# Apple Pay - REST API V2

Apple Pay is a digital wallet for storing payment details, providing an easy and secure way to pay in iOS applications, websites running on Safari browser and contactless POS terminals. In applications and on websites, users can quickly and securely provide their payment, shipping and contact information to check out with just one touch using Apple's Touch ID. Apple Pay's simplicity increases conversion rates and new user adoption that come with it.

's Apple Pay solution supports following scenarios:

- Web payments
- In-application payments

## Web payments

Apple Pay on the Web enables purchases while using Safari web browser. For shopping on iPhone or iPad, after selecting checkout with Apple Pay, a payment sheet appears, prompting the customer to confirm payment via Touch ID. For shopping on Mac, customers need to have an iOS device in close range, and they'll be prompted on that device to authorize the payment, which will then synchronize to the browser. The latest Mac Book Pro allows customers to authorize payment directly on the Mac Book.

Apple Pay requirements:

- iOS 10. Apple Pay Web payments are supported on all iOS devices with a Secure Element.
- macOS 10.12. Apple Pay Web payments are supported in Safari.
  - The user must have an iPhone or Apple Watch that can authorize the payment.
  - On the latest Mac Book Pro payments can be authorized directly via Touch ID
- All pages that incorporate Apple Pay must be served over HTTPS.
- Your website must comply with the Apple Pay guidelines. For more information, see Apple Pay on the Web Acceptable Use Guidelines
- Server requirements

## In-app payments

Apple Pay can be used for payments within the iOS applications. The main advantage of Apple Pay is that it is frictionless process with no need to re-type credit card data, shipping or billing address, which can be cumbersome on the smaller iPhone screens. Customer just choose to checkout with Apple Pay and confirms the payment with Touch ID.

Merchants can also use 's SDK for facilitating the in-app payment integration, making it easier and faster.

Apple Pay requirements:

- Apple Pay in-app payments are supported on all iOS devices with a Secure Element.

## Get started

supports Apple Pay in all of the integration types: Hosted Payment Page, Hosted forms and Direct integration. Follow the guide below tailored to your integration.

## Hosted payment page or Hosted forms

The process here is straight forward and requires minimum integration efforts.

**Key advantages:**

- No need of Apple developer account
- No need to set up any payment processing or merchant identity certificates
- No need to integrated with Apple Pay APIs
- takes care of entire communication with Apple Pay through out the payment

### Implementation steps

#### Enable Apple Pay

- Contact to activate Apple Pay for your merchant ID.

#### Add to checkout

- Submit payment request via Create checkout session or Create payment (`paymentMethods.integrationType=HOSTED`) depending on your integration.
- Apple pay automatically appears as on option to pay once your customers are redirected to the payment page.

## Direct integration

For merchants requiring full control over Apple Pay UI/UX use the Direct integration. However, this comes with some additional costs and integration efforts.

### Implementation steps

#### Enable Apple Pay

- **Create Merchant Identifier with Apple Pay:**
  - Access your Apple developer account and follow the instructions to create a merchant identifier. The identifier you enter should be a reverse DNS style identifier prefixed with the word `merchant`. Make sure that no umlauts or special characters are present.
  - Provide the merchant identifier to .
- **Retrieve Certificate Signing Request (CSR):**
  - With your Apple merchant identifier creates the Apple Pay CSR and provides it you.
  - CSR will be needed to create payment processing certificate that is required to sign and encrypt the payment token by Apple during the payment processing.
- **Create a payment processing certificate:**
  - In your Apple developer account follow the instructions to create a payment processing certificate.
  - Skip the step to create a certificate signing request. Instead use the CSR provided by in the next step to create the payment processing certificate.
- **Enable Apple pay for your app on Xcode:**
  - Follow the instructions to Enable Apple pay capability for your app in your Xcode project.
- **Create Merchant Identity Certificate:** This is a transport layer security (TLS) certificate used to authenticate your merchant sessions with Apple Pay
  - Follow the instructions to register, verify your domain and create a merchant identity certificate.
- **Integrate Apple Pay in your application or web shop:**
  - Follow the detailed guidelines documented by Apple to integrate to Apple Pay
  - Make sure when creating PKPaymentRequest, request.merchantCapabilities = PKMerchantCapability3DS as supports only 3DS types.
  - In case you are processing Carte Bancaire via Apple Pay, the choice of the default brand (CB over Visa or Visa over CB) is linked to the order of the brands indicated in the supportedNetworks field
    - Follow the instructions: supportedNetworks | Apple Developer Documentation
    - If "cartesBancaires" is mentioned before Visa, then a co-branded CB/Visa card will be processed via CB unless the customer changes the brand to Visa.

#### Add to checkout

- **Presenting Apple Pay button:**
  - Within apps, PassKit provides the API's that your app will use to determine if it is running on a device with a Secure Element and if the device has been provisioned with payment cards that you support.
  - On websites, WebKit provides APIs that allow your website to check if the individual has an Apple Pay capable device and if it is set up.
  - If the device is Apple Pay enabled you should present the Buy with Apple Pay button using APIs supplied within PassKit within apps and Webkit within Safari.
- **Presenting the payment sheet:**
  - When the customer selects Apple Pay as the payment method, you create a payment request and communicate with PassKit in apps or Webkit on your website to present the payment sheet to the user. The payment sheet must immediately follow the user tapping the Apple Pay button, without any interim screens or pop-ups except to prompt for necessary product details such as size or quantity.
  - Your app specifies the contents of the payment sheet but it does not control the user's interaction with the sheet. You must decide if it makes sense to present shipping and billing information, shipping method and other line items to the user. You should only request the information necessary to process the transaction.
- **Processing payment:**
  - Once authorized by the customer with Face ID/Touch ID (and bank PIN code in China), your app receives a payment token from PassKit/Webkit.
  - The payment token encapsulates the information needed to complete a payment transaction, including the device-specific account number, the amount, and a unique, one-time-use cryptogram.
  - Submit the payment token to via Create payment (`paymentMethods.integrationType=DIRECT`) in `paymentMethods.applePay.token` parameter along with other payment information. The token value should be submitted as Base64 encoded format.

```
"paymentMethods": {
  "integrationType": "DIRECT",
  "type": "APPLEPAY",
  "applePay": {
    "merchantIdentifierOfPublicKey": "merchant.com.demo_store",
    "token": "<<Base64_Encoded_Token_Data>>"
  }
}
```

- The encrypted payment token will be decrypted by  and submitted to the acquirer as part of the payment process.

> Apple Pay transactions are essentially card transactions but tokenized. Hence all the post processing actions such as refunds, captures and reversals use the same workflow as card transactions.

## Testing Apple pay transaction

Testing transaction with Apple pay test account

1) Axepta BNP Paribas Helpdesk can configure Apple pay on the Axepta merchant test MID.

Axepta BNP Paribas Helpdesk needs the apple pay merchant identifier in order to configure Apple pay on the Axepta BNP Paribas merchant test MID.

2) The Merchant creates a Sandbox Tester Account and add test card Number.

Apple pay documentation : Sandbox Testing - Apple Pay - Apple Developer