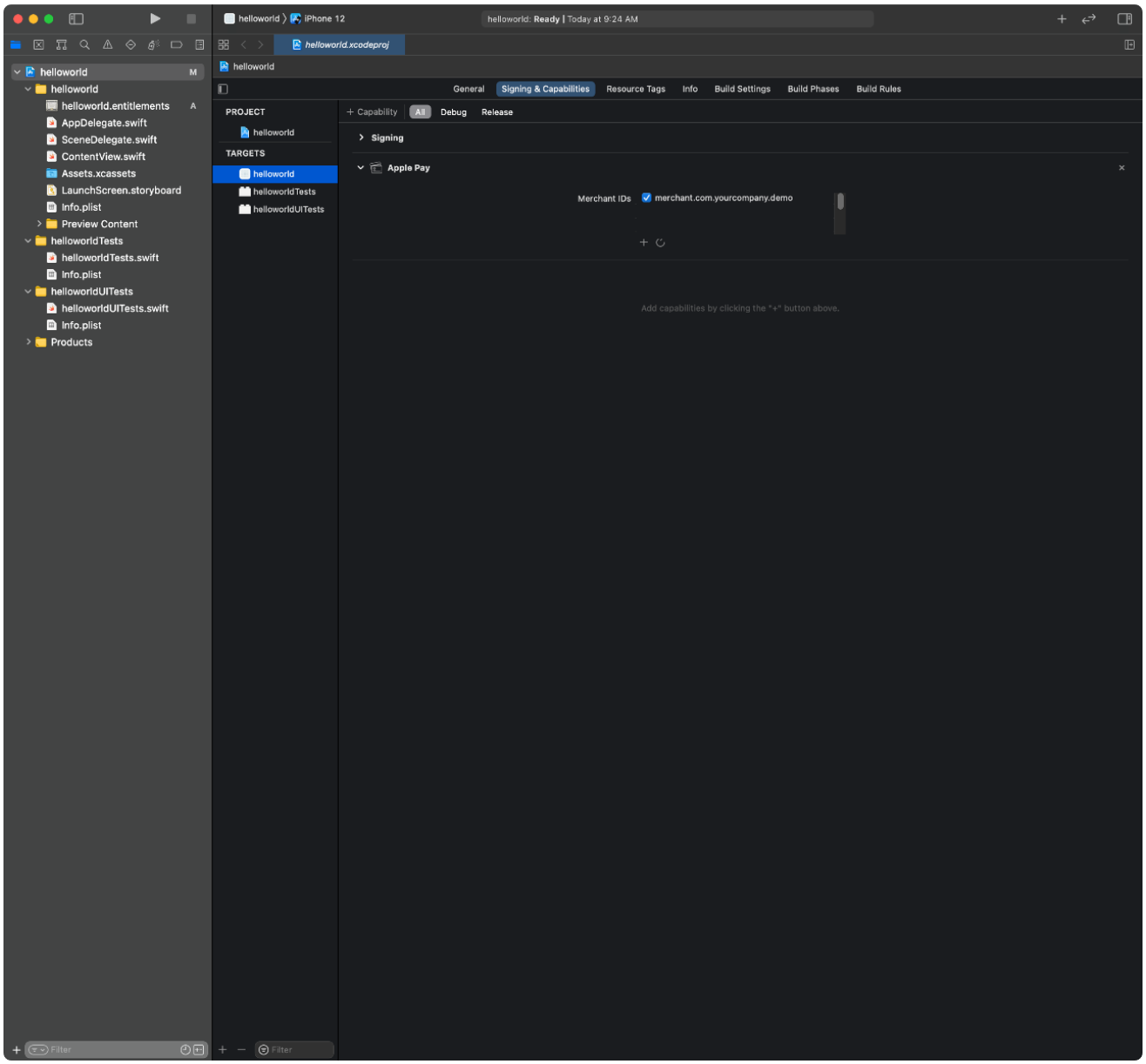
Click Download to download a file called apple_pay.cer



2.2.7 Integrate with Xcode

Add the ApplePay capability to your app. In Xcode, open your project settings, click the Signing & Capabilities tab, and add the ApplePay capability. You might be prompted to log in to your developer account at this point. Select the merchant ID you created earlier, and your app is ready to accept ApplePay.

2.2.8 Enable the ApplePay capability in Xcode



2.2.9 Send PKPaymentToken Data to MagTek

Your mobile application forwards the PKPaymentToken from ApplePay to MagTek servers.

```
// MARK: - PKPaymentAuthorizationViewControllerDelegate
 func paymentAuthorizationViewController(_ controller: PKPaymentAuthorizationViewController, didAuthorizePayment payment:
 PKPayment, handler completion: @escaping (PKPaymentAuthorizationResult) -> Void) {
     var paymentDataDictionary: [AnyHashable: Any]? = try? JSONSerialization.jsonObject(with: payment.token.paymentData,
     options: .mutableContainers) as! [AnyHashable : Any]
     MagTekAPI().send(paymentDataDictionary!)
```

2.2.10 “SALE” with InApp ApplePay token:

SOAP request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mpp="http://www.magensa.net/MPPGv4/"
xmlns:mpp1="http://schemas.datacontract.org/2004/07/MPPGv4WS.Core"
xmlns:sys="http://schemas.datacontract.org/2004/07/System.Collections.Generic">
  <soapenv:Header/>
  <soapenv:Body>
    <mpp:ProcessInAppApplePay>
      <mpp:ProcessInAppApplePayRequests>
        <mpp1:ProcessInAppApplePayRequest>
          <mpp1:ApplePayToken>{ApplePayToken}</mpp1:ApplePayToken>
          <mpp1:Authentication>
            <mpp1:CustomerCode>{CustomerCode}</mpp1:CustomerCode>
            <mpp1:Password>{Password}</mpp1:Password>
            <mpp1:Username>{Username}</mpp1:Username>
          </mpp1:Authentication>
          <mpp1:CustomerTransactionID>12345</mpp1:CustomerTransactionID>
          <mpp1:TransactionInput>
            <mpp1:Amount>1.00</mpp1:Amount>
            <mpp1:ProcessorName>{ProcessorName} </mpp1:ProcessorName>
            <mpp1:TransactionInputDetails>
            </mpp1:TransactionInputDetails>
            <mpp1:TransactionType>SALE</mpp1:TransactionType>
          </mpp1:TransactionInput>
        </mpp1:ProcessInAppApplePayRequest>
      </mpp:ProcessInAppApplePayRequests>
    </mpp:ProcessInAppApplePay>
  </soapenv:Body>
</soapenv:Envelope>
```

Appendix

- This model is designed to be implemented with a single merchant account; for other use cases, please contact MagTek.
- The tokens from ApplePay InApp transactions cannot be used with card on file transactions.