barclaycard

Smartpay Advance

Google Pay Integration Guide

for use with Smartpay Advance payment gateway
Web Payment SOAP API and Payments REST API

Version 07

10 April 2024

Table of Contents

1 Overview

Google Pay™ makes it easier for customers to checkout, allowing them to pay with a payment card stored in their Google Pay digital wallet. You can either add the Google Pay button to your checkout page or you may use the Barclaycard Secure Hosted Payment Page. When the customer clicks on the Google Pay button they are presented with a list of payment cards stored in their wallet. They can then select a card and Google Pay will return an encrypted payment token to be used for payment in Smartpay in place of the card detail fields.

1.1 Integration Options and Features

Smartpay Integration methods

- *Secure Hosted Payment Page* - Smartpay manages all communications with Google Pay and automatically decrypts payload before processing authorisation request.
- *Google Pay direct* - Merchant requests encrypted payload directly from Google Pay, passes it to Smartpay for decryption before it processes authorisation request.

Smartpay APIs

Merchants may integrate their e-commerce website or application with Smartpay Advance using either the SOAP API or REST API. This document describes the message flow required to process Google Pay transactions, for all other payment options you should refer to the relevant integration guide:

- **Web Payment SOAP API** (from v30) – download *Web Payment Web Service Integration Guide* here
- **Payments REST API** - download *Payments REST API Integration Guide* here

Smartpay supports the following Google Pay features:

Google Pay Integration methods

- Web integration, supported on all major browsers

Payment type

- CARD

Authentication methods

- PAN_ONLY
- CRYPTOGRAM_3DS

Card networks

- AMEX
- MASTERCARD
- VISA

Recurring Payments not Supported

Please note that Google Pay and the card schemes don't currently support recurring payments.

1.2 Google Pay Authentication Status and ECI Value

Since Google Pay doesn't provide the authentication status in its response to a tokenised card payment request (i.e. where the payload contains a 3DS cryptogram instead of a PAN), Smartpay will determine the status from the ECI value and `cardHolderAuthenticated` parameter found in the decrypted token. However, Google Pay doesn't always include the ECI in the token (e.g. for an in-app transaction, or it's just not been provided by the issuer).

Cardholder Authenticated - ECI Value Returned

If the `cardHolderAuthenticated` parameter is *true* and the decrypted token contains an ECI value, Smartpay will determine the authentication status from the ECI.

Cardholder Authenticated - No ECI Value Returned

If the `cardHolderAuthenticated` parameter is *true* and the decrypted token does NOT contain an ECI value, but does contain a CAVV, then Smartpay will inject one of the following ECI values based on the card scheme:

- Mastercard – 06
- Visa – 07
- Amex – 20
- Diners - 04

In each of the above cases the authentication status is set to *AUTHENTICATED*.

Cardholder Not Authenticated or PAN Token

If the `cardHolderAuthenticated` parameter is *false*, or the decrypted token contains a PAN (rather than 3DS cryptogram), the transaction will be processed as a standard Payer Authentication transaction (providing authentication has been requested).

1.3 Enabling Google Pay in Smartpay

Before you can start accepting payments from Google Pay using Smartpay you'll need to request your Barclaycard account manager to enable Google Pay within your merchant's configuration, if not done so already; they will also manage all required certificates and keys with Google Pay.

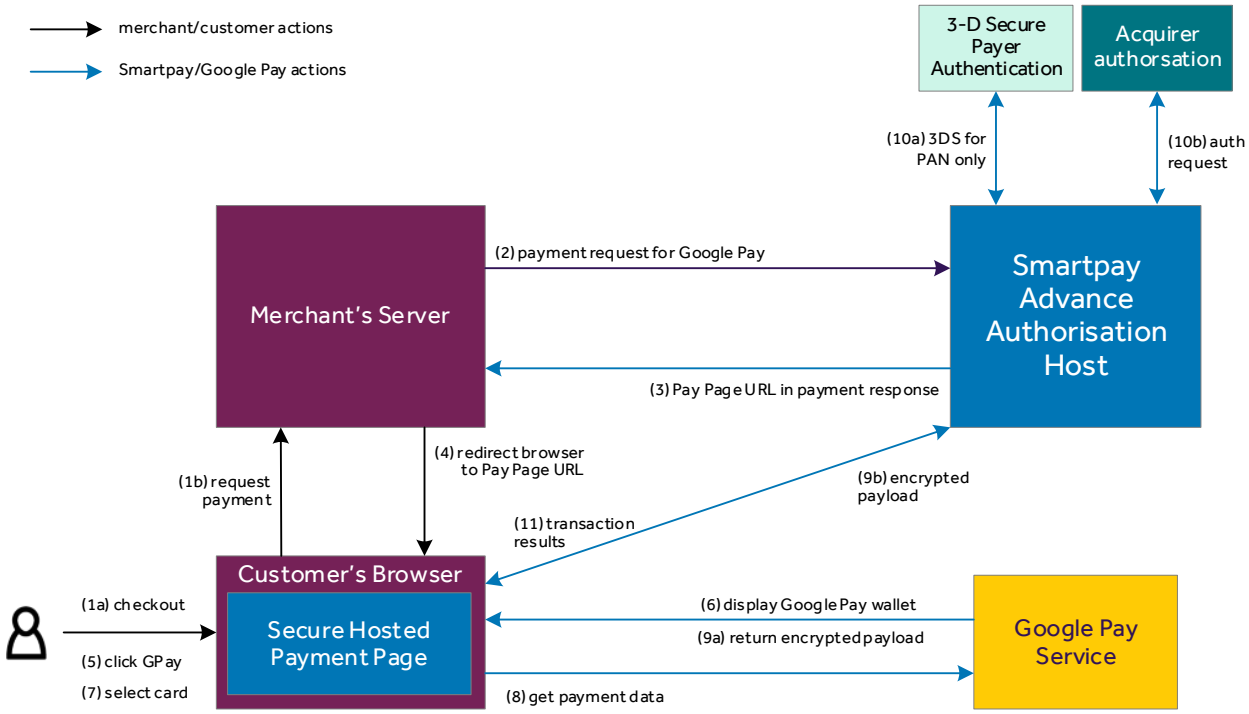
This document describes how to integrate either using the Hosted Payment Page, or alternatively integrating directly with Google Pay. The former being the simplest approach where you need only send messages to Smartpay; whereas the latter gives you greater control but requires you to set up a Google Pay merchant account and submit messages to both Google Pay API and the Smartpay Web Payments SOAP API or Payments REST API.

2 Integrating via Hosted Payment Page

Using the Hosted Payment Page for taking Google Pay payments is much simpler than the direct integration method described below. You only have to integrate directly with Smartpay, and you don't need a Google Pay merchant account or integrate directly with Google Pay. Smartpay and the Hosted Payment Page will manage all necessary communications with Google Pay on your behalf.

Please note that the allowed network cards must be configured in your merchant configuration by your Barclaycard account manager. They can also specify the style of Google Pay button that appears on the Payment Page. If the customer has cards in their wallet that are not in the 'allowed' list then those cards will be greyed-out/disabled in their wallet.

2.1 Payment Flow using Payment Page



1. Customer selects the payment button on merchant checkout page.
2. Merchant server sends a payment request message to Smartpay with data that includes the *GooglePay* payment method, authorisation type, and merchant's store result page URL.
3. Initial response will include data such as the transaction status and redirection URL to the Payment Page.
4. Redirect the customer's browser to the Payment Page. This contains an embedded iFrame that will display the various payment options available, but first Smartpay will query the Google Pay API to determine the user's ability to pay (e.g. the browser supports Google Pay, and the user either has a saved payment method or can add one). If the user is able to pay using Google Pay then the Payment Page will display the Google Pay button; otherwise, it will not be displayed.
5. Customer clicks the Google Pay button.
6. Google Pay displays the customer's digital wallet.
7. Customer selects a card to be used for payment.
8. Payment Page requests Google Pay payment data.
9. Google Pay returns payment data object containing the encrypted payload; Payment Page passes this to Smartpay.
10. Smartpay decrypts the Google Pay payload to extract the payment credential (either PAN or cryptogram), then sends transaction details first to 3DS if it's a PAN then to the acquirer for authorisation; or for a cryptogram, directly to authorisation. Fraud checking may also be performed if it's a PAN (not shown in diagram), but if it's a cryptogram then this will be skipped.
11. Transaction results are returned to Payment Page.

2.2 Web Payment SOAP API

This section is only applicable to the Web Payment SOAP API; for the REST API, see *2.3 Payments REST API*.

2.2.1 Smartpay Request

In order to process a Google Pay transaction be sure to include the following fields in the `webPaymentRequest`, in addition to the standard payment request fields:

- `paymentMethod` = *GooglePay*
- `authType` = *AuthOnly* or *AuthAndSettle*
- `storeResultPage` = your merchant page that displays the transaction results
- `version` = *30*, or above

2.2.2 Smartpay Response

Providing you set the version to 30, or above, in the request then the response will contain these additional fields specific to Google Pay:

```
<configParams>
    <googlePayGatewayMerchantId> MerchantIdentifier </googlePayGatewayMerchantId>
    <googlePayMerchantId> 12345678901234567890 </googlePayMerchantId>
</configParams>

<wallet>
    <type>GooglePay</type>
    <id>GP</id>
    <authMethod>PAN_ONLY</authMethod>
</wallet>
```

2.3 Payments REST API

Using the Hosted Payment Page for taking Google Pay payments is much simpler than the direct integration method described below. You only have to integrate directly with Smartpay, and you don't need a Google Pay merchant account or integrate directly with Google Pay. Smartpay and the Hosted Payment Page will manage all necessary communications with Google Pay on your behalf.

Please note that the allowed network cards must be configured in your merchant configuration by your Barclaycard account manager. They can also specify the style of Google Pay button that appears on the Payment Page. If the customer has cards in their wallet that are not in the 'allowed' list then those cards will be greyed-out/disabled in their wallet.

2.3.1 Smartpay Request

In order to process a Google Pay transaction be sure to include the following fields in the payment POST request, in addition to the standard payment request fields:

- `type` = *AuthOnly* or *AuthAndCapture*
- `paymentMethodType` = *GooglePay*
- `urlAddress` = your merchant page that displays the transaction results

2.3.2 Smartpay Response

The response will contain these fields specific to Google Pay:

- `walletType` = *GooglePay*

- `walletId` = *GP*
- `authMethod` = *PAN_ONLY* or *CRYPTOGRAM_3DS*

Please note that *CV2AVS_CHECK* status is not returned for a Google Pay transaction; and the `transactionResponse/cvvResponse` object is ignored.

Payment Choice State

If you initiated a payment request without specifying a `paymentMethodType`, so that it's currently in the *PAYMENT_CHOICE* state in the response, then you can submit a PATCH update request with the `paymentMethodType` set to *GooglePay* in order to provide card details.

3 Integrating Directly to Google Pay

If you're not using the Hosted Payment Page for taking Google Pay payments then you'll need to integrate directly with Google Pay in order to retrieve the card details in the form of an encrypted payload, and submit to Smartpay in the payment request (using either the SOAP API or REST API). But first you must set up your integration details with Google Pay and request production access.

3.1 Getting Started with Google Pay API

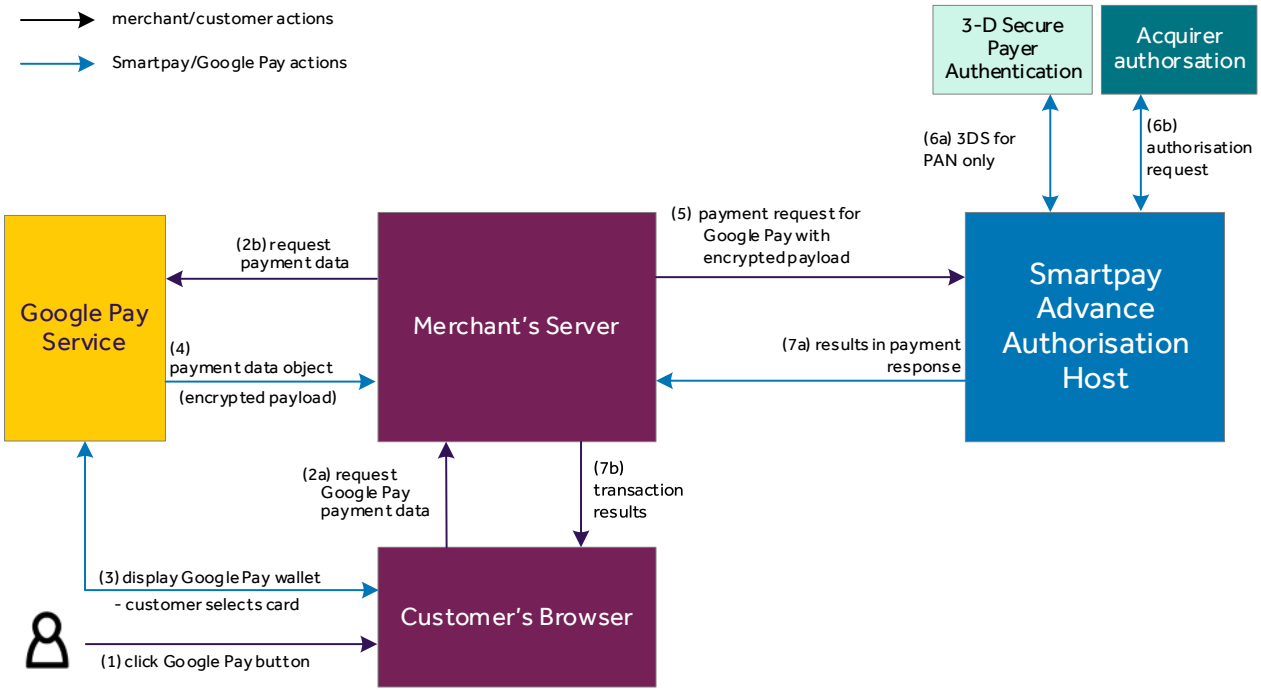
Before you can start using the Google Pay API you must perform the following steps:

1. Review and adhere to the Google Pay API Terms of Service and Acceptable Use Policy
2. Review and adhere to the Google Pay Web Brand Guidelines
3. Complete the tutorial and integration checklist Web integration tutorial and Web Integration Checklist
4. Providing you've fulfilled all criteria in the Integration Checklist then you may request production access through the Business Console.
5. Enter a Business Profile to identify your business with Google. You can enter information such as a business logo, name, support phone numbers or websites.
6. Add your integration type as 'Gateway web'.
7. Upload screenshots of your payment flow as proof you've followed the brand guidelines in order to request production access to the Google Pay API.
8. Once approved Google Pay will assign you a **merchantId**, which will be displayed under your account's *Public merchant profile* setting. You must include this in any requests you send to the Google Pay API production environment; while testing, this field can be set to a dummy value or omitted.
9. It's also advisable to add at least one additional user for the Business Console.
10. Once registered, click the *Google Pay API* tab in the Business Console to get started.

Google Pay Button

Please refer to Google Pay's developer documentation, mentioned above, for instructions on how to add a Google Pay payment button to your checkout page; and how to create a payment data request object in order to retrieve the customer's stored card details in an encrypted payload.

3.2 Payment Flow using Direct Integration



1. Customer clicks Google Pay button on merchant checkout page.
2. Merchant server requests Google Pay payment data object from the Google Pay service.
3. Google Pay displays the customer's digital wallet in the browser and the customer selects one of their stored cards to be used for payment.
4. Google Pay returns payment data object containing encrypted payload, containing all the card details.
5. Merchant server sends a payment request message to Smartpay requesting a Google Pay payment with the encrypted payload.
6. Smartpay decrypts the Google Pay payload to extract the payment credentials (either PAN or cryptogram), then sends transaction details first to 3DS if it's a PAN then to the acquirer for authorisation; or for a cryptogram, directly to authorisation. Fraud checking may also be performed if it's a PAN (not shown in diagram), but if it's a cryptogram then this will be skipped.
7. Authorisation response is returned to merchant who informs customer of the transaction results.

3.3 Requesting Google Pay Encrypted Payload

In your request to the Google Pay API to obtain the encrypted payload be sure to include the following parameters:

- **type** - Smartpay only supports the *CARD* type.
- **allowedAuthMethods** - Smartpay can process both *PAN_ONLY* and *CRYPTOGRAM_3DS* authentication methods.
- **allowedCardNetworks** - specify the card networks that you wish to allow. If the customer has cards in their wallet that are not in the 'allowed' list then those cards will be greyed-out/disabled in their wallet.

- **merchantId** - found in the Google Pay Business Console under your account's *Public merchant profile* setting. Please note that this is only required in Google Pay's production environment; while testing, this field can be set to a dummy value or omitted.
- **gateway** - a unique property that identifies Smartpay Advance as the processor; all encryption keys are associated with this ID. This field must be set to: `barclayssmartpayadvance`
- **gatewayMerchantId** - a property that uniquely identifies the merchant. For Smartpay Advance merchants this field must be set to the *Enterprise ID* given to you by your Barclaycard account manager. For Smartpay Bureau merchants it must be set to your *Merchant Group Identifier*, which will be provided by your implementation consultant.

These fields can be seen in the following JSON example:

```
{
  'type': 'CARD',
  'parameters': {
    'allowedAuthMethods': [
      'PAN_ONLY',
      'CRYPTOGRAM_3DS'
    ],
    'allowedCardNetworks': [
      'AMEX',
      'MASTERCARD',
      'VISA'
    ]
  },
  'tokenizationSpecification': {
    'type': 'PAYMENT_GATEWAY',
    'parameters': {
      'gateway': 'barclayssmartpayadvance',
      'gatewayMerchantId': '<your Smartpay merchant identifier>'
    }
  }
}
```

3.4 Sending Payment Request to Smartpay

Having obtaining the encrypted payload from Google Pay you now need to send it to Smartpay in a payment request. Smartpay will decrypt the payload, which contains a payment credential that's either a PAN or cryptogram, depending on how the card credentials are stored with Google Pay. Either cards are stored on file with Google (PAN_ONLY), and/or a device token on a device that's authenticated with a 3-D Secure cryptogram (CRYPTOGRAM_3DS). The process is described below for each API.

3.5 Web Payment SOAP API

This section is only applicable to the Web Payment SOAP API; for the REST API, see *3.6 Payments REST API*.

3.5.1 Smartpay Request

In order to process a Google Pay transaction be sure to include the following fields in the webPaymentRequest, in addition to the standard payment request fields:

- `paymentMethod` = *GooglePay*
- `authType` = *AuthOnly* or *AuthAndSettle*
- `storeResultPage` = your merchant page that displays the transaction results

- `card.encryptionSessionToken` = encrypted payload received from Google Pay
- `version` = *30*, or above

3.5.2 Sample Web Payment Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:msg="http://services.thelogicgroup.biz/EMIS_WEBPAYMENT_3_0">
<soapenv:Header/>
<soapenv:Body>
<msg:beginWebPayment>
<arg0>
    <requester>
        <authToken>A742DD37E4F7C903B95BAA847F034F65B85AC6739BCA9A4FD80A9B4E5AB8AB75</authToken>
        <enterpriseID>MERCHANT01</enterpriseID>
        <clientID>CLNT01</clientID>
        <transNo>123456</transNo>
        <environment>ECommerce</environment>
        <version>230</version>
    </requester>
    <transactionTime>2022-03-24T11:21:14.894</transactionTime>
    <acquirerReferenceData></acquirerReferenceData>
    <authType>AuthAndSettle</authType>
    <authenticate>true</authenticate>
    <billingAddress>
        <city>Fleet</city>
        <country>GBR</country>
        <county>Hampshire</county>
        <firstName></firstName>
        <lastName></lastName>
        <line1>123 High Street</line1>
        <line2></line2>
        <line3></line3>
        <middleName></middleName>
        <name>Billing Address</name>
        <postcode>GU51 3SB</postcode>
        <stateCode></stateCode>
    </billingAddress>
    <billingEmail>billing@email.com</billingEmail>
    <card>

<encryptionSessionToken>{"signature":"MEUCIQC9z5vgyTuGq9EEzMkXb+SDahEf/IrcEFt0adND4WlN4QIgB0wdbeEY
nIld0iavV8SWjkDFGRy2X6ML/iTQdWNkKRk\u003d","intermediateSigningKey":{"signedKey":"{\"keyValue\":\"
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE77m3kK3OPcm/oCdveKcE9AhzSC+YuBQMCT80FQ/MB99Eh7Ope+4eBHKWYugLER
QJ8RWnzo8qYzOu+87J9jq5Bw\\u003d\\u003d\",\"keyExpiration\":\"1648784148503\"}","signatures":["MEUC
IBknp3rAcnZqzHvAhpXX2stpyW5uCPVhXV6U/7AXXSKEAiEAnKT9BS+j1cJMZnwaPYKjOmgSv045FDRguqqB4LrCLFU\u003d"
]},"protocolVersion":"ECv2","signedMessage":"{\"encryptedMessage\":\"VCyI+RAF2mG14DZgyexA9aN5E7xRW
h1MEPn4lAT2qsURza204hZ3cVjRBSoWTZs6yRUGlxgumUV/CI6aeV4wB3h60ItzRx00unXO+4ayPFpk9bHsOfrJUz4KV080azU
uX84byQ0LF198784G9eWIkn1M6iFLL0ZvaL1hLfSEns7RhObTv9Fv+99xyHVTN3Hjoi0umOSqixwzroL1kjWG/Po7fb19dQH1H
Gb6b8NTHjXVIjeZqh8T1rfnfXIJjbJpGI4cp1GWzlfgG77nz9TD0GQ5KpOrj61fM+bv+JC9WBu9KLHCg/unXcmxU47XQTKoScH
KzlZggVayEmWHH9GshRdxpdWRZ2D9TGNK0N3fIOix4InWfIEgn7tQ5I5iScfUTrfBO7/ANR98VxT87kfjCWlZZOvOjZb7nFnS1
9gUsg8YFcYojPaSWV1+dSJx5HcL7Wu28b9F351Wqj/pH+GBcWTzXpi1jROjshLxA4P778rUaZ4dFP32vqt1MttOYjTxi8C7yDJ
LZTLgoPWkIPf4RBCkhB4WUm9sbng33QblCC5Q+xHmmTvWJ0lnhSJeFwKxP/2/sTAg\\u003d\\u003d\",\"ephemeralPubli
cKey\":\"BEcYTfI5vcLj9lP4LlXG/Z1qSyYdXUzFN0f9AaW3NeHKnSjrLqxXEWKuQkfPsRvpp0tOECGGj3fVczW48k/HwMY\\
u003d\",\"tag\":\"iGo2HBPL8OyiYk1zaMTxhLVe5CbMBsyuvTs2pBamdYk\\u003d\"}"}</encryptionSessionToken>
    </card>
    <currencyCode>GBP</currencyCode>
    <customerID></customerID>
    <dcc checkDCC="false" performDCC="false"/>
    <fraudProfile></fraudProfile>
    <fraudProvider></fraudProvider>
    <fraudType>BeforeAuthorisation</fraudType>
    <invoiceID></invoiceID>
    <merchantTransactionID>0009</merchantTransactionID>
    <operatorID></operatorID>
    <paymentMethod>GooglePay</paymentMethod>
```

```
    <purchaseAmount>24999</purchaseAmount>
    <purchaseDescription>Samsung Galaxy S7 Edge</purchaseDescription>
    <salesDetails>Testing</salesDetails>
    <schemeReferenceData></schemeReferenceData>
    <shippingAddress>
        <city>Fleet</city>
        <country>GBR</country>
        <county>Hampshire</county>
        <firstName></firstName>
        <lastName></lastName>
        <line1>123 High Street</line1>
        <line2></line2>
        <line3></line3>
        <middleName></middleName>
        <name>Shipping Address</name>
        <postcode>GU51 3SB</postcode>
        <stateCode></stateCode>
    </shippingAddress>
    <storeResultPage> http://localhost:8080/DemoShop/response.jsf </storeResultPage>
    <styleSheetID></styleSheetID>
    <validate>false</validate>
</arg0></msg:beginWebPayment>
</soapenv:Body>
</soapenv:Envelope>
```

3.5.2.1 Encrypted Payload

Note how the encrypted payload received from Google Pay is included in the request as Base64 text (`encryptionSessionToken`). After receiving the payload Smartpay will decrypt it using Google keys to extract the payment credential, which will be either a PAN or 3DS cryptogram, depending on the Google Pay **authMethod**. Please note that Smartpay uses the PAN for the current transaction only and doesn't store it within its database. Here's examples of what the payload might contain as seen by Smartpay once decrypted:

PAN Only example

The following code snippet shows the JSON payload, decrypted by Smartpay, for a PAN_ONLY **authMethod**:

```
{
  "paymentMethod": "CARD",
  "paymentMethodDetails": {
    "authMethod": "PAN_ONLY",
    "pan": "4444333322221111",
    "expirationMonth": 10,
    "expirationYear": 2025,
    "assuranceDetails": {
      "accountVerified": true,
      "cardHolderAuthenticated": false
    }
  },
  "gatewayMerchantId": "some-merchant-id",
  "messageId": "some-message-id",
  "messageExpiration": "1577862000000"
}
```

Cryptogram 3DS example

The following code snippet shows the JSON payload, decrypted by Smartpay, for a CRYPTOGRAM_3DS **authMethod**. Note the additional fields for **cryptogram** and **eciIndicator** (the 3DS authentication flag that may be returned for tokens on the Visa card network).

```
{
  "paymentMethod": "CARD",
  "paymentMethodDetails": {
    "authMethod": "CRYPTOGRAM_3DS",
```

```
    "pan": "4444333322221111",
    "expirationMonth": 10,
    "expirationYear": 2025,
    "cryptogram": "AAAAAA...",
    "eciIndicator": "eci indicator"
      "assuranceDetails": {
      "accountVerified": true,
      "cardHolderAuthenticated": true
    }
  },

  "gatewayMerchantId": "some-merchant-id",
  "messageId": "some-message-id",
  "messageExpiration": "1577862000000"
}
```

PAN payments

If the payment credential is a PAN then the payment flow will automatically follow the standard 3-D Secure authentication and authorisation flow (unless it's subject to TRA exemption).

3DS Cryptogram payments

If the payment credential is a cryptogram then provided **cardholderAuthenticated** is *true* in the Google Pay payload then it is exempt from 3-D Secure authentication and so the transaction will proceed directly to authorisation. Authentication has already been undertaken on the device and so having the token cryptogram and correct ECI meets PSD2 SCA requirements and provides liability shift to protect the merchant against fraud.

The transaction status flow for Google Pay payments follows the basic card payment flow, apart from a couple of differences depending on the contents of the payload. Fraud checking via Kount or ACI, and CV2/AVS checking, may be performed only when the payment credential in the decrypted payload is a PAN; otherwise for a 3DS cryptogram these stages will be skipped.

However, If the payment credential is a cryptogram and **cardholderAuthenticated** is *false* in the Google Pay payload, payment flow will follow the standard 3-D Secure authentication and authorisation flow (unless it's subject to TRA exemption).

When an intermediate response is received the transaction status will determine the next action to be taken. In some cases this will require you to redirect the customer's browser to a URL supplied in the response, or it may require you to decide whether to continue with the transaction, by submitting an update request; alternatively, you may decide to abort the transaction by submitting a cancel request.

3.5.3 Sample Web Payment Response for Google Pay

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<ns2:beginWebPaymentResponse xmlns:ns2="http://services.thelogicgroup.biz/EMIS_WEBPAYMENT_3_0">
    <return>
        <acquirer>
            <ID>R</ID>
            <name>Mastercard Debit</name>
        </acquirer>
        <authType>AuthAndSettle</authType>
        <authenticationResponse>
            <cardScheme>MASTERCARD</cardScheme>
            <cavv>3R18UYQOIKLTENPM9Q413PW9ND3W</cavv>
            <cavvAlgorithm>ucaf<\cavvAlgorithm>
            <eci>01</eci>
            <errorCode>0</errorCode>
            <status>AUTHENTICATED</status>
            <threeDSVersionNumber>UNKNOWN</threeDSVersionNumber>
        </authenticationResponse>
```

```
        <authorisationResponse>
            <authcode>D12345</authcode>
            <hostResponseCode>00</hostResponseCode>
            <hostResponseMessage>Approved</hostResponseMessage>
            <merchantNo>1211118</merchantNo>
            <paymentMethod>GooglePay</paymentMethod>
            <softDeclined>false</softDeclined>
            <tid>23368318</tid>
        </authorisationResponse>
        <bank>
            <ID>B</ID>
            <name>Barclays-Dummy-0001</name>
        </bank>
        <billingAddress>
            <city>Fleet</city>
            <country>GBR</country>
            <county>Hampshire</county>
            <firstName/>
            <lastName>Billing Address</lastName>
            <line1>123 High Street</line1>
            <line2/>
            <line3/>
            <name>Billing Address</name>
            <postcode>GU51 3SB</postcode>
            <stateCode/>
        </billingAddress>
        <billingEmail>billing@email.com</billingEmail>
        <captures>
            <capture id="1">
                <amount>24999</amount>
                <outcome>OK</outcome>
            </capture>
        </captures>
        <card>
            <source>keyed</source>
            <maskedPAN>444433******1111</maskedPAN>
            <onlineToken>4444333322221111</onlineToken>
            <updateOnlineToken>true</updateOnlineToken>
            <onlineTokenUpdated>false</onlineTokenUpdated>
            <expiry>2027-12</expiry>
            <wallet>
                <type>GooglePay</type>
                <id>GP</id>
                <authMethod>PAN_ONLY</authMethod>
            </wallet>
            <cardType>V</cardType>
        </card>
        <configParams>
            <displayBasketDetails>false</displayBasketDetails>
            <merchantStates/>
            <storeID>100001</storeID>
        </configParams>
        <currencyCode>GBP</currencyCode>
        <customerID/>
        <cvResponse>
            <rawCv2>no_information</rawCv2>
            <rawAddress>no_information</rawAddress>
            <rawPostcode>no_information</rawPostcode>
            <result>NoDecision</result>
        </cvResponse>
        <environment>ECommerce</environment>
        <merchantTransactionID>0009</merchantTransactionID>
        <purchaseAmount>24999</purchaseAmount>
        <purchaseDescription>Samsung Galaxy S7 Edge</purchaseDescription>
```

```
        <requester>
            <enterpriseID>MERCHANT01</enterpriseID>
            <clientID>CLNT01</clientID>
            <transNo>123456</transNo>
            <version>30</version>
        </requester>
        <responder>
            <name>EMIS-AG</name>
            <version>3.55.2.0 Build:c8eaf03b32dc.PROD</version>
            <releaseDate>09/02/2022</releaseDate>
            <id>TSI-COR301-P1</id>
        </responder>
        <response>validated</response>
        <salesDetails>Testing</salesDetails>
        <schemeReferenceData>01000000003755712AB</schemeReferenceData>
        <shippingAddress>
            <city>Fleet</city>
            <country>GBR</country>
            <county>Hampshire</county>
            <firstName/>
            <lastName>Shipping Address</lastName>
            <line1>123 High Street</line1>
            <line2/>
            <line3/>
            <name>Shipping Address</name>
            <postcode>GU51 3SB</postcode>
            <stateCode/>
        </shippingAddress>
        <status>CAPTURED</status>
        <storeResultPage> http://localhost:8080/DemoShop/response.jsf </storeResultPage>
        <styleSheetID/>
        <totalAmount>24999</totalAmount>
        <transactionReference>bd88eefd-94d6-4850-a570-8c7bf4329e1b</transactionReference>
        <validCardTypes>Amex</validCardTypes>
        <validCardTypes>Visa Business Debit</validCardTypes>
        <validCardTypes>Visa Debit</validCardTypes>
        <validCardTypes>Visa Electron Prepaid</validCardTypes>
        <validCardTypes>VISA</validCardTypes>
        <validCardTypes>Mastercard Commercial Prepaid</validCardTypes>
        <validCardTypes>Visa Business Prepaid</validCardTypes>
        <validCardTypes>International Maestro</validCardTypes>
        <validCardTypes>Visa Electron</validCardTypes>
        <validCardTypes>Mastercard Commercial Prepaid Credit</validCardTypes>
        <validCardTypes>Mastercard Prepaid</validCardTypes>
        <validCardTypes>Visa Prepaid</validCardTypes>
        <validCardTypes>Mastercard Debit</validCardTypes>
        <validCardTypes>Mastercard Commercial Debit</validCardTypes>
        <validCardTypes>Mastercard</validCardTypes>
        <validPayment>GooglePay</validPayment>
    </return>
</ns2:beginWebPaymentResponse>
</soap:Body>
</soap:Envelope>
```


3.5.4 Payment Choice State

If you initiated a web payment request without specifying a `paymentMethod`, so that it's currently in the *PAYMENT_CHOICE* state, and *GooglePay* is listed as an available `validPayment` in the response, then you can submit an `updateWebPayment` request with the `paymentMethod` set to *GooglePay* in order to provide card details.

3.6 Payments REST API

This section is only applicable to the Payments REST API; for the SOAP API, see *3.5 Web Payment SOAP API*.

3.6.1 Smartpay Request

In order to process a Google Pay transaction be sure to include the following fields in the payment request, in addition to the standard payment request fields:

- `paymentMethodType` = *GooglePay*
- `type` = *AuthOnly* or *AuthAndCapture*
- `urlAddress` = your merchant page that displays the transaction results
- `encryptionSessionToken` = encrypted payload received from Google Pay

3.6.1.1 Encrypted Payload

Note how the encrypted payload received from Google Pay is included in the request as Base64 text (`encryptionSessionToken`). After receiving the payload Smartpay will decrypt it using Google keys to extract the payment credential, which will be either a PAN or 3DS cryptogram, depending on the Google Pay **authMethod**. Please note that Smartpay uses the PAN for the current transaction only and doesn't store it within its database. Here's examples of what the payload might contain as seen by Smartpay once decrypted:

PAN Only example

The following code snippet shows the JSON payload, decrypted by Smartpay, for a PAN_ONLY **authMethod**:

```
{
  "paymentMethod": "CARD",
  "paymentMethodDetails": {
    "authMethod": "PAN_ONLY",
    "pan": "4444333322221111",
    "expirationMonth": 10,
    "expirationYear": 2025,
    "assuranceDetails": {
      "accountVerified": true,
      "cardHolderAuthenticated": false
    }
  },
  "gatewayMerchantId": "some-merchant-id",
  "messageId": "some-message-id",
  "messageExpiration": "1577862000000"
}
```

Cryptogram 3DS example

The following code snippet shows the JSON payload, decrypted by Smartpay, for a CRYPTOGRAM_3DS **authMethod**. Note the additional fields for **cryptogram** and **eciIndicator** (the 3DS authentication flag that may be returned for tokens on the Visa card network).

```
{
  "paymentMethod": "CARD",
  "paymentMethodDetails": {
    "authMethod": "CRYPTOGRAM_3DS",
    "pan": "4444333322221111",
    "expirationMonth": 10,
    "expirationYear": 2025,
    "cryptogram": "AAAAAA...",
    "eciIndicator": "eci indicator"
      "assuranceDetails": {
      "accountVerified": true,
```

```
      "cardHolderAuthenticated": true
    }
  },

  "gatewayMerchantId": "some-merchant-id",
  "messageId": "some-message-id",
  "messageExpiration": "1577862000000"
}
```

PAN payments

If the payment credential is a PAN then the payment flow will automatically follow the standard 3-D Secure authentication and authorisation flow (unless it's subject to TRA exemption).

3DS Cryptogram payments

If the payment credential is a cryptogram then provided **cardholderAuthenticated** is *true* in the Google Pay payload then it is exempt from 3-D Secure authentication and so the transaction will proceed directly to authorisation. Authentication has already been undertaken on the device and so having the token cryptogram and correct ECI meets PSD2 SCA requirements and provides liability shift to protect the merchant against fraud.

The transaction status flow for Google Pay payments follows the basic card payment flow, apart from a couple of differences depending on the contents of the payload. Fraud checking and CV2/AVS checking may be performed only when the payment credential in the decrypted payload is a PAN; otherwise for a 3DS cryptogram these stages will be skipped.

However, If the payment credential is a cryptogram and **cardholderAuthenticated** is *false* in the Google Pay payload, payment flow will follow the standard 3-D Secure authentication and authorisation flow (unless it's subject to TRA exemption).

When an intermediate response is received the transaction status will determine the next action to be taken. In some cases this will require you to redirect the customer's browser to a URL supplied in the response, or it may require you to decide whether to continue with the transaction, by submitting an update request; alternatively, you may decide to abort the transaction by submitting a cancel request.

3.6.2 Smartpay Response

The response will contain these fields specific to Google Pay:

- `walletType` = *GooglePay*
- `walletId` = *GP*
- `authMethod` = *PAN_ONLY* or *CRYPTOGRAM_3DS*

Please note that *CV2AVS_CHECK* status is not returned for a Google Pay transaction; and the `transactionResponse/cvvResponse` object is ignored.

If you initiated a payment request without specifying a `paymentMethodType`, so that it's currently in the *PAYMENT_CHOICE* state, then you can submit an update request with the `paymentMethodType` set to *GooglePay* in order to provide card details.

What's New

Version 07 – 10 April 2024 - Smartpay Core release 4.3.3.x

- Added support for Secure Hosted Payment Page for Payments REST API - *2.3*

Version 06 - 12 December 2023 - Smartpay Core release 4.2.2.x

- Added Payments REST API -*3.6*

Version 05 -11 October 2022 - Smartpay Core release 4.1.4.x

- Enhancement: ECI injected when not returned by Google Pay in order to determine authentication status - *1.2*

Version 04 - 17 November 2022 - Smartpay Core release 3.59.3.x

- Added Google Pay authMethod in response card > wallet object
- Improved definition of gatewayMerchantID appropriate to Advance and Bureau clients

Version 03 - 15 September 2022 - Smartpay Core release 3.59.1.x

- Updated message code samples

Version 02 - 12 April 2022 - Smartpay Core release 3.56.1.x

- Added support for integration using Secure Hosted Payment Page

Version 01 - 11 March 2022 - Smartpay Core release 3.55.2.x

- Added Google Pay digital wallet payments to Web Payments SOAP API