Special Remarks - Apple Pay

- 1 Introduction
  - 1.1 Availability
  - 1.2 Clearingtype / Clearingsubtype
  - 1.3 Requests
  - 1.4 Sequence Diagram
- 2 Prerequisites
  - 2.1 Onboarding
    - 2.1.1 Apple Developer Account
    - 2.1.2 Server Setup
    - 2.1.3 Create Merchant Identifier and Merchant Identity Certificate
    - 2.1.4 Create Payment Processing Certificate
  - 2.2 Apple Pay on Your Website
    - 2.2.1 How Apple Pay Works
    - 2.2.2 Initiating The Payment Session
      - 2.2.2.1 Apple Pay on the Web
      - 2.2.2.2 Apple Pay In-App
    - 2.2.3 Forwarding the Data to the Payone API
- 3 API Requests
  - 3.1 Overview of Special Parameters
    - 3.1.1 Apple Pay specific parameter Values
    - 3.1.2 Apple Pay Token Values
- 4 Apple Pay Specific Error Messages

Introduction

Apple Pay enables customers with iOS devices or Macs to pay using payment methods stored in their wallet-app. Merchants need to display an Apple Pay button to eligible customers, who then get presented a payment sheet for easy review of the order and payment. Merchants can configure the look and feel of both buttons and payment sheet, but should adhere to the Apple guidelines.

Availability

Countries	Payment Methods	Currencies
Check if Apple Pay is available in your target region	• Visa • Mastercard • girocard (expected soon)	All currencies that are also supported by the PAYONE platform

Please make sure you only make payment methods available for Apple Pay which are part of your contract with us.

Clearingtype / Clearingsubtype

clearingtype	wallettype
wlt	APL

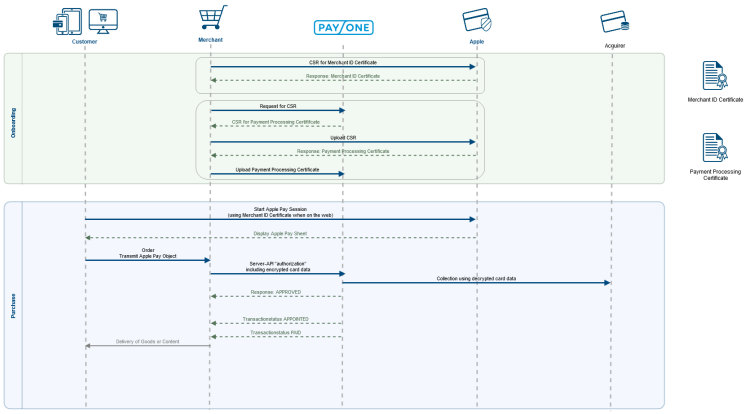
Requests

These Requests and Usecases are applicable:

Request	Comment
Preauthorization	
Capture	only after preauthorization
Authorization	
Debit	only with amount<0 to initiate a refund
Refund	

Sequence Diagram

Merchant View



Prerequisites

Onboarding

Merchants who want to offer Apple Pay must take these preparatory steps:

1

Apple Developer Account

First, please make sure your organization is enrolled in the Apple Developer Program.

2

Server Setup

Then, please make sure to follow the guidelines for Server Setup.

3

Create Merchant Identifier and Merchant Identity Certificate

Apple uses certificates for two steps in the payment process:

1. The Merchant Identity Certificate is used to authenticate your server when starting an Apple Pay session
2. The Payment Processing Certififcate is used to decrypt the payment object of a successful Apple Pay session

To create the Merchant ID and Merchant Identity Certificate, please follow the instructions on this Apple site.

You don't need a Mac to generate a CSR for a Merchant ID Certificate. Here's how to do it with openssl:

```
openssl genrsa -out private.key 2048
```

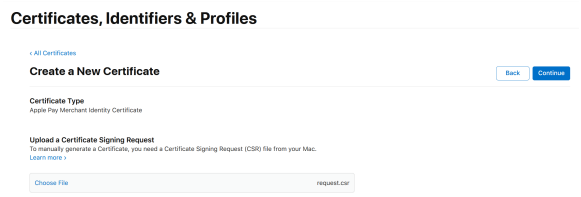
This generates a private.key file in your current folder. Keep this safe!

```
openssl req -new -sha256 -key private.key -nodes -out
request.csr
```

you will be asked some basic questions about your organization. After this, a request.csr file is generated. You can then use this file to generate your Merchant Identification Certificate at Apple.

If you want to convert the `merchant_id.cer` file into the more widely used `.pem` format, you can use this command:

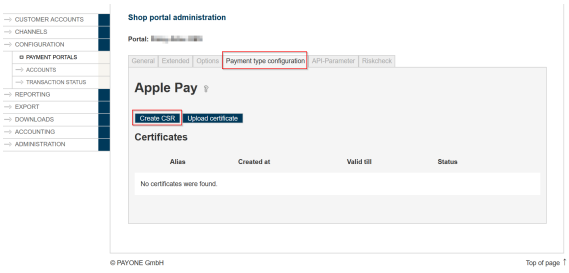
```
openssl x509 -inform der -in merchant_id.cer -outform pem
-out merchant_id.pem
```

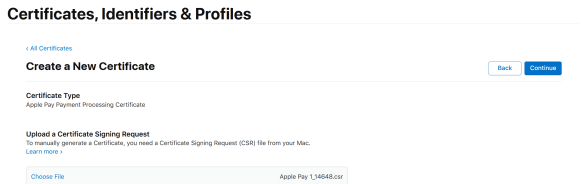


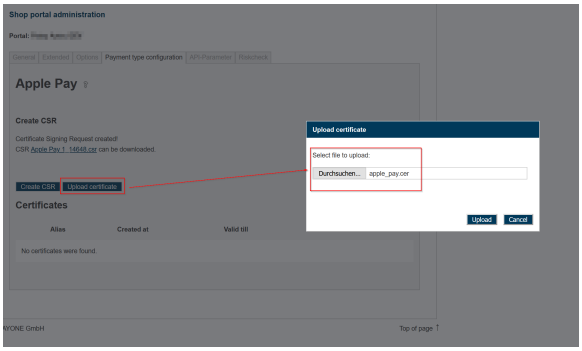
4

Create Payment Processing Certificate

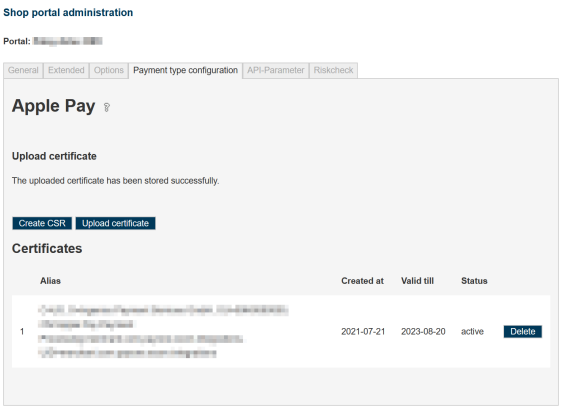
For the PAYONE Platform to be able to decrypt your Apple Pay objects, we need a Payment Processing Certificate. For this, you first need to create a "Certificate Signing Request" in your PAYONE Merchant Interface. This CSR can then be uploaded to Apple at https://developer.apple.com/account/resources/certificates/add, resulting in a Certificate in the .cer format. This file should then be uploaded to our Merchant Interface again.







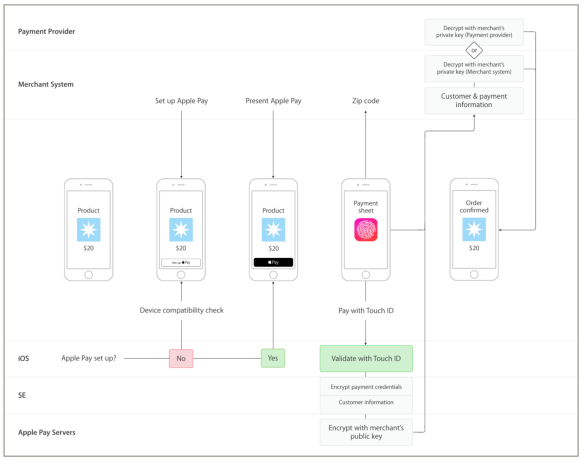
When done, you should have an active Apple Pay Certificate:



Apple Pay on Your Website

How Apple Pay Works

Like other payment buttons, Apple Pay aims to skip the usual checkout steps and presents a complete payment sheet to the customer.



source: Apple

Initiating The Payment Session

Apple Pay on the Web

Apple Pay on the Web uses JS-APIs built into Safari on Mac and mobile. For additional security, all Apple Pay sessions have to be initiated using the Merchant Identification Certificate. Additionally, your domains have to be whitelisted in the Apple Dev Portal.

For info on how to display the buttons and initiating the payment session, please refer to the Apple documentation: https://developer.apple.com/documentation/apple_pay_on_the_web/displaying_apple_pay_buttons and https://developer.apple.com/documentation/apple_pay_on_the_web/apple_pay_js_api/creating_an_apple_pay_session

> Head to https://applepaydemo.apple.com/ for a nice overview and some demo code.

Please make sure to correctly configure your payment request for your merchant account capabilities. For example, a basic request for a merchant who can use Mastercard, Visa and girocard in live mode could look like this:

```
{
  "countryCode": "DE",
  "currencyCode": "EUR",
  "merchantCapabilities": [
    "supports3DS" // mandatory
  ],
  "supportedNetworks": [
    "visa",
    "masterCard",
        "girocard"
  ],
  "total": {
    "label": "Demo (Card is not charged)",
    "type": "final",
    "amount": "1.99"
  }
}
```

Apple Pay In-App

In-App Payments use the Apple PassKit API. For info on how to accept Apple Pay payments in your app, refer to the Apple documentation: https://developer.apple.com/documentation/passkit/apple_pay/offering_apple_pay_in_your_app

As in Apple Pay on the web, you should configure your app to accept only the card schemes that your merchant account supports:

```
static let supportedNetworks: [PKPaymentNetwork] = [
    .masterCard,
    .visa,
        .girocard
]
```

This code snippet from the Apple documentation shows how you can send the resulting payment data to your backend.

```
func paymentAuthorizationController(_ controller:
PKPaymentAuthorizationController, didAuthorizePayment payment: PKPayment,
handler completion: @escaping (PKPaymentAuthorizationResult) -> Void) {

    // Perform some very basic validation on the provided contact
information
    var errors = [Error]()
    var status = PKPaymentAuthorizationStatus.success
    if payment.shippingContact?.postalAddress?.isoCountryCode != "US" {
        let pickupError = PKPaymentRequest.
paymentShippingAddressUnserviceableError(withLocalizedDescription: "Sample
App only picks up in the United States")
        let countryError = PKPaymentRequest.
paymentShippingAddressInvalidError(withKey: CNPostalAddressCountryKey,
localizedDescription: "Invalid country")
        errors.append(pickupError)
        errors.append(countryError)
        status = .failure
    } else {
        // Here you would send the payment token to your server or payment
provider to process
        // Once processed, return an appropriate status in the completion
handler (success, failure, etc)

                // PAYONE suggests sending the data to your backend first
and requesting the PAYONE Server API from there
         }

    self.paymentStatus = status
    completion(PKPaymentAuthorizationResult(status: status, errors:
errors))
}
```

Forwarding the Data to the Payone API

After the customer has completed the payment sheet and authenticated themselves by biometric means (TouchID, FaceID), you'll receive an Apple Pay Object like this:

Apple Pay Object


```
{
   "token":{
      "paymentData":{
         "version":"EC_v1",
         "data":"3+f4oOTwPa6f1UZ6tG...CE=",
         "signature":"MIAGCSqGSIb3DQ...AAAA==",
         "header":{
            "ephemeralPublicKey":"MFkwEK...Md==",
            "publicKeyHash":"l0CnXdMv...D1I=",
            "transactionId":"32b...4f3"
         }
      },
      "paymentMethod":{
         "displayName":"Visa 1234",
         "network":"Visa",
         "type":"debit"
      },
      "transactionIdentifier":"32b...4f3"
   },
   "billingContact":{
      "addressLines":[
         "1 Street",
         ""
      ],
      "administrativeArea":"",
      "country":"United Kingdom",
      "countryCode":"GB",
      "familyName":"Appleseed",
      "givenName":"John",
      "locality":"London",
      "postalCode":"AB12 3CD",
      "subAdministrativeArea":"",
      "subLocality":""
   },
   "shippingContact":{
      "addressLines":[
         "1 Street",
         ""
      ],
      "administrativeArea":"",
      "country":"United Kingdom",
      "countryCode":"GB",
      "familyName":"Appleseed",
      "givenName":"John",
      "locality":"London",
      "postalCode":"AB12 3CD",
      "subAdministrativeArea":"",
      "subLocality":"",
      "phoneNumber":"01234 567890",
      "emailAddress":"john.appleseed@apple.com"
   }
}
```


Many contents of this object can be mapped to existing Server API parameters.

Apple Pay Object

```
"billingContact":{
      "addressLines":[
         "1 Street",
         ""
      ],
      "administrativeArea":"",
      "country":"United Kingdom",
      "countryCode":"GB",
      "familyName":"Appleseed",
      "givenName":"John",
      "locality":"London",
      "postalCode":"AB12 3CD",
      "subAdministrativeArea":"",
      "subLocality":""
   },
```

PAYONE Server API

```
country=GB
lastname=Appleseed
firstname=John
street=1 Street
city=London
zip=AB12 3CD
```

However, the actual payment part of the object is encrypted and has to be sent to the PAYONE API in special parameters.

API Requests

Overview of Special Parameters

Apple Pay specific parameter Values		
API Parameter	**Required**	**Comments**
clearingtype	+	Fixed Value`wlt`
wallettype	+	Fixed Value`APL`

cardtype	+	can be obtained from the unencrypted part of the payment token
		
		Allowed Values`V`
		`M`
		
		`G`

Apple Pay Token Values		
add_paydata [paymentdata_token_version]	+	Sample`EC_v1`Format`String`

add_paydata[paymentdata_token_data]	+	Sample `rhHAQUrR118u[...]cwDw==` Format `String`
add_paydata [paymentdata_token_signature]	+	Sample `MIAGCSqGSIb3DQEHAqCAMIACAQE[...]AAAAAAA==` Format `String`
add_paydata [paymentdata_token_ephemeral_publickey]	+	Sample `MFkwEwYHKoZIzj0[...]Y2A==` Format `String`
add_paydata [paymentdata_token_publickey_hash]	+	Sample `ilecVF58bpB8qio[...]l6eirw2Y1v1KUCsdVgQ=` Format `String`
add_paydata [paymentdata_token_transaction_id]	+	Sample `be2e745845b31dfac7778c6e29[...]b658cbcca971c0e0` Format `String`

Example Request

```
add_paydata[paymentdata_token_data]=FpFyA6zSGkZC[...]
xi8xeXCNbpGBpvlNXfcang==
add_paydata[paymentdata_token_ephemeral_publickey]=MFkwEwYHKoZIzj0CA[...]
iXv34cYJ4lxZsjVgnsE0i6RX+mg==
add_paydata[paymentdata_token_publickey_hash]=tWOdQ0ARSRiQNsrS4[...]
7X6KBxLLAa8=
add_paydata[paymentdata_token_signature]=MIAGCSqGSIb3DQEHAq[...]
s9oHcqWMnolhsgAAAAAAAA
add_paydata[paymentdata_token_transaction_id]=12d7[...]d4eebc2e54109386
add_paydata[paymentdata_token_version]=EC_v1
aid=12345
amount=1000
api_version=3.11
cardtype=V
clearingtype=wlt
country=DE
currency=EUR
encoding=UTF-8
firstname=Demo
key=your key as md5
lastname=Dude
mid=12345
mode=test
portalid=123456
reference=your unique reference
request=preauthorization
wallettype=APL
```

Example Response

```
status=APPROVED
txid=123456789
userid=987654321
```

Apple Pay Specific Error Messages

Error	Description	Suggested Activity
2700	Request amount differs from apple pay token amount.	Make sure to use the same amount as in your Apple Pay payment sheet
2701	Request currency differs from apple pay token amount.	Make sure to use the same currency as in your Apple Pay payment sheet
2702	Failed to decrypt apple pay token	Check whether your Payment Processing Certificate is valid and uploaded to our merchant backend
2703	Certificate service declined request because of validation errors.	
2704	Required parameter in apple pay token is missing or empty	Check if all required parameters for the Apple Pay token are set