



M14: Apple Pay Integration using REST API

Version	1.0
Date	November 2020

1 Version Control

Revision	Date	Description
1.0	November 2020	Initial release

2 Table of Contents

1	Version Control	2
2	Table of Contents.....	3
3	Introduction	4
4	Apple Pay integration	5
4.1	Apple Pay In-App integration	5
4.2	Apple Pay integration for iOS browser.....	6
4.3	Process Flow.....	6
4.4	Apple Pay follow-on transactions workflows.....	7
4.5	Recurring payments overview.....	7
5	Smartpay Fuse Apple Pay profile configuration.....	8
6	REST API Examples Using Apple Pay payload.....	11
6.1	Apple Pay Authorisation request	11
6.2	Apple Pay Authorisation reply	13
	Disclaimer	14

3 Introduction

This document describes how to add an Apple Pay payment option to your Website or mobile App using the Smartpay Fuse Gateway. The document is intended for development staff who will perform the implementation.

There are two ways that your customers can use Apple Pay to make payments:

- In-App transactions on a supported IOS device
- Web transactions in a supported version of Safari.

In both cases, Smartpay Fuse does not provide guidance for the frontend Apple Pay integration. You must use the Apple PassKit Framework - provided by Apple - to request the encrypted payment data from the device. This data can then be used in a Smartpay Fuse REST API request.

Please note that Payer Authentication (3DS) is not required for Apple Pay authorizations, since Apple Pay provides Strong Customer Authentication.

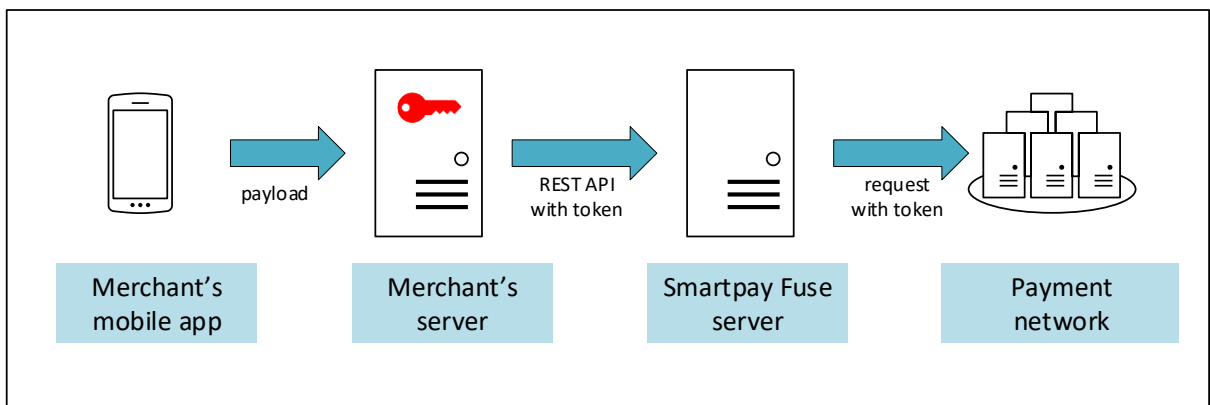
4 Apple Pay integration

4.1 Apple Pay In-App integration

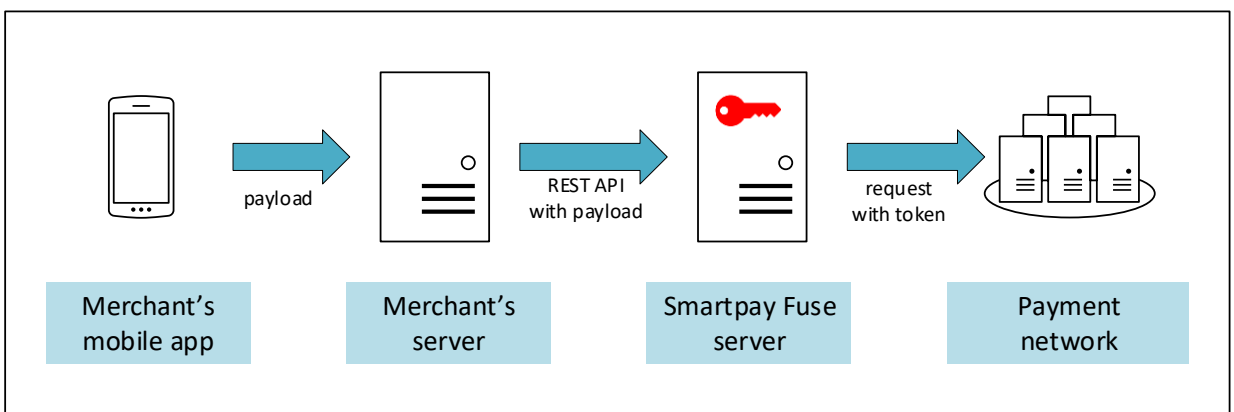
When the customer chooses to pay with Apple Pay, you use the Apple PassKit Framework to request the encrypted payment data from Apple in the form of a PKPaymentToken object, also known as a payload. This payload contains a Network payment token, purchase description and various other parameters. To make a payment this payload must be decrypted before sending on to the payment network for authorization.

There are two options for decrypting the payload:

- 1) Merchant decryption - you have a private key to decrypt the BLOB on your server and then populate Smartpay Fuse Authorisation API request fields with decoded parameters.



- 2) Smartpay Fuse decryption – your merchant server receives the encrypted payload from the mobile device or browser and sends it on to Smartpay Fuse for decryption and payment processing. In that case, Smartpay Fuse must have the private key to decrypt the payload.



In most cases you should use the second option – Smartpay Fuse decryption. This is the recommended option.

If you have multiple Smartpay Fuse transacting MID's that will accept Apple Pay authorizations, you will need to generate a CSR for each one and ensure that your app sends the authorization request to the Smartpay Fuse MID with the appropriate CSR. If the payload is sent into the incorrect Smartpay Fuse MID, decryption will be impossible and payments will fail.

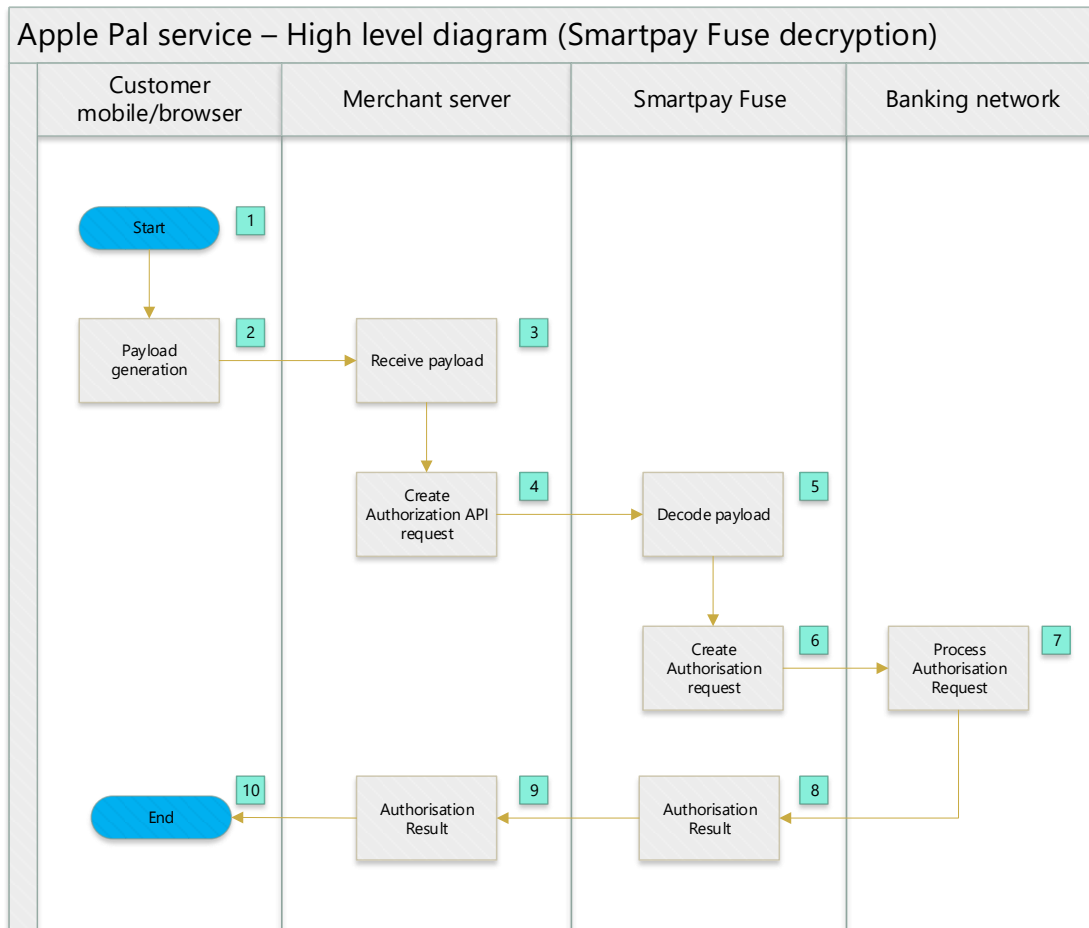
4.2 Apple Pay integration for iOS browser

The payment workflow for Apple Pay payments in the iOS Safari browser is very similar to payments in a mobile application. You use the Apple Pay JavaScript to request the cardholder's payment data from the device in the form of a PKPaymentToken structure, also known as a payload. The Secure Element of the Apple device creates the PKPaymentToken and sends it back to your application using the **onpaymentauthorized** call-back function.

The customer authentication for iPhone, iPad and MacBook Pro is provided via Touch ID / Face ID, or in the case of other Macs, Touch ID is provided with the customer's nearby iPhone, or Apple Watch that supports Apple Pay.

4.3 Process Flow

The following diagram and description illustrate the high level flow of an Apple Pay payment with Smartpay Fuse decryption option.



M14: Apple Pay Integration using REST API - 6

1	Customer places an order using Apple mobile device or browser and confirm payment via TouchID or FaceID.
2	Apple mobile device generates the payload and sends it to your server.
3	You receive payload from the mobile device.
4	You create a REST API request, using the payload as one of request parameters and send the Authorisation request on dedicated Smartpay Fuse MID.
5	Smartpay Fuse decodes the payload, using the private key.
6	Smartpay Fuse creates an Authorization request to the Banking network.
7	Issuing Bank grant or decline Authorisation and send response back to Smartpay Fuse.
8	Smartpay Fuse sends the response to your server.
9	Your server sends the response to the customer's Apple mobile device.
10	End of payment workflow.

4.4 Apple Pay follow-on transactions workflows

Apple Pay follow-on payment services are:

- Authorization Reversal
- Capture
- Void
- Follow-on Credit
- Stand-alone Credit

These services are the same as card payments transactions and do not require any additional fields. All transactions use the **requestID** parameter from the previous transaction (except Stand-alone Credit).

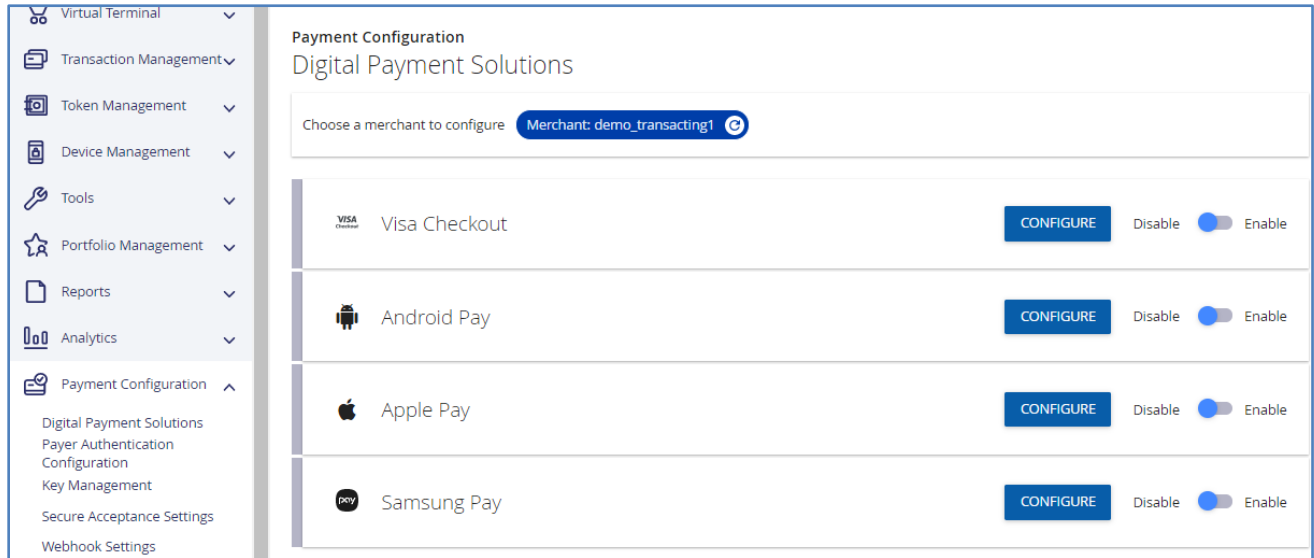
For descriptions of these transactions, please refer to the **M16: REST API Integration** document.

4.5 Recurring payments

To create a recurring subscription based on the Apple Pay payment information, create a TMS token with the first customer initiated authorization transaction. Then use this TMS token for the follow-on recurring payments, but add COF/MIT mandate fields for proper transaction definition.

5 Smartpay Fuse Apple Pay profile configuration

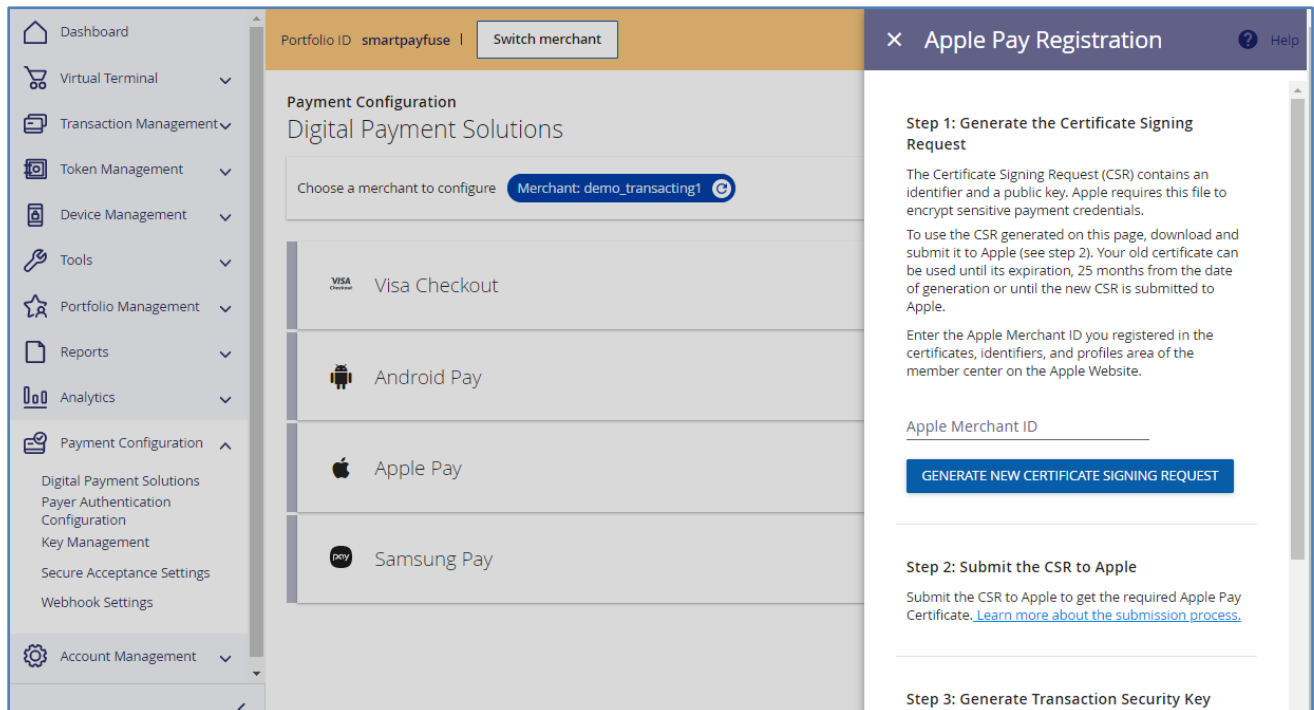
To enable Apple Pay on your Smartpay Fuse MID, login into Enterprise Business Centre (EBC) and select **Payment Configuration-> Digital Payment Solutions** from the side bar menu. Then click on the **CONFIGURE** button next to Apple Pay.



The screenshot displays the 'Payment Configuration' page for 'Digital Payment Solutions'. The left sidebar contains a navigation menu with the following items: Virtual Terminal, Transaction Management, Token Management, Device Management, Tools, Portfolio Management, Reports, Analytics, and Payment Configuration (expanded). Under 'Payment Configuration', there are sub-items: Digital Payment Solutions, Payer Authentication Configuration, Key Management, Secure Acceptance Settings, and Webhook Settings. The main content area shows a 'Choose a merchant to configure' dropdown set to 'Merchant: demo_transacting1'. Below this, there are four rows of payment solutions, each with a 'CONFIGURE' button and a toggle switch for 'Disable' and 'Enable':

- Visa Checkout: CONFIGURE button, Disable/Enable toggle (disabled).
- Android Pay: CONFIGURE button, Disable/Enable toggle (disabled).
- Apple Pay: CONFIGURE button, Disable/Enable toggle (enabled).
- Samsung Pay: CONFIGURE button, Disable/Enable toggle (disabled).

In the **Apple Merchant ID** input field provide your Apple Merchant ID you registered in Certificates, Identifiers & Profiles of the Member Center on the Apple website and then click **GENERATE NEW CERTIFICATE SIGNING REQUEST** button.



The screenshot shows the 'Apple Pay Registration' configuration page. The left sidebar is the same as in the previous screenshot. The main content area shows the 'Apple Pay Registration' form. The form includes a 'GENERATE NEW CERTIFICATE SIGNING REQUEST' button and instructions for generating and submitting a Certificate Signing Request (CSR). The form also includes an 'Apple Merchant ID' input field.

Step 1: Generate the Certificate Signing Request

The Certificate Signing Request (CSR) contains an identifier and a public key. Apple requires this file to encrypt sensitive payment credentials.

To use the CSR generated on this page, download and submit it to Apple (see step 2). Your old certificate can be used until its expiration, 25 months from the date of generation or until the new CSR is submitted to Apple.

Enter the Apple Merchant ID you registered in the certificates, identifiers, and profiles area of the member center on the Apple Website.

Apple Merchant ID _____

GENERATE NEW CERTIFICATE SIGNING REQUEST

Step 2: Submit the CSR to Apple

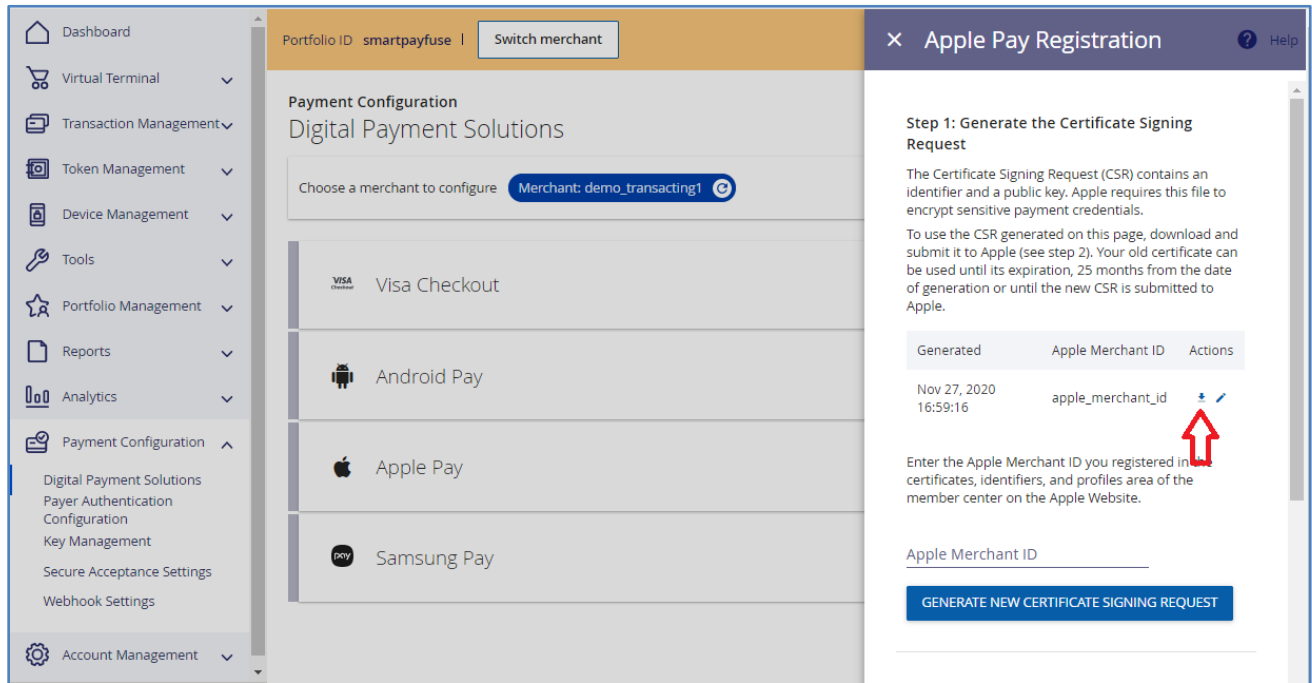
Submit the CSR to Apple to get the required Apple Pay Certificate. [Learn more about the submission process.](#)

Step 3: Generate Transaction Security Key

If you are going to use Smartpay Fuse payload decryption, you need to download the CSR and then submit the CSR to Apple. Apple will provide you with an Apple Pay Certificate for your Apple Merchant ID.

The generated Certificate Signing Request (CSR) contains an identifier and a public key, which will be used to encrypt sensitive payment information on the customer device, before sending this information to your server. The private key is stored in Smartpay Fuse and used for decryption. You will not have access to this private key.

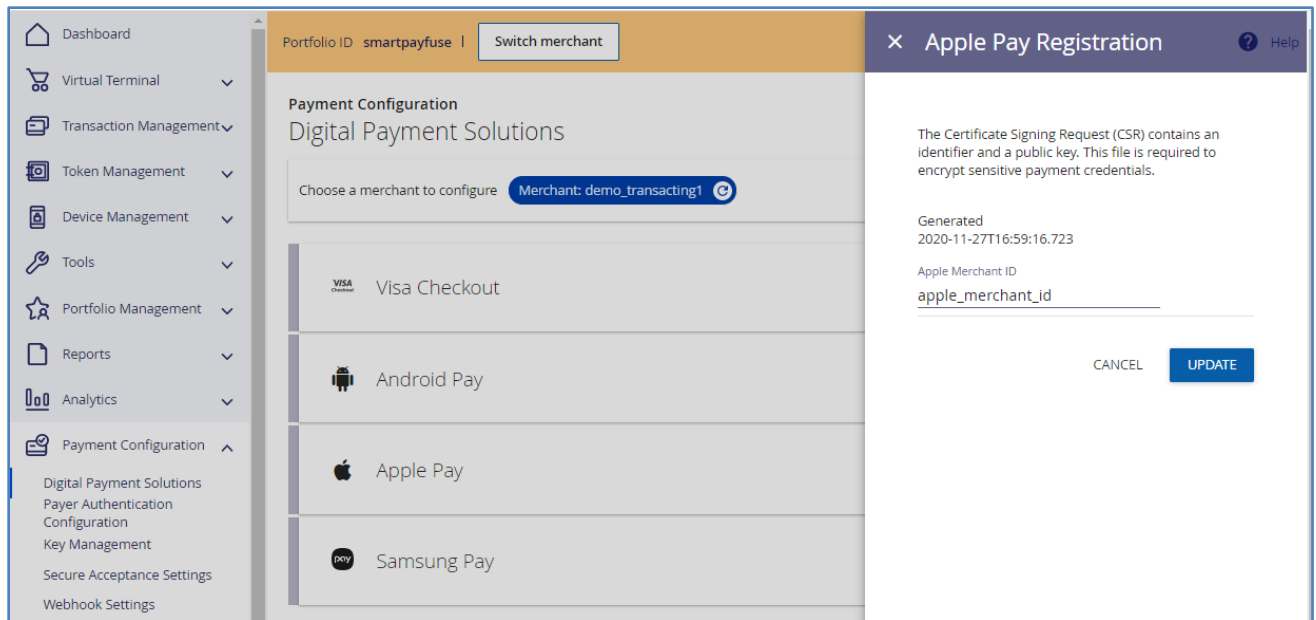
Click on the Download icon to download text file with CSR.



Below is a CSR file example generated in EBC:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBezCbugIBADBaMQswCQYDVQQGEwJVUzETMBEGA1UECAwKV2FzaGluZ3RvbjEj
MAsGA1UEBwwEVk1TQTENMAsGA1UECgwEVk1TQTEYMBYGA1UECwwPQ3liZXJzb3Vy
Y2UuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEId3fOMfnfIzn5s1uvfNz
QV1FppQzZrwrmlfYh50oeNDEKga9StC7usxgEik/3UUF/6iCAdi1ZiTUyeaDefxf
wzAKBggqhkJOPQQDAgNIADBFAiAp10gDZ5jJmoNUkOykj1f2++sHfXYbA321cew+
pgUrGgIhALIAa3d4784+Q9ZZ6H3ZUcmyNuCJaRr9Hcr1kFT18Bo2
-----END CERTIFICATE REQUEST-----
```

If you want to update the CSR file, click on the Update icon (next to Download icon) and then click the **UPDATE** button.



IMPORTANT

You will need different CSR files for Test and Live environments.

6 REST API Examples Using Apple Pay payload

For a detailed description of REST API integration, including authentication methods, method please refer to **M16: REST API Integration**

6.1 Apple Pay Authorisation request

This example shows an authorization request using the Apple Pay payload in place of the card details.

```
{
  "clientReferenceInformation": {
    "code": "12345678ABCD"
  },
  "processingInformation": {
    "paymentSolution": "001",
    "authorizationOptions": {
      "ignoreCvResult": "true",
      "ignoreAvsResult": "false"
    }
  },
  "paymentInformation": {
    "fluidData": {
      "descriptor": "RklEPUNPTU1PTi5BUFBMRS5JTkFQUC5QQV1NRU5U",
      "value": "eyJkYXRhIjoizEJjdmtsTzZpZWJsT3k2VkQydyswY3crWEs0cjBRb1FCM05RbXQ3WFQ3UmJlUjdST3ZLMGNZajVocHh2U3BwUjM4Y0tkWTY4RTV3UitYWZQUkpmU2FBuk9Txc9LRloyalRubW1cL2VKXC9lM3BcL2dYN1-----QVZvakE9PSJ9",
      "encoding": "Base64"
    }
  },
  "orderInformation": {
    "amountDetails": {
      "currency": "GBP",
      "totalAmount": "10"
    },
    "billTo": {
      "address1": "221B Baker Street",
      "country": "GB",
      "locality": "London",
      "email": "sherlock221b@example.com",
      "postalCode": "NW1 6XE"
    }
  }
}
```

Field name	Field description	Recommended values
paymentInformation->value	The encrypted payment data value (payload). Populate this field with the encrypted payment data value obtained from the paymentData property of the PKPaymentToken object.	Payload length may be very long and it is completely up to Apple consideration to change the length of payload.
paymentInformation->descriptor	Format of the encrypted payment data – constant value	RKIEPUNPTU1PTi5BUFBMRS5JTk FQUC5QQVINRU5U
paymentInformation->encoding	Encoding method used to encrypt the payment data	Base64
processingInformation->paymentSolution	Identifies Apple Pay as the payment solution that is being used for the transaction:	001
processingInformation->authorizationOptions->ignoreCvResult	Allows to proceed with capture, even if authorization receives CVN decline.	true as Apple Pay doesn't have CVN field
processingInformation->authorizationOptions->ignoreAvsResult	Allows to proceed with capture, even if authorization receives AVS decline.	false if real billing address is used true if dummy billing address is used

If you do not collect customer billing address, you must use the dummy address fields below.

Address:	1295 Charleston Road
City:	Mountain View
State:	CA
Post Code:	94043
Country	US

6.2 Apple Pay Authorisation reply

This example shows the response to an Apple Pay authorization request

```
{
  "_links": {
    "authReversal": {
      "method": "POST",
      "href": "/pts/v2/payments/6079705955236559904001/reversals"
    },
    "self": {
      "method": "GET",
      "href": "/pts/v2/payments/6079705955236559904001"
    },
    "capture": {
      "method": "POST",
      "href": "/pts/v2/payments/6079705955236559904001/captures"
    }
  },
  "clientReferenceInformation": {
    "code": "12345678ABCD"
  },
  "id": "6079705955236559904001",
  "orderInformation": {
    "amountDetails": {
      "authorizedAmount": "10.00",
      "currency": "GBP"
    }
  },
  "paymentAccountInformation": {
    "card": {
      "type": "001"
    }
  },
  "paymentInformation": {
    "tokenizedCard": {
      "expirationYear": "2021",
      "prefix": "411111",
      "expirationMonth": "07",
      "suffix": "1111",
      "type": "001"
    }
  },
  "processorInformation": {
    "approvalCode": "10",
    "networkTransactionId": "123456789012345",
    "transactionId": "123456789012345",
    "responseCode": "0",
    "avs": {
      "code": "U",
      "codeRaw": "00"
    }
  },
  "status": "AUTHORIZED",
  "submitTimeUtc": "2020-12-14T18:29:55Z"
}
```

Disclaimer

Barclays and Barclaycard offers corporate banking products and services to its clients through Barclays Bank PLC. This presentation has been prepared by Barclays Bank PLC ("Barclays"). This presentation is for discussion purposes only, and shall not constitute any offer to sell or the solicitation of any offer to buy any security, provide any underwriting commitment, or make any offer of financing on the part of Barclays, nor is it intended to give rise to any legal relationship between Barclays and you or any other person, nor is it a recommendation to buy any securities or enter into any transaction or financing. Customers must consult their own regulatory, legal, tax, accounting and other advisers prior to making a determination as to whether to purchase any product, enter into any transaction of financing or invest in any securities to which this presentation relates. Any pricing in this presentation is indicative. Although the statements of fact in this presentation have been obtained from and are based upon sources that Barclays believes to be reliable, Barclays does not guarantee their accuracy or completeness. All opinions and estimates included in this presentation constitute Barclays' judgement as of the date of this presentation and are subject to change without notice. Any modelling or back testing data contained in this presentation is not intended to be a statement as to future performance. Past performance is no guarantee of future returns. No representation is made by Barclays as to the reasonableness of the assumptions made within or the accuracy or completeness of any models contained herein.

Neither Barclays, nor any officer or employee thereof, accepts any liability whatsoever for any direct or consequential losses arising from any use of this presentation or the information contained herein, or out of the use of or reliance on any information or data set out herein.

Barclays and its respective officers, directors, partners and employees, including persons involved in the preparation or issuance of this presentation, may from time to time act as manager, co-manager or underwriter of a public offering or otherwise deal in, hold or act as market-makers or advisers, brokers or commercial and/or investment bankers in relation to any securities or related derivatives which are identical or similar to any securities or derivatives referred to in this presentation.

Copyright in this presentation is owned by Barclays (© Barclays Bank PLC, 2012). No part of this presentation may be reproduced in any manner without the prior written permission of Barclays.

Barclays Bank PLC is a member of the London Stock Exchange.

Barclays is a trading name of Barclays Bank PLC and its subsidiaries. Barclays Bank PLC is registered in England and authorised and regulated by the Financial Services Authority (FSA No. 122702). Registered Number is 1026167 and its registered office 1 Churchill Place, London E14 5HP.

Barclaycard is a trading name of Barclays Bank PLC and Barclaycard International Payments Limited. Barclays Bank PLC is authorised by the Prudential Regulation Authority and regulated by the Financial Conduct Authority and the Prudential Regulation Authority (Financial Services Register number: 122702). Registered in England No. 1026167. Registered Office: 1 Churchill Place, London E14 5HP. Barclaycard International Payments Limited, trading as Barclaycard, is regulated by the Central Bank of Ireland. Registered Number: 316541. Registered Office: One Molesworth Street, Dublin 2, Ireland, D02 RF29. Directors: Paul Adams (British), James Kelly, Mary Lambkin Coyle and Michael Reed (USA). Calls may be recorded for security and other purposes.