GATEWAY

INTEGRATION GUIDE

Version: 3.07

VERSION CONTROL

Ver.	Date	Author	Update information
1.00	30/11/19	N Turner	New guide replacing original separate Hosted and Direct Integration Guides.
1.01	24/08/20	N Turner	Updated appendix A-11.1to include settlement outcome in test amounts table.
1.02	23/09/20	N Turner	Updated appendix A-22.2.1 and appendix A-23.1.2 for 3DS v2.
1.03	22/10/20	N Turner	Updated section 5.5.4 to reflect changes to 3DS options.
1.04	25/10/20	N Turner	Removed CUROPT_SSL_VERIFYPEER from sample code.
1.05	25/11/20	N Turner	Removed `threeDSVersion` and `merchantCategoryCode` fields from sample code.
1.06	07/12/20	N Turner	Replaced Merchant ID and MID with Merchant Account ID. Updated appendix A-14 to indicate specific MOTO or CA account may be required.
1.07	08/12/20	N Turner	Added appendix detailing Visa Stored Credentials Framework (Credentials on File).
1.08	23/01/21	N Turner	Updated section 26 to add Google Pay details.
1.09	28/01/21	N Turner	Corrected 'AmazonPay' to 'Amazon Pay' and other minor spelling corrections.
1.10	03/02/21	N Turner	Removed details for eReceipts integration.
1.11	21/02/21	N Turner	Updated appendix A-11 to detail 3DS response simulation.
1.12	18/03/21	N Turner	Added section 26.8 covering the features of Digital Wallet Payment Tokens.
1.13	23/03/21	N Turner	Updated appendix A-11 to add SCA soft-decline test amounts.
1.14	13/06/21	N Turner	Revised appendix A-10 adding more details and support for Discover and Diners Club. Updated appendix A-11 to add further testing details for Referral response testing. Corrected mistakes in the available threeDSOptions in section 5.5.4. Amended section 1.7.8 to remove XML format and explain how to pass boolean values. Added `rtAgreementType`, `rtSequenceCount` and `rtSequenceNumber` fields to section 2.1. Added `rtSequenceCount` and `rtSequenceNumber` fields to appendix A-14.3. Corrected the descriptions for response code's 65794 and 65796 in appendix A-1. Removed the text '(reserved for future use)' from response code descriptions in appendix A-1. Reformatted CSS selectors and attribute names and added details regarding custom fonts to the Hosted Fields Styling appendix A-20.3.10. Added `formAmountEditable` field to appendix A-19. Updated `threeDSEnrolled` and `threeDSAuthenticated` values in section 5 and appendix A-3.
1.15	16/06/21	N Turner	Added further explanations and updated the response codes in appendix A-1.
1.16	22/07/21	S Fowle	Added links to fields mentioned in Response Codes in appendix A-1. Updated Response Code descriptions in appendix A-1. Anonymised content, removing any specific URLs, Merchant Account IDs, etc.
1.17	14/08/21	N Turner	Various formatting changes and clarifications. Added more details to sections 1.7.1 and 1.7.8. Added details on how to switch between latest and legacy 3-D Secure responses to section 5.3. Updated details about mandatory 3-D Secure information to sections 5.3, 5.4 and 5.5.4. Updated details regarding complex fields `threeDSRequest`, `threeDSResponse`, `checkoutRequest` and `checkoutResponse` to sections 5.4, 21.4 and 22.4. Updated details on 3-D Secure requests in sections 5.5 and 5.6. Added details on 3-D Secure fallback to version 1 and PSD2 SCA in section 5.7. Added PSD2 SCA Compliance and exemptions in appendix A-17. Added clarification that Hosted Fields can be used with standard fields in appendix A-20.3.1. Updated and moved Hosted Payment Form Options to appendix A-19. Moved details on Duplicate Transaction checking to appendix A-8. Revised details on Delaying Capture and Settlement in appendix A-9. Moved details on Recurring transaction and Credentials on File to section 12. Removed `rtSequenceCount` and `rtSequenceNumber` fields from section 2.1. Added details on Acquirer specific options and response details in section 18. Added card token details to section 19. Updated cloned fields in appendix A-16.1. Added `cardFlags` field and 'Scheme Code' table column to appendix A-10.
2.00	03/10/21	N Turner	Added example `threeDSOptions` to code samples in appendix A-22.2.1 and A-23.1.2. Added addition information on the 3-D Secure fields to appendix A-3. Added matrix for correct Credentials on File (COF) flagging in new appendix A-17. Added more details on SCA declines and new appendix A-18.2. Added note that Diners Club do not support AVS to appendix A-11.2.9
2.01	29/10/21	N Turner	Added details on maximum transaction storage period to section 1.5, 12.1, 12.3, 13.1 and A-15. Added details on maximum capture and refund periods to section 1.9. Corrected capitalisation of 'Gateway' and 'Card Scheme'.
2.02	19/11/21	N Turner	Added details on Google Pay Merchant Identifier to section 26.3.3.
2.03	15/12/21	N Turner	Added emphasis to use the `callbackURL` when updating backend systems to section 1.7.6.
2.04	13/01/22	N Turner	Correct reference to Android Pay in section 26.3.1.
2.05	15/02/22	N Turner	Replaced reference to 'Hosted Payment Fields SDK Guide' in section 1.3.1.
2.06	28/04/22	N Turner	Updates in preparation of 3-D Secure version 1 being phase out. Added details on HTTP POST parameter limit in section 1.7.1. Corrected card flags in appendix A-10.

3.00	17/05/22	N Turner	Rewrites to how the `redirectURL` is handled in section 1.7.6. Removed all references to 3-D Secure version 1 including Legacy 3-D Secure API appendix. Rewrite of 3-D Secure section 5.4 to show direct integration steps clearer. Amended references to section 5.5.3 that should have been 5.5.4.
3.01	02/06/22	N Turner	Added list of fields which can be tokenised by the Hosted Payment Fields Library to section A-20.3.3 Corrected 'data-hostedform-modal' attribute in sample code in section A-23.2.1.
3.02	09/06/22	N Turner	Corrected response codes 935 and 936 in appendix A-1.1.
3.03	25/07/22	N Turner	Updated appendix A-15 to link to the Gateway Wallet details in section 19. Updated section 20 to cover the new Masterpass Click to Pay implementation. Updated possible values for `action` field in section 3.1.
3.04	03/09/22	N Turner	Corrected some cross references.
3.05	10/09/22	N Turner	Clarified that fields in 'browserInfo' are appended to the request, in appendix A-22.2.1 and A-23.1.3.
3.06	08/11/22	N Turner	Added section heading 5.5.5 to highlight mandatory 3-D Secure Information and then linked it to the 'browserInfo' comment in sample code in appendix A-22.2.1 and A-23.1.3. Corrected spelling of 'hire purchase' in section 12.3.1. Expanded the description for response code 65796 in appendix A-1.2.
3.07	24/11/22	N Turner	Expanded the description for response code 65566 in appendix A-1.2. Clarified IP address formats for `remoteAddress` in section 2.1, `IPAD` in section 6.4.2 and `browserIPAddress` in section 5.5.4. Clarified that the `threeDSXID` contains the Directory Server Transaction ID in appendix A-3.3. Corrected section 26.4 to indicate that Digital Wallets are supported when using the Hosted Integration.

CONTENTS

1 Gateway Integration

1.1 About This Guide

This guide provides the information required to integrate with our Payment Gateway and gives a very basic example of code for doing so. It is expected that you have some experience in server-side scripting with languages such as PHP or ASP; or that an off-the-shelf software package is being used that has inbuilt or plug-in support for our Gateway.

Disclaimer

This guide provides the integration documentation necessary for enabling Merchants to process payments via our Gateway. Whilst every effort has been made to ensure these guides are accurate and complete, we expect Merchants undertaking any integration to test all their technical work fully and satisfy their own standards. The authors of this guide are not responsible or liable for any Merchant or Third-Party integration

1.2 Terminology

The following terms are used throughout this guide:

Gateway
The Payment Gateway.

Merchant
The Merchant using the Gateway's services.

Our
The Payment Gateway Provider.

You/your
The Merchant or its representative performing the integration.

Acquirer
The bank or financial institution used by the Merchant.

Customer
A Customer of the Merchant making a payment.

Card
A payment credit, debit, prepayment or gift card issued by the Card Schemes.

Card Scheme
The operator of a payment Card network, such as Visa, Mastercard, et al.

Cardholder
The person who owns the payment Card, usually the Customer.

Issuer
The bank or financial institution that issued the payment Card to the Cardholder.

Merchant Account
An account on the Gateway mapped to an Acquirer-provided account.

Checkout
Third-party checkout solution such as PayPal, Amazon Pay other alternative payment methods.

Wallet
Third-party wallet solution such as Masterpass.

Hosted Payment Page (HPP)
A page hosted on our secure server used to collect Customer details.

Hosted Payment Field (HPF)
An individual form field hosted on our secure server used to collect sensitive Cardholder data.

1.3 Integration Methods

There are three methods of integration provided to process your transactions through the Gateway, allowing for different levels of control and communication from your website.

1.3.1 Hosted Integration

The Hosted Integration method makes it easy to add secure payment processing to your e-commerce business, using our Hosted Payment Pages (HPP). You can use this method if you do not want to collect and store Cardholder data.

The Hosted Integration method works by redirecting the Customer to our Gateway's Hosted Payment Page, which will collect the Customer's payment details and process the payment before redirecting the Customer back to a page on your website, letting you know the payment outcome. This allows you the quickest path to integrating with the Gateway.

The standard Hosted Payment Page is designed to be shown in a lightbox over your website and styled with logos and colours to match. Alternatively, you can arrange for fully customised Hosted Payment Pages to be produced that can match your website's style and layout. These fully customised pages are usually provided using a browser redirect, displaying full-page in the browser, or can be displayed embedded in an iframe on your website.

For greater control over the customisation of the payment page, our Gateway offers the use of Hosted Payment Fields, as detailed in append, where only the individual input fields collecting the sensitive Cardholder data are hosted by the Gateway while the remainder of the payment form is provided by your website. These Hosted Payment Fields fit seamlessly into your payment page and can be styled to match your payment fields. When your payment form is submitted to your server, the Gateway will submit a payment token representing the sensitive card data it collected and your webserver can then use the Direct Integration to process the payment without ever being in contact with the collected Cardholder data. For more information, please refer to appendix A-20.3.

1.3.2 Direct Integration

The Direct Integration works by allowing you to keep the Customer on your system throughout the checkout process, collecting the Customer's payment details on your own secure server before sending the collected data to our Gateway for processing. This allows you to provide a smoother, more complete checkout process to the Customer.

In addition to basic sales processing, the Direct Integration can be used to perform other actions such as refunds and cancellations, which can provide a more advanced integration with our Gateway.

1.3.3 Batch Integration

The Batch Integration is an enhancement to the Direct Integration, allowing you to send multiple transactions in a single request and monitor their status. This is useful if you wish to capture multiple transactions or collect multiple payments – for example, collecting subscription charges or loan repayments.

In addition to basic sales processing, the Batch Integration can be used to perform other actions, such as refunds and cancellations, which can provide a more advanced integration with our Gateway.

Unlike the Hosted and Direct Integrations, the Batch Integration does not process transactions sent to it immediately. Instead, the Gateway queues these transactions to be processed and returns a batch reference number which can be used to download a file that contains the current status of the transactions.

Batch Processing does not support transactions that require Customer interaction such as 3-D Secure transactions, or alternative payment methods with interactive Wallet or Checkout pages.

1.4 Integration Libraries

We can provide a range of libraries to help you to integrate with the Gateway.

These libraries include simple server-side classes in many popular programming languages, through to client-side scripts to help with the integration of the Hosted Payment Page or Hosted Payment Fields.

For more information about these libraries, please refer to appendix A-20.

1.5 Security and Compliance

Each method requires a different level of server security and compliance with the Payment Card Industry Data Security Standard (PCI DSS).

If you use Hosted Payment Pages with the Hosted Integration or Hosted Payment Fields with the Direct or Batch Integrations, then your webserver does not need an SSL certificate and you require the lowest level of PCI DSS compliance.

If your website collects and/or stores sensitive Cardholder data, such as the card number (PAN) or card security code (CVV/CV2), then your webserver must have an SSL certificate and serve all payment forms using HTTPS. You will also need a higher level of PCI DSS compliance and to complete a PCI validation form annually.

The Gateway will make transaction details available for a maximum period of 13 months, card data will be held as part of the transaction details for that period or for a shorter period at your request. Your Acquirer may hold information for a different period of time.

For more information, please see https://www.pcisecuritystandards.org/

1.6 Prerequisites

You will need the following information to integrate with the Gateway which will be provided during onboarding:

Merchant Account ID	Your unique Merchant Account ID.
Hosted Integration URL	Your unique URL to use the Hosted Integration.
Direct Integration URL	Your unique URL to use the Direct Integration.
Batch Integration URL	Your unique URL to use the Batch Integration.

You will be provided with unique production and test Merchant Account IDs during the onboarding process. You will also be provided with the integration URLs.

You must use the URLs provided and no others as they may be customised to your account.
Any code samples in this guide use dummy Merchant Account IDs in the form 100001 and dummy integration URLs of the form 'https://gateway.example.com/<integration>'. These dummy values must be replaced in full by the values you have been provided with.

For testing purposes, you can use test amounts and test cards provided in appendix A-11 to run a test transaction.

1.7 Integration Details

1.7.1 HTTP Requests

A request can be sent to the Gateway by submitting a HTTP POST[1] request to the integration URL provided.

The request should have a `Content-Type: application/x-www-form-urlencoded` HTTP header and the request should be name, value pairs URL encoded as per RFC 1738.

Complex fields consisting of single or multidimensional records or arrays must be formatted as per the PHP http_build_query() method using square brackets to represent multiple dimensions. The sub-field names must be numeric or alphanumeric only, alphanumeric fields must not start with a numeric. Any square brackets around the nested field names should be URL encoded, [as %5B and] as %5D.

The following example request contains a complex **`items`** field consisting of an array of records representing the following table of purchased items.

Description	Quantity	Amount
Newspaper	1	110
Chocolate bar	3	249
Carrier bag	1	10

For example, a request would be URL encoded as:

```
merchantID=100001&action=SALE&type=1&amount=1001&currencyCode=826&countryCode=826&trans
actionUnique=55f6db1c81d95&orderRef=Test+purchase&customerPostCode=NN17+8YG&responseCod
e=0&responseMessage=AUTHCODE%3A350333&state=captured&xref=15091702MG47WN32MM88LPK&cardN
umber=4929+4212+3460+0821&cardExpiryDate=1215&items%5B0%5D%5Bdescription%5D=Newspaper&i
tems%5B0%5D%5Bquantity%5D=1&items%5B0%5D%5Bamount%5D=110&items%5B1%5D%5Bdescription%5D=
Chocolate+bar&items%5B1%5D%5Bquantity%5D=3&items%5B1%5D%5Bamount%5D=249&items%5B2%5D%5B
description%5D=Carrier+bag&items%5B2%5D%5Bquantity%5D=1&items%5B2%5D%5Bamount%5D=1
```

For more information on field encoding please refer to section 1.7.8.

Please note that the field and sub-field names must be alphanumeric only and are cAsE sEnSiTiVe. Root integration fields must be numeric only and alphanumeric fields must not start with a numeric.

The response will use the same URL encoding and return the request fields in addition to any dedicated response field. If the request contains a field that is also intended as a response field, then any incoming request value will be overwritten by the correct response value.

The Gateway may add new response fields at any time and so your integration must be able to handle and ignore such fields until the time you upgrade your integration to use the new response data.

[1] There is currently a limit of 255 POST parameters in a request, please note that to allow for parameters added in the response you should not use more than 200 parameters in your request.

1.7.2 Hosted HTTP Requests

When using the Hosted Integration, the request must be sent from the Customer's web browser as the response will be a HTML Hosted Payment Page (HPP), used to collect the Customer's details. The format of the request is designed so that it can be sent using a standard HTML form with the data in hidden form fields. The browser will then automatically encode the request correctly according to `application/x-www-form-urlencoded` format

When the Hosted Payment Page has been completed and the payment processed, the Customer's browser will be automatically redirected to the URL provided via the **`redirectURL`** field. The response will be returned to this page in `application/x-www-form-urlencoded` format, using a HTTP POST request.

All request fields will be returned in the response and a Merchant may add custom request fields as detailed in section 16. If the request contains a field that is also intended as a response field, then any incoming request value will be overwritten by the correct response value.

The Gateway may add new request and response fields at any time and so your integration must take care not to send request fields that may conflict with future Gateway fields and be able to ignore response fields which it doesn't yet understand.

An example of a Hosted Integration request is provided in appendix A-21.1 and sample code is provided in appendix A-22.1.

1.7.3 Direct HTTP Requests

When using the Direct Integration, the response will be received in the same URL encoded format, unless a **`redirectURL`** field is provided.

If a **`redirectURL`** field is provided, then the response will be a HTML page designed to redirect a browser to the URL provided, using a HTTP POST request containing the response. This allows you to collect the Cardholder's payment details on your own server, using a HTML form which POSTs to the Direct Integration, which then effectively POSTs the results back to this URL your webserver, where you can display the transaction outcome.

All request fields will be returned in the response and a Merchant may add custom request fields as detailed in section 16. If the request contains a field that is also intended as a response field, then any incoming request value will be overwritten by the correct response value.

The Gateway may add new request and response fields at any time and so your integration must take care not to send request fields that may conflict with future Gateway fields and be able to ignore response fields which it doesn't yet understand.

An example of a Direct Integration request is provided in appendix A-21.2 and sample code is provided in appendix A-22.2.

1.7.4 Batch HTTP Requests

When using the Batch Integration, a single HTTP POST request can contain multiple individual requests using the `multipart/mixed` content type with a boundary string specified. Within that main HTTP request, each of the parts contains a nested Direct Integration HTTP request, separated by the boundary string.

Each part should begin with a `Content-Type: application/x-www-form-urlencoded` HTTP header and contain a single Direct Integration HTTP request, as documented in section 1.7.3.

You can optionally specify a `Content-Id` HTTP header to identify each part message uniquely; if not provided, the Gateway will assign a unique id to each part. The `Content-Id` HTTP header is returned in the response. The Gateway will not validate the uniqueness of any id provided. After the mandatory `Content-type` and the optional *`Content-Id`* header, two carriage return/line feed pairs must follow (ie $\backslash r\backslash n\backslash r\backslash n$). Any deviation from this structure might lead to the part being rejected or incorrectly interpreted. The part request payload, formatted as a regular HTTP URL encoded request, must follow the two line breaks directly.

To reduce the size of large batch requests, the Gateway supports compression using a `Content-Encoding` HTTP header with either a 'gzip' or 'x-gzip' value. This header can be provided in the main request or in the part request or both.

An `Authorization` HTTP header can be used in the request to provide the username and password of a Gateway Merchant Management System user account. If correct, the batch details will be recorded as having been submitted by that user; if invalid, then the request will fail and respond with a 401 (Unauthorised) HTTP status code.

The Gateway will respond in the same manner as the request with a `multipart/mixed` content type; each part is the response to one of the requests in the batched request. In addition, the response will contain a standard `Location` HTTP header, providing a URL from which further batch update responses can be downloaded; and a standard `Content-Disposition` header, allowing a browser to download the response to a file. If the request contained an `Authorization` HTTP header, then the response will contain an `X-P3-Token` HTTP header containing an authentication token that can be sent in future requests instead of the username and password. The authentication token has a limited life span, but each future request will return a new token and thus effectively rejuvenate the token's life.

Like the parts in the request, each response part contains a HTTP response, including headers and body. Each response part is preceded by a `Content-Type` HTTP header and `Content-ID` HTTP header. In addition, an `X-Transaction-ID` HTTP header is added containing the requests transaction id together with an `X-Transaction-Response` HTTP header containing a textual description of the transaction processing status.

The Gateway will not process the transactions immediately but will queue them up to process over time. The transactions may not be processed in the order provided, so should not have interdependencies. Transactions will only appear in the Merchant Management System when they have been processed. The status of queued transaction is only available by querying the status of the batch.

The current status of a batch can be queried at any time by issuing a HTTP GET request to the URL provided in the initial responses `Location` HTTP header.

An `Authorization` HTTP header must be provided in the status request, containing either the username and password of a Gateway Merchant Management System user account or an authentication token returned in the batch submission response's `X-P3-Token` HTTP header. If a valid username and password or a valid token is provided, then the response will be an updated version of the initial submission response providing the current status of each transaction. The response will only contain transactions that the authenticated user has permission to view.

All request fields will be returned in the response and a Merchant may add custom request fields as detailed in section 16. If the request contains a field that is also intended as a response field, then any incoming request value will be overwritten by the correct response value.

The Gateway may add new request and response fields at any time and so your integration must take care not to send request fields that may conflict with future Gateway fields and be able to ignore response fields which it doesn't yet understand.

An example of a Batch Integration request is provided in appendix A-21.3 and sample code is provided in appendix A-22.3.

1.7.5 Handling Errors

When the Gateway is uncontactable due to a communications error, or problem with the internet connection, you may receive a HTTP status code in the 500 to 599 range. In this situation, you may want to retry the transaction. If you do choose to retry a transaction, then we recommend that you perform a limited number of attempts with an increasing delay between each attempt.

If the Gateway is unavailable during a scheduled maintenance period, you will receive a HTTP status code of 503 'Service Temporarily Unavailable'. In this situation, you should retry the transaction after the scheduled maintenance period has expired. You will be notified of the times and durations of any such scheduled maintenance periods in advance, by email, and given a time when transactions can be reattempted.

If you are experiencing these errors, then we recommend you consider the following steps as appropriate for the integration method being used:

- Ensure the request is being sent to HTTPS and not HTTP. HTTP is not supported and is not redirected.
- Send transactions sequentially rather than concurrently.
- Configure your integration code with try/catch loops around individual transactions to determine whether they were successful or not and retry if required, based on the return code or HTTP status returned.
- Configure the integration so that if one transaction fails, the entire batch does not stop at that point – ie log the failure to be checked and then skip to the next transaction rather than stopping entirely.

1.7.6 Redirect URL

The **`redirectURL`** request field is used to provide the URL of a webpage on your server.

For the Hosted Integration, the Customers browser will load this URL when the Hosted Payment Page has completed and can be used to continue the payment journey on your website. The URL will be loaded using a HTTP POST request containing transaction response data allowing you to tailor the journey depending on the outcome of the transaction.

For the Direct Integration, this allows you to collect the Customer's payment details on your own server using a HTML form that you design, but which POSTs directly to the Gateway rather than your own server and thus not exposing any sensitive card data to your server. The Gateway will respond to the HTML form submission with a request to Customer's browser to redirect to this URL allowing you to continue the payment journey on your website. The URL will be loaded using a HTTP GET request containing transaction response data allowing you to tailor the journey depending on the outcome of the transaction. This usage is not recommended as it makes it harder to sign the message.

The URL is mandatory for the Hosted Integration and optional for the Direct Integration. It is not supported by the Batch Integration.

The **`redirectURL`** must be a fully qualified URL, containing at least the scheme and host components.

It is strongly recommended that the response data sent to the `redirectURL` be used to display a payment confirmation page only and not used to update your backend systems. The Customer may close their browser before the redirection happens resulting in you never receiving this data. Please use the `callbackURL` if you need to update your backend systems.

1.7.7 Callback URL

The **`callbackURL`** request field allows you optionally to request that the Gateway sends a copy of the response to an alternative URL. In this case, each response will then be POSTed to this URL in addition to the normal response. This allows you to specify a URL on a secure shopping cart or backend order processing system, which will then fulfil any order associated with the transaction.

The URL is optional for both the Hosted Integration and the Direct Integration. It is not supported by the Batch Integration.

The **`callbackURL`** must be a fully qualified URL, containing at least the scheme and host components.

1.7.8 Field Formats

Most integration field values are either numerical or textual, and either free format or from a range of predetermined values. Some field values are records or arrays of records.

Unless otherwise stated, numerical values are whole integer values with no decimal points. Textual values should use the UTF-8 character set and will be automatically truncated if too long, unless stated otherwise in the field's description. Textual values may be transliterated[2] when sending to third parties such as Acquirers but the original value is stored by Gateway and displayed in the Merchant Management System.

Field values should use the following formats unless otherwise stated in the field's description:

Field Type	Value Format
`Monetary Amounts`	Either major currency units by providing a value that includes a single decimal point such as '10.99'; or in minor currency units by providing a value that contains no decimal points such as '1099'.
`Timestamps`	Date in the format 'YYYY-MM-DD HH:MM:SS'
`Dates`	Date in the format 'YYYY-MM-DD'
`Country Codes`	Either the ISO-3166-1 2-letter, 3-letter or 3-digit code.
`Currency Codes`	Either the ISO-4217 3-letter or 3-digit code.
`Records`	Records can be provided using the *[Y]* notation, where *Y* is the record's sub-field name. Records can be nested to any depth, that is a sub-field's value can be another record. Arrays can be provided by using numeric sub-fields starting with the value 0 and incrementing by 1. For example: to send a value for the sub-field Z, of the sub-field *Y* in the integration field *X,* use the field name *X[Y][Z]*; however, to send a value for the sub-field *Z* in the fourth record for integration field *X,* then use the field name *X[4][Z]* etc. Boolean values must be sent as the words 'true' or 'false'.
`Serialised Records`	Certain fields allow records to be sent as JSON or URL serialised strings. If the first character of the serialised string is '{', then the string is assumed to be in JSON format with any boolean values sent as their JSON equivalents, all other strings will be assumed to be application/x-www-form-urlencoded format with any boolean values sent as the words 'true' or 'false'.

Note: *Record* format is useful when posting sub-fields directly from individual field in a HTML FORM. However, unlike the main integration fields, a record's sub-fields are not sorted when constructing the signature and are processed in the order received. *Serialised record* format can overcome any problems caused by the sub-fields of a *record* being received in a different order to that used when generating the signature. Not all fields using the *record* format also support the *serialised record* format especially the **`threeDSRequest`**, **`threeDSResponse`**, **`checkoutRequest`**, **`checkoutResponse`** and the purchase **`items`** field.

Boolean values cannot be represented when using the *record* format or the application/x-www-form-urlencoded *serialised record* format and the words 'true' and 'false' must be used. The JSON *serialised record* format does not have this restriction and a JSON boolean can be used.

[2] Transliteration involves the changing of character case, stripping of accents from characters and removal of unsupported characters so that the values meet the requirements of the third-party.

1.8 Authentication

All requests must specify which Merchant Account they are for, using the **`merchantID`** request field. In addition to this, the following security measures can be used:

1.8.1 Password Authentication

You can configure a password for each Merchant Account, using the Merchant Management System (MMS). This password must then be sent in the **`merchantPwd`** field in each request. If an incorrect password is received by the Gateway, then the transaction will be aborted, and an error response is returned.

Warning: Use of a password is discouraged in any integration where the transaction is posted from a form in the client browser as the password may appear in plain text in code.

1.8.2 Message signing

You can configure a signing secret phrase for each Merchant Account using the Merchant Management System (MMS). When configured, each request will need to be 'signed' by providing a **`signature`** field containing a hash generated from the combination of the serialised request and this signing secret phrase. On receipt, the Gateway will then re-generate the hash and compare it with the one sent. If the two hashes are different, then the request received must not be the same as that sent and so the contents must have been tampered with and the transaction will be aborted, and an error response is returned.

The Gateway will also return hash of the response message in the returned **`signature`** field, allowing you to create your own hash of the response (minus the **`signature`** field) and verify that the hashes match.

Message signing maybe mandatory on some Merchant Accounts and if so, the Merchant Management System will only allow the secret phrase to be changed but not removed entirely.

If message signing is enabled, then the data POSTed to any callback URL will also be signed.

See appendix A-12 for information on how to create the hash.

1.8.3 Allowed IP addresses

You can configure a list of IP addresses using the Merchant Management System (MMS). Two different address lists can be configured, one for standard requests, such as sales: and one for advanced requests, such as refunds and cancellations. If a request is received from an address other than those configured, then it will be aborted, and an error response is returned.

1.9 Supported Actions

All requests must specify what action they require the Gateway to perform, using the **`action`** request field. The Direct and Batch Integrations support all actions; however, the Hosted Integration only supports the basic payment actions.

1.9.1 SALE

This will create a new transaction and attempt to seek authorisation for a sale from the Acquirer. A successful authorisation will reserve the funds on the Cardholder's account until the transaction is settled.

The **`captureDelay`** field can be used to state whether the transaction should be authorised only and settled at a later date. For more details on delayed capture, refer to appendix A-9.

1.9.2 VERIFY

This will create a new transaction and attempt to verify that the card account exists with the Acquirer. The transaction will result in no transfer of funds and no hold on any funds on the Cardholder's account. It cannot be captured and will not be settled. The transaction **`amount`** must always be zero.

This transaction type is the preferred method for validating that the card account exists and is in good standing; however, it cannot be used to validate that it has sufficient funds.

1.9.3 PREAUTH

This will create a new transaction and attempt to seek authorisation for a sale from the Acquirer. If authorisation is approved, then it is immediately voided (where possible) so that no funds are reserved on the Cardholder's account. The transaction will result in no transfer of funds. It cannot be captured and will not be settled.

This transaction type can be used to check whether funds are available and that the account is valid. However, due to the problem highlighted below, it is recommended that Merchants use the VERIFY action when supported by their Acquirer.

Warning: If the transaction is to be completed then a new authorisation must be sought using the SALE action. If the PREAUTH authorisation could not be successfully voided, then this will result in the funds' being authorised twice effectively putting two holds on the amount on the Cardholder's account and thus requiring twice the amount to be available in the Cardholder's account. It is therefore recommended only to PREAUTH small amounts, such as £1.00 to check mainly account validity.

1.9.4 REFUND_SALE

This will create a new transaction and attempt to seek authorisation for a refund of a previous SALE from the Acquirer. The transaction will then be captured and settled if and when appropriate. It can only be performed on transactions that have been successfully settled. Up until that point, a CANCEL or partial CAPTURE can be used, to refund or partially refund the original SALE transaction. The previous SALE transaction should be specified using the **`xref`** field. This may take up to 180 days from the date of the original sale, however different Card Schemes and Acquirers may set shorter time periods.

Partial refunds are allowed by specifying the **`amount`** to refund. Any amount must not be greater than the original received amount minus any already refunded amount. Multiple partial refunds may be made while there is still a portion of the originally received amount un-refunded.

The **`captureDelay`** field can be used to state whether the transaction should be authorised only and settled at a later date. For more details on delayed capture refer to appendix A-9.

This action is not supported by the Hosted Integration.

1.9.5 REFUND

This will create a new transaction and attempt to seek authorisation for a refund from the Acquirer. The transaction will then be captured and settled if and when appropriate. This is an independent refund and need not be related to any previous SALE. The amount is therefore not limited by any original received amount.

The **`captureDelay`** field can be used to state whether the transaction should be authorised only and settled at a later date. For more details on delayed capture refer to appendix A-9.

This action is not supported by the Hosted Integration.

1.9.6 CAPTURE

This will capture an existing transaction, identified using the **`xref`** request field, making it available for settlement at the next available opportunity. It can only be performed on transactions that have been authorised but not yet captured. An **`amount`** to capture may be specified but must not exceed the original amount authorised. This may take up to 30 days from the date of authorisation, however different Card Schemes and Acquirers may set shorter time periods.

The original transaction must have been submitted with a **`captureDelay`** value that prevented immediate capture and settlement leaving the transaction in an authorised but un-captured state. For more details on delayed capture refer to appendix A-9.

This action is not supported by the Hosted Integration.

1.9.7 CANCEL

This will cancel an existing transaction, identified using the **`xref`** request field, preventing it from being settled. It can only be performed on transactions, which have been authorised but not yet settled, and it is not reversible. Depending on the Acquirer it may not reverse the authorisation and release any reserved funds on the Cardholder's account. In such cases authorisation will be left to expire as normal, releasing the reserved funds. This may take up to 30 days from the date of authorisation, however different Card Schemes and Acquirers may set shorter time periods.

This action is not supported by the Hosted Integration.

1.9.8 QUERY

This will query an existing transaction, identified using the **`xref`** request field, returning the original response. This is a simple transaction lookup action.

This action is not supported by the Hosted Integration.

2 New Transactions

You can perform a new transaction, such as a sale, by sending a request with the required action and transaction type together with details about the order and payment method.

2.1 Request Fields

Field Name	Mandatory?	Description
merchantID	Yes	Your Gateway Merchant Account ID.
merchantPwd	No[1]	Any password used to secure this account. Refer to section 1.8.1 for details.
signature	Yes[2]	Any hash used to sign this request. Refer to section 1.8.2 for details.
action	Yes	The action requested. Refer to section 1.9 for supported actions. Possible values are: **PREAUTH, VERIFY, SALE, REFUND, REFUND_SALE.** If a **REFUND_SALE** action is used, then the request must not attempt to change the payment details, or it will fail with a **responseCode** of **65542** (**REQUEST MISMATCH**) because the refund must be made to the original card.
amount	Yes[3]	The amount of the transaction.
type	Yes[3]	The type of transaction. Refer to appendix A-14 for details. Possible values are: **1** – E-commerce (ECOM) **2** – Mail Order/Telephone Order (MOTO). **9** – Continuous Authority (CA).
countryCode	Yes[3]	Merchant's location.
currencyCode	Yes[3]	Transaction currency.
paymentMethod	No	The payment method required. For card payments either omit this field or use the value **card**.
cardNumber	Yes[3,4]	The primary account number (PAN) as printed on the front of the payment card. Digits and spaces only.
cardExpiryMonth	Yes[3,4]	Payment card's expiry month from 1 to 12.
cardExpiryYear	Yes[3,4]	Payment card's expiry year from 00 to 99.
cardExpiryDate	No[3,4]	Payment card's expiry date in MMYY format as an alternative to sending a separate **cardExpiryMonth** and **cardExpiryYear**.

Field Name	Mandatory?	Description
cardCVV	Yes[3,4]	Payment card's security number. The 3-digit number printed on the signature strip.
transactionUnique	No[3]	You can supply a unique identifier for this transaction. This is an added security feature to combat transaction spoofing.
orderRef	No[3]	Free format text field to store order details, reference numbers, etc. for the Merchant's records.
orderDate	No	Optional date to record with the transaction.
captureDelay	No	Number of days to wait between authorisation of a payment and subsequent settlement. Refer to appendix A-8 for details.
xref	No[5]	Reference to a previous transaction. Refer to appendix A-15 for details.
redirectURL	No[6]	URL to which the hosted form will redirect the Customer's browser after the transaction has been completed. The URL must be fully qualified and include at least the scheme and host components. Refer to section 1.7.6 for details.
callbackURL	No[6]	URL which will receive a copy of the transaction result by POST. The URL must be fully qualified and include at least the scheme and host components. Refer to section 1.7.7 for details.
remoteAddress	No[7]	IP v4 address of client making the transaction using dotted decimal notation (eg. 1.12.123.255). This should be provided where possible to aid fraud prevention.
rtAgreementType	No[8]	Agreement between Merchant and Cardholder for the storage of, or subsequent use of, payment details. Refer to section 12 for further details.

[1] A password is not recommended if using the Hosted Integration, use a signature instead.
[2] A signature maybe mandatory on some Merchant Accounts and requests.
[3] Optional if an **xref** is provided as the value will be taken from the cross-referenced transaction.
[4] Optional if using the Hosted Integration, any value provided will be used to initialise any HPP input field.
[5] Mandatory for a REFUND_SALE request to specify the original SALE transaction.
[6] Not supported by the Batch Integration.
[7] Not supported by the Hosted Integration, which will automatically use the Customer's IP address.
[8] Mandatory for recurring transactions or other transactions using stored credentials.

2.2 Response Fields

The response will contain all the fields sent in the request (minus any **cardNumber** and **cardCVV**) plus the following:

Field Name	Returned?	Description
responseCode	Always	A numeric code providing the specific outcome. Common values are: **0** - Successful / authorised transaction. **1** - Card referred – Refer to card issuer. **2** - Card referred – Special condition. **4** - Card declined – Keep card. **5** - Card declined. Check **responseMessage** for more details of any error that occurred. Refer to appendix A-1 for details.
responseStatus	Always	A numeric code providing the outcome category. Possible values are: **0** – Authorisation Approved / No reason to decline **1** – Authorisation Declined. **2** – Authorisation Error / Transaction malformed.
responseMessage	Always	Message received from the Acquiring bank, or any error message.
transactionID	Always	A unique ID assigned by the Gateway.
xref	Always	You may store the cross reference for repeat transactions. Refer to appendix A-15 for details.
state	Always	Transaction state. Refer to appendix A-13.2 for details.
timestamp	Always	Time the transaction was created or last modified.
transactionUnique	If supplied	Any value supplied in the initial request.
authorisationCode	On success	Authorisation code received from Acquirer.
referralPhone	If provided	Telephone number supplied by Acquirer to phone for voice authorisation when provided.
amountReceived	On success	Amount the Acquirer authorised. This should always be the full **amount** requested.
amountRefunded	If refund	Total amount of original SALE that has so far been refunded. Returned when **action** is REFUND_SALE.
orderRef	If supplied	Any value supplied in the initial request.
cardNumberMask	Always	Card number masked for Merchant storage.

Field Name	Returned?	Description
cardTypeCode	Always	Code identifying the type of card used. Refer to appendix A-10 for details.
cardType	Always	Description of the type of card used. Refer to appendix A-10 for details.
cardSchemeCode	Always	Code identifying the Card Scheme used. Refer to appendix A-10 for details.
cardScheme	Always	Description of the Card Scheme used. Refer to appendix A-10 for details.
cardIssuer	Always	Card Issues name (when known).
cardIssuerCountry	Always	Card issuing country's name (when known).
cardIssuerCountryCode	Always	Card issuing country's ISO-3166 2-letter code (when known).

Other response fields may be returned as documented elsewhere in this guide. Undocumented fields may be returned at the Gateways discretion but should not be relied upon.

The **acquirerResponse*XXXX*** fields are dependent on the Acquirer in use and are supplied for additional information only.

The response is also POSTed to any URL provided by optional **callbackURL**.

3 Management Requests

You can perform a management action on an existing transaction, such as a capture or cancellation, by sending a request with the required action together with the cross reference for the transaction to act on.

Management requests are supported by the Direct and Batch Integrations, they are not supported by the Hosted Integration.

3.1 Request Fields

Field Name	Mandatory?	Description
merchantID	Yes	Your Gateway Merchant Account ID.
merchantPwd	No[1]	Any password used to secure this account. Refer to section 1.8.1 for details.
signature	Yes[2]	Any hash used to sign this request. Refer to section 1.8.2 for details.
action	Yes	The action requested. Refer to section 1.9 for supported actions. Possible values are: **CAPTURE, CANCEL, QUERY**.
xref	Yes	Reference to a previous transaction. Refer to appendix A-15 for details.
amount	No[3]	The amount to capture or refund.
callbackURL	No	A non-public URL which will receive a copy of the transaction result by POST. The URL must be fully qualified and include at least the scheme and host components. Refer to section 1.7.7 for details.

[1] A password is not recommended if using the Hosted Integration, use a signature instead.
[2] A signature maybe mandatory on some Merchant Accounts and requests.
[3] An amount is only required for partial refunds or partial captures.

3.2 Response Fields

Apart from the fields below, the response will be the same as for a new transaction but will contain the details of the existing transaction.

Field Name	Returned?	Description
responseCode	Always	A numeric code providing the outcome of the management request. Check **responseMessage** for more details of any error that occurred. Refer to appendix A-1 for details.
responseStatus	Always	A numeric code providing the outcome category. Possible values are: **0** – Authorisation Approved / No reason to decline **1** – Authorisation Declined. **2** – Authorisation Error / Transaction malformed.
responseMessage	Always	Description of above response code.
action	Always	The requested action and original action separated by a colon. For example. **CANCEL:SALE**

Other response fields may be returned as documented elsewhere in this guide. Undocumented fields may be returned at the Gateways discretion but should not be relied upon.

The response is also POSTed to any URL provided by optional **callbackURL**.
.

4 AVS/CV2 Checking

4.1 Background

AVS and CV2 fraud checking is available on all card transactions processed by the Gateway where supported by the Acquirer.

These fraud prevention checks are performed by the Acquirer while authorising the transaction. You can choose how to act on the outcome of the check (or even to ignore them altogether).

4.1.1 AVS Checking

The Address Verification System (AVS) uses the address details that are provided by the Cardholder to verify that the address is registered to the card being used. The address and postcode are checked separately.

4.1.2 CV2 Checking

CV2, CVV, or Card Verification Value is a 3-digit or 4-digit security code. The check verifies that the code is the correct one for the card used.

For most cards, the CVV is a 3-digit number to the right of the signature strip. For American Express cards, this is a 4-digit number printed, not embossed, on the front right of the card.

The AVS/CV2 checking preferences can be configured per Merchant Account within the Merchant Management System (MMS). These preferences can be overridden per transaction by sending one of the preference fields documented in section 4.3 that hold a comma separated list of the check responses that should be allowed in order to continue to completion. Responses not in the list will result in the transaction being declined with a **`responseCode`** of **5 (AVS/CV2 DECLINED)**.

AVS/CV2 fraud checking is not available with all Acquirers and must be enabled on your Merchant Account before it can be used. Please contact support to find out whether your Acquirer supports it and if it can be enabled on your Merchant Account.

4.2 Benefits and Limitations

4.2.1 Benefits

- Can be enabled with just a few extra integration fields.
- The results are available immediately and returned as part of the transaction.
- The checks can be managed independently, allowing you the utmost control over how the results are used.
- The checks can be configured to decline a transaction automatically, where required.
- There are no extra costs for using AVS/CV2 checking with your transactions.
- Fully configurable within the Merchant Management System (MMS).

4.2.2 Limitations

- The AVS checks are mainly supported by Visa, MasterCard and American Express in the USA, Canada and United Kingdom. Cardholders with a bank that does not support the checks might receive declines due to the lack of data.
- Because AVS only verifies the numeric portion of the address and postcode, certain anomalies such as apartment numbers and house names can cause false declines.
- The checks are meant for consumer cards. Company cards are not fully supported due to the Acquirers' not having access to this information.

4.3 Request Fields

These fields should be sent in addition to basic request fields in section 2.1.

Field Name	Mandatory?	Description
customerAddress	Yes[1]	For AVS checking, this must be a registered billing address for the card.
customerPostCode	Yes[2]	For AVS checking, this must be a registered postcode for the card.
cardCVV	Yes[3]	For CVV checking, this must be the Card Verification Value printed on the card.
avscv2CheckRequired	No[4]	Is AVS/CV2 checking required for this transaction? Possible values are: **N** – Checking is not required. **Y** – Abort if checking is not enabled.
cv2CheckPref	No[4]	List of **cv2Check** response values that are to be accepted; any other value will cause the transaction to be declined. Refer to appendix A-2 for details. Value is a comma separated list containing one or more of the following**: not known, not checked, matched, not matched, partially matched**.
addressCheckPref	No[4]	List of **addressCheck** values that are to be accepted; any other value will cause the transaction to be declined. Refer to appendix A-2 for details. Value is a comma separated list containing one or more of the following: **not known, not checked, matched, not matched, partially matched**.
postcodeCheckPref	No[4]	List of **postcodeCheck** response values that are to be accepted; any other value will cause the transaction to be declined. Refer to appendix A-2 for details. Value is a comma separated list containing one or more of the following: **not known, not checked, matched, not matched, partially matched**.

[1] Mandatory if AVS address checking is required.
[2] Mandatory if AVS postcode checking is required.
[3] Mandatory if CV2 checking is required.
[4] Overrides any Merchant Account setting configured via the Merchant Management System (MMS).

4.4 Response Fields

These fields will be returned in addition to the AVS/CV2 request fields in section 4.3 and the basic response fields in section 2.2.

Field Name	Returned?	Description
avscv2CheckEnabled	Always	Is AVS/CV2 checking enabled for this Merchant Account? Possible values are: **N** – Merchant account is not enabled. **Y** – Merchant account is enabled.
avscv2ResponseCode	If checks performed	The result of the AVS/CV2 check. Refer to appendix A-2 for details.
avscv2ResponseMessage	If checks performed	The message received from the Acquiring bank, or any error message with regards to the AVS/CV2 check. Refer to appendix A-2 for details.
avscv2AuthEntity	If checks performed	Textual description of the AVS/CV2 authorising entity as described in appendix A-2. Possible values are: **not known, merchant host, acquirer host, card scheme, issuer.**
cv2Check	If checks performed	Description of the AVS/CV2 check as described in appendix A-2. Possible values are**: not known, not checked, matched, not matched, partially matched**.
addressCheck	If checks performed	Description of the AVS/CV2 address check as described in appendix A-2. Possible values are**: not known, not checked, matched, not matched, partially matched**.
postcodeCheck	If checks performed	Description of the AVS/CV2 postcode check as described in appendix A-2. Possible values are**: not known, not checked, matched, not matched, partially matched**.

5 3-D Secure Authentication

5.1 Background

3-D Secure (3DS) authentication is an additional fraud prevention scheme that is on all e-commerce card transactions processed by the Gateway, where supported by the Acquirer.

It allows the Cardholder to assign a password to their card that is then verified whenever a transaction is processed through a site that supports the use of the scheme. The addition of password protection allows extra security on transactions that are processed online.

3-D Secure stands for Three Domain Secure. There are 3 parties that are involved in the 3-D Secure process:

- The company from which the purchase is being made.
- The Acquiring Bank (the bank of the company)
- VISA and Mastercard (the card issuers themselves)

The Gateway supports EMV 3-D Secure as implemented by Visa, Mastercard and American Express and marketed under the brand names of Visa Secure, Mastercard ID Check and American Express SafeKey. Implementations by JCB (J/Secure) and DCI (ProtectBuy) are not currently supported.

The 3-D Secure preferences can be configured per Merchant Account within the Merchant Management System (MMS). These preferences can be overridden per transaction by sending one of the preference fields documented in section 5.5.1, which hold a comma separated list of the check responses that should be allowed to continue to completion. Responses not in the list will result in the transaction being declined with a **`responseCode`** of **65803 (3DS_NOT_AUTHENTICATED)**.

3-D Secure is not available with all Acquirers and must be enabled on your Merchant Account before it can be used. Please contact support to find out whether your Acquirer supports it and if it can be enabled on your Merchant Account.

3-D Secure is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

5.2 Benefits and Limitations

5.2.1 Benefits

- The results are available immediately and returned as part of the transaction.
- The checks can be managed independently allowing you the utmost control over how the results are used.
- The checks can be configured to decline the transaction automatically, where required.
- There are no extra Gateway costs for using 3-D Secure. Your Acquirer may charge to add this onto your business account; however, you may also find that your transaction charges are lower as a result of using 3-D Secure.
- Fully configurable within the Merchant Management System (MMS).

5.2.2 Limitations

- The Gateway does not support 3-D Secure for JCB or Diner's club cards.
- 3-D Secure transactions require a browser in order to display the Customer authentication dialog.

5.3 Hosted Implementation

If your Merchant Account is set up for 3-D Secure, the Hosted Payment Page will automatically attempt to display the 3-D Secure authentication page for the Customer's bank.

The 3-D Secure authentication form is designed and controlled by the Customer's Issuing bank and will display the Merchant Account name and any website address added when the account was onboarded. You can change the displayed name and website address by sending the **`merchantName`** and/or **`merchantWebsite`** request fields. Any **`merchantWebsite`** must be a fully qualified URL containing at least the scheme and host components.

You may also pass additional information about the transaction and Cardholder, using the **`threeDSOptions`** field as documented in section 5.5.4. This extra information can help the Issuer decide on whether a challenge is required

Your Merchant Account must be configured with your Merchant Category Code (MCC). This value can be configured in the Merchant Management System (MMS) or provided using the **`merchantCategoryCode`** or **`threeDSOptions`** request fields.

If you would like an example of a 3-D Secure Hosted Integration, please refer to our sample code appendix A-22.1**.**

5.4 Direct Implementation

If your Merchant account is set up for 3-D Secure, the Gateway will require further authentication details provided by the 3-D Secure system.

The 3-D Secure authentication form is designed and controlled by the Customer's Issuing bank and will display the Merchant Account name and any website address added when the account was onboarded. You can change the displayed name and website address by sending the **`merchantName`** and/or **`merchantWebsite`** request fields. Any **`merchantWebsite`** must be a fully qualified URL containing at least the scheme and host components.

You may also pass additional information about the transaction and Cardholder, using the **`threeDSOptions`** field as documented in section 5.5.4. This extra information can help the Issuer decide on whether a challenge is required

Your Merchant Account must be configured with your Merchant Category Code (MCC). This value can be configured in the Merchant Management System (MMS) or provided using the **`merchantCategoryCode`** or **`threeDSOptions`** request fields.

You will need to implement a callback page on your web server which the ACS can redirect the Cardholder's browser to on completion of any challenges. You will need to provide the address of this page to the Gateway in your initial payment request via the **`threeDSRedirectURL`** field.

The direct integration uses two complex fields to pass data between the 3-D Secure Access Control Server (ACS) and the Gateway. The **`threeDSRequest`** is a record whose name/value properties must be sent via a HTTP POST request to the ACS. The corresponding **`threeDSResponse`** field should be returned to the Gateway and must be a record containing name/value properties taken from the HTTP POST received from the ACS when it redirects the Cardholder's browser back to your callback page on completion of any challenge.

Note that the contents of the **`threeDSRequest`** and **`threeDSResponse`** fields are formatted using the *record* format detailed in section 1.7.8 and their contents should be regarded as opaque and all name/value pairs received from the Gateway must be sent to the ACS, and vice versa. The Gateway does not currently support these fields to be provided in the serialised record format.

5.4.1 Request Flow

Each step of the 3-D Secure flow is described below. At a high level it consists of a secure conversation between the Customer's issuing bank and the Acquiring bank, facilitated by both the Merchant and the Gateway, in the form of several challenges. Each step of this conversation may contain a visible or invisible challenge. A 3-D Secure authentication form represents a visible challenge. An invisible challenge may be presented in the form of a Device Fingerprinting Method as described in section 5.4.6. The bank controls the number of challenges each transaction requires, and some transactions may be sufficiently trusted to require zero challenges. Because of this, your implementation will need to loop through multiple possible Continuation Requests, as described in section 5.4.3, thus continuing the 3-D Secure process until you have received a message from the Gateway informing you that the 3-D Secure process has been completed.

1. You should send an initial request, as described in section 5.4.2, to the Gateway containing the payment details, device details and any required **`threeDSOptions.`** *This request must include your callback page, as described above, in the* ***`threeDSRedirectURL`*** *field.*

2. If you receive a **`responseCode`** of **65802** then go to step 3 else display the results of the finished transaction as indicated by the **`responseCode`** and **`responseStatus`** as normal.

3. You must store the received **`threeDSRef`** value as it will be needed by the continuation request in step 6.

4. You should then send the contents of the received **`threeDSRequest`** to the 3-D Secure Access Control Server (ACS) at the received **`threeDSURL`** as described in section 5.4.2.

5. Once the ACS has completed the challenge then it will return the outcome back to your callback page as originally provided in step 1, as described in section 5.4.3.

6. You must then send a continuation request, as described in section 5.4.3, to the Gateway containing the **`threeDSRef`** as stored in step 3 and a **`threeDSResponse`** containing the data received from the ACS in step 5.

7. As there can be multiple challenges you must now repeat the sequence from step 2.

If you would like an example of a 3-D Secure Direct Integration, please refer to our sample code appendix A-22.2.

5.4.2 Initial Request (Verify Enrolment)

If no 3-D Secure authentication details are provided in the initial request, the Gateway will determine if the transaction is eligible for 3-D Secure by checking whether the card is enrolled in the 3-D Secure scheme.

If the Gateway determines that the transaction is not eligible for 3-D Secure, then it will continue to process it as a normal transaction without 3-D Secure, unless the **threeDSRequired** request field indicates that the transaction should be aborted instead.

To support 3-D Secure, you must pass the **threeDSRedirectURL** field in the initial request. This field must contain the complete URL to a web page on your server that the 3-D Secure Access Control Server (ACS) will HTTP POST the authentication results back to, when the authentication has been completed.

You must also provide details about the Cardholder's device, using the fields documented in section 17.7 or using the associated options in the **threeDSOptions** field as documented in section 5.5.4. You may also use the **threeDSOptions** field to pass additional information about the transaction and Cardholder, which can help the Issuer decide on whether a challenge is required.

If the Gateway determines that the transaction is eligible, it will respond with a **responseCode** of **65802 (3DS AUTHENTICATION REQUIRED)** and included in the response will be a **threeDSRef** field, a **threeDSReqest** field and a **threeDSURL** field.

The **threeDSRequest** field is a record whose name/value properties must be sent, using a HTTP POST request, to the 3-D Secure Access Control Server (ACS) at the URL provided by the **threeDSURL** field. This is usually achieved via means of a hidden HTML input fields in a form rendered within an IFRAME displayed on the Cardholder's browser and then submitted using JavaScript. The IFRAME must be of sufficient size to display the challenge screen, however, if the **threeDSRequest** contains a **threeDSMethodData** component, then the challenge is invisible, and a small hidden IFRAME can be used instead.

You must store the value of the **threeDSRef** field for use in the continuation request.

5.4.3 Continuation Request (Check Authentication and Authorise)

On completion of the 3-D Secure authentication the ACS will send the challenge results to you callback page, as originally specified using the **threeDSRedirectURL** field in the initial request.

The data will be received via means of a HTTP POST request and the contents of this POST request should be returned to the Gateway unmodified as name/value properties in the **threeDSResponse** field together with the **threeDSRef** received in the initial response.

This new request will check the authentication results and either respond with the details for a further challenge, send the transaction to the Acquirer for approval, or abort the transaction, depending on the authentication result and your preferences, either sent in the **threeDSCheckPref** field or set in the Merchant Management System (MMS).

5.4.4 Multiple Challenges and Frictionless Flow

The API supports the issuing of multiple challenges where the continuation request may indicate the requirement to perform another challenge by responding with a **`responseCode`** of **65802 (3DS AUTHENTICATION REQUIRED)** and including a further **`threeDSReqest`**, **`threeDSURL`** and **`threeDSRef`**. When this happens, these further challenge details should be treated the same as the first and POSTed to the ACS.

An initial device fingerprinting method might have to be invoked on the ACS, the results of which are used to determine whether the Cardholder must complete a challenge or whether a frictionless flow can be achieved where the transaction can continue unchallenged.

5.4.5 Cardholder Challenge

The Cardholder challenge takes place with the Cardholder's browser, usually within an IFRAME embedded on the payment form. To start the challenge, the IFRAME should contain a HTML FORM with hidden INPUT fields storing the name/value pairs returned in the **`threeDSRequest`** record. JavaScript should then be used to submit the form automatically, causing the form data to be sent via a HTTP POST to the **`threeDSURL`**.

The IFRAME should be of sufficient size to display the ACS challenge form. The challenge form supports a limited number of different sizes, giving the Merchant more flexibility in the design of their payment form. The required size can be set using the 'challengeWindowSize' option, passed in the **`threeDSOptions`** field in the initial request.

5.4.6 Device Fingerprinting Challenge

The device fingerprinting method invocation is handled in the same way as a normal Cardholder challenge, except that it can be done silently in a hidden IFRAME, invisible to the normal payment flow. This silent device fingerprinting method request can be determined by the presence of a **`threeDSMethodData`** element in the **`threeDSRequest`** record (this is one time when the normally opaque data does need to be checked).

This method should take no longer than 10 seconds and therefore if the ACS has not POSTed the results back within 10 seconds, then the browser can stop waiting and the transaction can be continued as normal but the **`threeDSResponse`** field should be returned indicating the timeout by including a **`threeDSMethodData`** element with the value of 'timeout', for example, **`"threeDSResponse[threeDSMethodData]=timeout"`**

5.4.7 External Authentication Request

You can choose to obtain the 3-D Secure authentication details from a third-party, in which case you should provide them as part of a standard request. If the Gateway receives valid third-party authentication details, then it will use those and not attempt to perform the 3-D Secure challenge flow.

5.5 Request Fields

5.5.1 Initial Request (Hosted and Direct Integration)

These fields should be sent in addition to basic request fields in section 2.1.

Field Name	Mandatory?	Description
merchantName	No[1]	Merchant name to use on 3DS challenge form.
merchantWebsite	No[1]	Merchant website to use on 3DS challenge form. The website must be a fully qualified URL and include at least the scheme and host components.
merchantCategoryCode	No[1]	Merchant category code.
threeDSRequired	No[1]	Is 3DS required for this transaction? Possible values are: **N** – 3DS is not required. **Y** – Abort if 3DS is not enabled.
threeDSCheckPref	No[1]	List of **threeDSCheck** response values that are to be accepted, any other value will cause the transaction to be declined. Value is a comma separated list containing one or more of the following values: **'not known', 'not checked', ' not authenticated', 'attempted authentication', 'authenticated'**.
threeDSRedirectURL	Yes[2]	A URL on the Merchant's server to which the ACS can POST the challenge results, thus redirecting the challenge IFRAME to this page.
threeDSOptions	No[3]	Record containing 3DS options that can be used by the ACS for advance fraud checking. Refer to section 5.5.4 for further details.
threeDSPolicy	No[1]	3DS Policy used. Refer to appendix A-18.4 for further details.
threeDSVersion	No[1]	Force a particular version to be used rather than using the highest available for the transaction details. **Use of this request field is not encouraged under normal circumstances.**
scaExemption	No[1]	An SCA exemption can be used to request that a frictionless flow is preferable. Refer to appendix A-18.3 for further details.

[1] Overrides any Merchant Account setting configured via the Merchant Management System (MMS).
[2] Not required for Hosted Integration
[3] Some browser configuration options are mandatory if the corresponding device detail fields are not provided.

5.5.2 External Authentication Request (Direct Integration)

These fields should be sent in addition to basic request fields from section 2.1.

Field Name	Mandatory?	Description
threeDSEnrolled	If 3DS enabled	The 3DS enrolment status for the credit card. Refer to appendix A-3 for details. Possible values are: **Y** – Enrolled. **N** – Not enrolled. **U** – Unable to verify if enrolled. **E** – Error verifying enrolment.
threeDSAuthenticated	If 3DS enrolled	The 3DS authentication status for the credit card. Refer to appendix A-3 for details. Possible values are: **Y** – Authenticated. **A** – Attempted to authenticate. **N** – Not authenticated. **I** – Information only. **U** – Unable to authenticate. **E** – Error checking authentication.
threeDSXID	If 3DS authenticated	The unique identifier for the transaction in the 3DS system. Refer to appendix A-3 for details.
threeDSECI	If 3DS authenticated	The Electronic Commerce Indicator (ECI). Refer to appendix A-3 for details.
threeDSCAVV	If 3DS authenticated	The Cardholder Authentication Verification Value (CAVV). Refer to appendix A-3 for details.

Note: If 3-D Secure is not enabled for the Merchant Account, then any 3-D Secure authentication fields sent in the request are ignored and the transaction is processed as normal without 3-D Secure.

When an external 3-D Secure provider is used then you are responsible for deciding whether to continue at the different 3-D Secure stages and any preferences provided in the MMS or using the **threeDSCheckPref** request field are ignored.

If the external provider returns an authentication status of ‘R’ then you must not continue with the transaction either with or without 3-D Secure. Do not attempt to send a **threeDSAuthentication** status of ‘R’ expecting the Gateway to reject the transaction.

5.5.3 Continuation Request (Direct Integration)

These fields may be sent alone[1].

Field Name	Mandatory?	Description
threeDSRef	Yes	The value of the **threeDSRef** field in the initial Gateway response.
threeDSResponse	Yes	The data POSTed back from the ACS when the challenge has completed.

[1] Note: It is only necessary to send the **threeDSRef** and the **threeDSResponse** in the continuation request, because the **threeDSRef** will identify the Merchant Account and initial request. The message does not need to be signed. However, you can send any of the normal request fields to modify or supplement the initial request. Any card details and transaction amount sent in the second request must match those used in the first request, else the second request will fail with a **responseCode** of **64442 (REQUEST MISMATCH)**.

5.5.4 3-D Secure Options (Hosted and Direct Integration)

The following options may be sent in the **`threeDSOptions`** field to provide additional information to help customise the 3-D Secure experience or to help the ACS decide if a challenge is required.

Some additional information will be automatically provided by the Gateway from standard integration fields unless overridden by providing the associated option. The standard integration field associated with each option is shown in brackets below the options field name. The standard integration field should be used rather than the option, apart from the very rare circumstances where the two must have different values.

A few additional information values, such as the Cardholder's browser details[1], are mandatory and therefore either the standard integration field or the option must be provided. These fields are marked as 'Yes' in the Mandatory column of the table below.

Further details what information is mandatory can be found at the end of this section.

The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Option Name	Mandatory?	Description
`accountAgeIndicator`	No	Length of time that the cardholder has had the account with the 3DS Requestor. Possible values are: **01** – No account (guest check-out) **02** – Created during this transaction **03** – Less than 30 days **04** – 30-60 days **05** – More than 60 days
`accountChangeDate`	No	Date that the cardholder's account with the 3DS Requestor was last changed. Accepted date format is YYYYMMDD.
`accountChangeIndicator`	No	Length of time since the cardholder's account information with the 3DS Requestor was last changed. Possible values are: **01** – Changed during this transaction **02** – Less than 30 days **03** – 30-60 days **04** – More than 60 days
`accountDate`	No	Date that the cardholder opened the account with the 3DS Requestor. Accepted date format is YYYYMMDD.
`accountDayTransactions`	No	Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous 24 hours.

[1] The Hosted Integrations Payment Page will ignore any browser options and use its own details.

Option Name	Mandatory?	Description
accountId	No	Identifier for any Cardholder's account with the Merchant.
accountPasswordChangeDate	No	Date that cardholder's account with the 3DS Requestor had a password change or account reset. Accepted date format is YYYYMMDD.
accountPasswordChangeIndicator	No	Indicates the length of time since the cardholder's account with the 3DS Requestor had a password change or account reset. Possible values are: **01** – No change **02** – Changed during this transaction **03** – Less than 30 days **04** – 30-60 days **05** – More than 60 days
accountProvisioningAttempts	No	Number of account provisioning attempts in the last 24 hours.
accountPurchaseCount	No	Number of purchases with this cardholder account during the previous six months.
accountType	No	Indicates the type of account. Possible values are: **01** – Not applicable **02** – Credit **03** – Debit
accountYearTransactions	No	Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous year.
acquirerBIN	No	Acquirer bank identification number. (Default on file)
acquirerCountryCode	No	Acquirer country code when the Acquirer country differs from the Merchant country and the Acquirer is in the EEA (this could mean that the transaction is covered by PSD2). (Default on file)
acquirerMerchantID	No	Acquirer-assigned merchant identifier. (Default on file)
acsChallengeMandatedIndicator	No	Indication of whether a challenge is required for the transaction to be authorized due to local/regional mandates or other variables.
addressMatch	No	Indicates to the ACS whether the cardholder shipping address and billing address are the same. Possible values are: **Y** – Shipping address matches billing address. **N** – Shipping address does not match billing address.

Option Name	Mandatory?	Description
authenticationIndicator	No	Provides additional information to the ACS to determine the best approach for handing an authentication request. Possible values are: **01** – Payment - default **02** – Recurring **03** – Instalment **04** – Add Card **05** – Maintain Card **06** – Verify Cardholder **07** – Billing Agreement
billingAddressCity (**customerTown**)	No	The city of the address. Maximum length is 50 characters.
billingAddressCountryCode (**customerCountryCode**)	No	The country of the address as an ISO 3166-1 3-digit code or its 2 or 3 letter equivalents.
billingAddressLine1 (**customerAddress**)	No	The first line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
billingAddressLine2	No	The second line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
billingAddressLine3	No	The third line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
billingAddressPostcode (**customerPostcode**)	No	The ZIP or other postal code of the address. Maximum length is 16 characters.
billingAddressState	No	The state or province of the address as an ISO 3166-2 country subdivision code. Maximum length is 3 characters.
browserAcceptHeader (**deviceAcceptContent**)	Yes[1]	HTTP accept header sent from the Cardholder's browser.
browserIPAddress (**remoteAddress**)	Yes[1]	IP address of the Cardholder's browser. This can be an IP v4 address using dotted decimal notation (eg. 1.12.123.255) or an IP v6 address using colon hexadecimal notation (eg. 2011:0db8:85a3:0101: 0101:8a2e:0370:7334)
browserJavaEnabled (**deviceCapabilities**)	Yes[1]	Ability of the Cardholder's browser to execute Java. Possible values are[2]: **true** – browser can execute Java **false** – browser cannot execute Java

Option Name	Mandatory?	Description
browserJavaScriptEnabled (**deviceCapabilities**)	Yes[1]	Ability of the Cardholder's browser to execute JavaScript. Possible values are[2]: **true** – browser can execute JavaScript **false** – browser cannot execute JavaScript
browserLanguage (**deviceAcceptLanguage**)	Yes[1]	The Cardholder's browser language. The browser language as defined in IETF BCP 47.
browserScreenColourDepth or **browserScreenColorDepth** (**deviceScreenResolution**)	Yes[1,3]	The screen colour depth of the Cardholder's browser. A value representing the bit depth of the colour palette, in bits per pixel, for displaying images. Possible values are: **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48**.
browserScreenHeight (**deviceScreenResolution**)	Yes[1,3]	The screen height of the Cardholder's browser.
browserScreenWidth (**deviceScreenResolution**)	Yes[1,3]	The screen width of the Cardholder's browser.
browserTimeZone (**deviceTimeZone**)	Yes[1]	The difference between UTC time and the cardholder's browser local time in minutes.
browserUserAgent (**deviceIdentity**)	Yes[1]	The User-Agent header provided by the Cardholder's browser.
cardholderEmail (**customerEmail**)	No	The Cardholder's email address.
cardholderHomePhone (**customerPhone**)	No	The Cardholder's home phone number specified in the following format: "+CountryCode Subscriber" (eg "+44 1234567899").
cardholderMobilePhone (**customerMobile**)	No	The Cardholder's mobile phone number specified in the following format: "+CountryCode Subscriber" (eg "+44 1234567899").
cardholderName (**customerName**)	No	Name of the Cardholder. Limited to the alphanumeric characters listed in EMV Book 4, Annex B.
cardholderWorkPhone	No	The Cardholder's work phone number specified in the following format: "+CountryCode Subscriber" (eg "+44 1234567899").

Option Name	Mandatory?	Description
challengeWindowSize	No	The dimensions of the challenge window that has been displayed to the cardholder. The ACS shall reply with content that is formatted to appropriately render in this window to provide the best possible user experience. Possible values are: **01** – 250 x 400 **02** – 390 x 400 (default) **03** – 500 x 600 **04** – 600 x 400 **05** – Full screen
deliveryEmailAddress (**deliveryEmail**)	No	Email Address to which the merchandise was delivered for electronic delivery.
deliveryTimeframe	No	Merchandise Delivery Timeframe. Possible values are: **01** – Electronic Delivery **02** – Same day shipping **03** – Overnight shipping **04** – Two-day or more shipping
giftCardAmount	No	For prepaid or gift card purchase, the purchase amount total of prepaid or gift card(s) in major units (for example, USD 123.45 is 123).
giftCardCount	No	For prepaid or gift card purchase, total count of individual prepaid or gift cards/codes purchased.
giftCardCurrencyCode	No	For prepaid or gift card purchase, the currency code of the card as an ISO 4217 3-digit code or its 3 letter equivalent.
instalmentPaymentData or **installmentPaymentData**	No	Indicates maximum number of authorizations permitted for instalment payments. Value must be greater than 1.
merchantCategoryCode (**merchantCategoryCode**)	No	Code describing the Merchant's type of business, product, or service.
merchantCountryCode (**countryCode**)	No	Country code of the merchant as an ISO 3166-1 3-digit code or its 2 or 3 letter equivalents.

Option Name	Mandatory?	Description
merchantFraudRate	No	Merchant's fraud rate in the EEA (all EEA card fraud divided by all EEA card volumes) calculated as per PSD2 RTS. This value is sent to Mastercard only who will not calculate or validate the fraud score. Value will be a numeric value, between 1 and 99, representing the fraud rate, such as: 1 – less than or equal to 1 basis point (bp), which is 0.01% 2 – between 1 bp + - and 6 bps 3 – between 6 bps + - and 13 bps 4 – between 13 bps + - and 25 bps 5 – greater than 25 bps
merchantName (**merchantName**)	No	Merchant name.
paymentAccountAge	No	Date that the payment account was enrolled in the cardholder's account with the 3DS Requestor. Accepted date format is YYYYMMDD.
paymentAccountAgeIndicator	No	Indicates the length of time that the payment account was enrolled in the cardholder's account with the 3DS Requestor. Possible values are: **01** – No account (guest check-out) **02** – Created during this transaction **03** – Less than 30 days **04** – 30-60 days **05** – More than 60 days
preOrderDate	No	For a pre-ordered purchase, the expected date that the merchandise will be available. Accepted date format is YYYYMMDD.
preOrderPurchaseIndicator	No	Indicates whether Cardholder is placing an order for merchandise with a future availability or release date. Possible values are: **01** – Merchandise available **02** – Future availability
priorAuthData	No	Data that documents and supports a specific authentication process.

Option Name	Mandatory?	Description
priorAuthMethod	No	Mechanism used by the Cardholder to previously authenticate to the 3DS Requestor. Part of the optional 3DS Requestor Prior Transaction Authentication Information that contains information about a 3DS cardholder authentication that occurred prior to the current transaction. Possible values are: **01** – Frictionless authentication occurred by ACS **02** – Cardholder challenge occurred by ACS **03** – ACS verified **04** – Other issuer methods
priorAuthTimestamp	No	Date and time in UTC of the prior cardholder authentication. Accepted date format is YYYYMMDDHHMM.
priorReference	No	Provides additional information to the ACS to determine the best approach for handling a request. It contains an ACS Transaction ID for a prior authenticated transaction (for example, the first recurring transaction that was authenticated with the cardholder).
recurringExpDate	No	This field contains the date after which no further authorizations shall be performed. The format of this field must be: YYYYMMDD
recurringFrequency	No	This field indicates the minimum number of days between authorizations.
reorderItemsIndicator	No	Indicates whether the cardholder is reordering previously purchased merchandise. Possible values are: **01** – First time ordered **02** – Reordered
reqAuthData	No	Data that documents and supports a specific authentication process. In the current version of the specification, this data element is not defined in detail, however the intention is that for each 3DS Requestor Authentication Method, this field carry data that the ACS can use to verify the authentication process.

Option Name	Mandatory?	Description
reqAuthMethod	No	Method used by the Cardholder to authenticate to the 3DS Requestor. Part of the 3DS Requestor Authentication Information which contains optional information about how the cardholder authenticated during login to their 3DS Requestor account. Possible values are: **01** – No 3DS Requestor authentication occurred (ie cardholder "logged in" as guest) **02** – Login to the cardholder account at the 3DS Requestor system using 3DS Requestor's own credentials **03** – Login to the cardholder account at the 3DS Requestor system using federated ID **04** – Login to the cardholder account at the 3DS Requestor system using issuer credentials **05** – Login to the cardholder account at the 3DS Requestor system using third-party authentication **06** – Login to the cardholder account at the 3DS Requestor system using FIDO Authenticator
reqAuthTimestamp	No	Date and time in UTC of the cardholder authentication. Accepted date format is YYYYMMDDHHMM.
requestorChallengeIndicator	No	Indicates whether a challenge is requested for this transaction. For example: you may be using this to obtain SCA and not want to be challenged as you have provided an exemption. This also allows you to request a challenge when adding a card to a wallet or to follow mandates. Possible values are: **01** – No preference **02** – No challenge requested **03** – Challenge requested: 3DS Requestor Preference **04** – Challenge requested: Mandate **05** – No challenge requested (TRA is already performed) **06** – No challenge requested (data share only) **07** – No challenge requested (SCA is already performed) **08** – No challenge requested (utilize whitelist exemption) **09** – Challenge requested (whitelist prompt requested) Values 05 to 09 are for protocol version 2.2.0 but will be accepted for 2.1.0 and 05 to 08 will be downgraded to 02, and 09 to 01. Values 05 to 07 will be passed to Mastercard as part of their 'Merchant Data' extension. Values 08 and 09 are only valid for protocol version 2.2.0.

Option Name	Mandatory?	Description
requestorID	No	Directory server assigned 3DS Requestor identifier. (Default on file)
requestorName	No	Directory server assigned 3DS Requestor name. *(Default on file)*
requestorURL (**merchantWebsite**)	No	3DS Requestor website or customer care site.
secureCorporatePaymentIndicator	No	Indicates dedicated payment processes and procedures were used, potential secure corporate payment exemption applies.
serverOperatorID	No	3DS Server identifier. *(Default on file)*
serverRefNumber	No	Assigned server reference number. *(Default on file)*
shipAddressUsageDate	No	Date when the shipping address used for this transaction was first used with the 3DS Requestor. Accepted date format is YYYYMMDD.
shipAddressUsageIndicator	No	Indicates the length of time since the shipping address used for this transaction was first used with the 3DS Requestor. Possible values are: **01** – This transaction **02** – Less than 30 days **03** – 30-60 days **04** – More than 60 days
shipIndicator	No	Indicates shipping method chosen for the transaction. Merchants must choose the Shipping Indicator code that most accurately describes the cardholder's specific transaction, not their general business. If one or more items are included in the sale, the Shipping Indicator code for the physical goods is used, or if all digital goods, the Shipping Indicator code that describes the most expensive item. Possible values are: **01** – Ship to cardholder's billing address **02** – Ship to another verified address on file with merchant **03** – Ship to address that is different than the cardholder's billing address **04** – "Ship to Store" / Pick-up at local store (Store address shall be populated in shipping address fields) **05** – Digital goods (includes online services, electronic gift cards and redemption codes) **06** – Travel and Event tickets, not shipped **07** – Other (for example, Gaming, digital services not shipped, emedia subscriptions, etc.)

Option Name	Mandatory?	Description
shipNameIndicator	No	Indicates if the Cardholder Name on the account is identical to the shipping Name used for this transaction. Possible values are: **01** – Account Name identical to shipping Name **02** – Account Name different than shipping Name
shippingAddressCity (**deliveryTown**)	No	The city of the address. The maximum length is 50 characters.
shippingAddressCountryCode (**deliveryCountryCode**)	No	The country of the address as an ISO 3166-1 3-digit code or its 2 or 3 letter equivalents.
shippingAddressLine1 (**deliveryAddress**)	No	The first line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
shippingAddressLine2	No	The second line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
shippingAddressLine3	No	The third line of the street address or equivalent local portion of the address. Maximum length is 50 characters.
shippingAddressPostcode (**deliveryPostcode**)	No	The ZIP or other postal code of the address. Maximum length is 16 characters.
shippingAddressState	No	The state or province of the address as an ISO 3166-2 country subdivision code. Maximum length is 3 characters.
suspiciousAccountActivity	No	Indicates whether the 3DS Requestor has experienced suspicious activity (including previous fraud) on the cardholder account. Possible values are: **01** – No suspicious activity has been observed **02** – Suspicious activity has been observed
transactionType	No	Identifies the type of transaction being authenticated. This field is required in some markets (eg for Merchants in Brazil). Possible values are: **01** – Goods/ Service Purchase **03** – Check Acceptance **10** – Account Funding **11** – Quasi-Cash Transaction **28** – Prepaid Activation and Load

Option Name	Mandatory?	Description
whitelistStatus	No	Whitelist status. Possible values are: **Y** – 3DS Requestor is whitelisted by cardholder **N** – 3DS Requestor is not whitelisted by cardholder

[1] The Hosted Integrations Payment Page will ignore any browser options and use its own details.
[2] Boolean values must be sent as the strings 'true' or 'false' unless JSON serialisation is used (refer to section 1.7.8)
[3] This value is not mandatory if the browser doesn't support JavaScript.

5.5.5 Mandatory 3-D Secure Information

Some 3-D Secure information is mandatory and must be provided using the following t**hreeDSOptions** field or preferably using the associated standard integration field, in brackets, apart from the very rare circumstances where the two must have different values.

If this mandatory information is not available, then 3-D Secure will fail.

The following information is mandatory:

- **requestorName** (**merchantName**) *default stored on file*
- **requestorURL** (**merchantWebsite**) default *stored on file*
- **merchantCategoryCode** (**merchantCategoryCode**) *default stored on file*
- **browserIPAddress** (**remoteAddress**)
- **browserUserAgent** (**deviceIdentity**)
- **browserAcceptHeader (deviceAcceptContent**)
- **browserLanguage** (**deviceAcceptLanguage**)

If the **browserJavaScriptEnabled** (**deviceCapabilites**) field is provided and indicates that JavaScript is enabled on the Cardholder's browser, then the following information is also mandatory:

- **browserScreenColourDepth/browserScreenColorDepth** (**deviceScreenResolution**)
- **browserScreenWidth** (**deviceScreenResolution**)
- **browserScreenHeight** (**deviceScreenResolution**)
- **browserTimeZone** (**deviceTimeZone**)

5.6 Response Fields

5.6.1 Challenge Response (Direct Integration)

These fields are returned when a 3-D Secure challenge is required and provide the information necessary for you to request the Access Control Server (ACS) perform that challenge.

These fields will be returned in addition to the request fields from section 5.5.1 and the basic response fields in section 2.2.

Field Name	Returned?	Description
threeDSEnabled	Always	Is 3DS enabled for this Merchant Account? Possible values are: **N** – Merchant Account is not enabled. **Y** – Merchant Account is enabled.
threeDSPolicy	If available	The unique identifier for the transaction in the 3DS system.
threeDSVETimestamp	If 3DS enabled	The time the card was checked for 3DS enrolment, and any initial challenge determined.
threeDSEnrolled	If 3DS enabled	The 3DS enrolment status for the credit card. Refer to appendix A-3 for details. Possible values are: **Y** – Enrolled. **N** – Not enrolled. **U** – Unable to verify if enrolled. **E** – Error verifying enrolment.
threeDSRef	If 3DS enabled	Value to return in the continuation request.
threeDSURL	If 3DS enrolled	The URL of the ACS to which the challenge data should be sent via a HTTP POST request from the Cardholder's browser.
threeDSRequest	If 3DS enrolled	Record containing the name/value pairs that should be sent to the ACS via HTTP POST request from the Cardholder's browser.

5.6.2 Final Response (Hosted and Direct Integration)

These fields are returned when the 3-D Secure stage has completed, and no further challenge is required.

These fields will be returned in addition to the request fields from section 5.5.1; any continuation request fields from section 5.5.3; any challenge response fields from section 5.6.1; and the basic response fields in section 2.2.

Field Name	Returned?	Description
threeDSEnabled	Always	Is 3DS enabled for this Merchant Account? Possible values are: **N** – Merchant Account is not enabled. **Y** – Merchant Account is enabled.
threeDSPolicy	If available	3DS Policy used. Refer to appendix A-18.4 for further details.
threeDSXID	If 3DS enabled	The unique identifier for the transaction in the 3DS system. Refer to appendix A-3 for details.
threeDSVETimestamp	If 3DS enabled	The time the card was checked for 3DS enrolment, and any initial challenge determined.
threeDSEnrolled	If 3DS enabled	The 3DS enrolment status for the credit card. Refer to appendix A-3 for details. Possible values are: **Y** – Enrolled. **N** – Not enrolled. **U** – Unable to verify if enrolled. **E** – Error verifying enrolment.
threeDSCATimestamp	If 3DS enrolled	The time the last challenge was checked.
threeDSAuthenticated	If 3DS enrolled	The 3DS authentication status for the credit card. Refer to appendix A-3 for details. Possible values are: **Y** – Authenticated. **A** – Attempted to authenticate. **N** – Not authenticated. **R** – Reject transaction. **I** – Information only. **U** – Unable to authenticate. **E** – Error checking authentication.
threeDSECI	If 3DS authenticated	This contains a two-digit Electronic Commerce Indicator (ECI) value. Refer to appendix A-3 for details. The data contained within this property is only valid if the **threeDSAuthenticated** value is **Y** or **A**.

Field Name	Returned?	Description
threeDSCAVV	If 3DS authenticated	This contains a 28-character Base-64 encoded Cardholder Authentication Verification Value (CAVV). Refer to appendix A-3 for details. The data contained within this property is only valid if the **threeDSAuthenticated** value is **Y** or **A**.
threeDSDetails	If 3DS authenticated	Record containing further details about the 3-D Secure processing stage. Notable sub fields are: • version – 3-D Secure version used • versions – 3-D Secure versions available • psd2Region – whether payment in PSD2 jurisdiction
threeDSErrorCode	If 3DS error	Any error response code returned by the ACS if there is an error in determining the card's 3DS status.
threeDSErrorDescription	If 3DS error	Any error response description returned by the ACS if there is an error in determining the card's 3DS status.
threeDSResponseCode	Always	A numeric code providing the specific outcome of the 3-D Secure processing stage. Check **threeDSResponseMessage** for more details of any error that occurred. Refer to appendix A-1 for details.
threeDSResponseMessage	Always	Any error message relating to the outcome of the 3-D Secure processing stage.

5.6.3 External Authentication Response (Direct Integration)

These fields will be returned in addition to the request fields from section 5.5.2 and the basic response fields in section 2.2.

Field Name	Returned?	Description
threeDSEnabled	Always	Is 3DS enabled for this Merchant Account? Possible values are: **N** – Merchant Account is not enabled. **Y** – Merchant Account is enabled.

Note: If 3-D Secure is not enabled for the Merchant Account, then any 3-D Secure authentication fields sent in the request are ignored and the transaction is processed as normal without 3-D Secure.

5.6.4 Cardholder Information (Hosted and Direct Integration)

In the case of a frictionless flow, the card Issuer may sometimes wish to provide a message to the Cardholder. In this case, the **threeDSResponseMessage** will start with the text 'Cardholder Info: ' and be followed by the message from the card Issuer.

5.7 Advanced Features

5.7.1 PSD2 Strong Customer Authentication

3-D Secure can be used to provide the Strong Customer Authentication (SCA) required by the European Union's Payment Services Directive 2 (PSD2).

For more details on how to use 3-D Secure to maintain PSD2 SCA Compliance please refer Appendix A-18.4.

6 Risk Checking

6.1 Background

The Gateway is integrated with Kount, the leading solution for digital fraud prevention.

If you have an existing account with Kount, or sign up for one, you can request that the Gateway pass your transactions to them for risk checking before they are sent to the Acquirer for authorisation.

Kount's patented fraud prevention technology combines device fingerprinting; supervised and unsupervised machine learning; a robust policy and rules engine; business intelligence tools; and a web-based case-management and investigation system.

Their team of experts can help you understand and identify the rules necessary to optimise your protection, as well as provide ongoing support. To get the most out of your investment, you may want to dedicate an individual or a team to monitor your rules and ensure they continue to work as intended.

The risk checking preferences can be configured per Merchant Account within the Merchant Management System (MMS). These preferences can be overridden per transaction by sending new preferences as documented in section 6.4.1. You must use the Kount management portal to configure your risk parameters and thresholds.

Risk checking is an advanced feature and must be enabled on your Merchant Account before it can be used. Please contact support if you wish to have it enabled.

6.2 Benefits and Limitations

6.2.1 Benefits

- The results are available immediately and returned as part of the transaction.
- The checks can be managed independently, allowing you the utmost control over how the results are used.
- The checks can be configured to decline the transaction automatically where required.
- Leverage the ability to review transactions and decide what course of action to take.
- The checks can reduce chargebacks by blocking transactions made without the Cardholder's consent that would have resulted in the Cardholder raising a chargeback to recover the fraudulent transaction amount.
- Providing enhanced risk checking increases Customer confidence and thus increases the likelihood of their making a purchase.
- Fully configurable within the Merchant Management System (MMS).

6.2.2 Limitations

- Checking cannot prevent all fraudulent transactions and could even prevent some non-fraudulent transactions.
- There are additional fees associated with having a Kount account.
- You will have to spent time analysing your transactions and establishing fraud rules.

6.3 Implementation

When risk checking is required, each transaction will be sent to Kount for checking and the result of the check will be returned in the **`riskCheck`** response field with one of the following values:

- **not known** - the checks could not be performed due to our error
- **not checked** - the checks could not be performed by Kount
- **approve** - the transaction is not risky and should proceed
- **decline** - the transaction is risky and should be declined
- **review** - the transaction is risky but proceed with caution
- **escalate** - the transaction is risky but proceed with caution

The actions to take for each **`riskCheck`** response can be configured for the Merchant Account, using the Merchant Management System. Alternatively, the preferred actions can be passed with the transaction request in the **`riskCheckPref`** field. The possible actions are as follows:

- **continue** - continue processing as normal
- **authonly** - authorise only, don't capture
- **decline1** - decline without reason
- **decline2** - decline with reason
- **finished** - abort with reason

The **continue** action allows the transaction to continue as normal and be sent to the Acquirer for authorisation. *A **`riskCheck`** value of **approve** will always be treated as if the action was **continue,** regardless of whether the preferences say otherwise.*

The **authonly** action allows the transaction to be authorised but not automatically captured giving you time to review it and decide whether you want to take the risk and capture the transaction or assume it to be fraudulent and cancel it.

The **decline1** and **decline2** actions will cause the transaction to be declined. Both decline the transaction and return with a **responseCode** of **5** (**DECLINED**) and a **`responseMessage`** of 'DECLINED' or 'RISK DECLINED' respectively. The first action should be used if you don't wish to alert the Customer to the fact that you suspected that their transaction was fraudulent and declined it for that reason.

The finished action will abort the transaction, causing it to return with a **responseCode** of either **65857** (**RISK_CHECK_ERROR**) or **65862** (**RISK_CHECK_DECLINED**) depending on whether an error prevented the transaction from being checked by Kount, resulting in a **`riskCheck`** value of 'not known' or 'not checked'.

The **`riskCheckPref`** field can be provided in the request to override any settings configured in the Merchant Management System (MMS) for this Merchant Account. The value should be a comma separated list of *result=actions* pairs. If a result is not specified in the list, then an action of **decline1** is assumed. For example: "**decline=decline1,review=authonly,escalate=authonly**".

6.4 Request Fields

6.4.1 Request Fields

These fields should be sent in addition to basic request fields in section 2.1.

Field Name	Mandatory?	Description
riskCheckRequired	No[1]	Is risk checking required for this transaction? Possible values are: **N** – risk checking is not required. **Y** – risk checking is required.
riskCheckPref	No[1]	List of **riskCheck** response values and the action to be taken for those responses. Value is a comma separated list containing one or more of the following risk check results and associated actions: Results: **not known**, **not checked**, **approve, decline, review, escalate**. Actions: **continue**, **decline1**, **decline2**, **authonly**, **finished**.
riskCheckOptions	No	Record containing options used to customise the risk checking. Refer to section 6.4.2 for values.

[1] Overrides any Merchant Account setting configured via the Merchant Management System (MMS).

6.4.2 Risk Check Options

The following options may be sent in the **`riskCheckOptions`** field to customise the risk checking. Where possible, the options will be initialised from standard integration fields as shown in brackets below the option name. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Option Name	Description
`IPAD` (`remoteAddress`)	IP v4 address of client making the transaction using dotted decimal notation (eg. 1.12.123.255).
`MACK`	Merchant's acknowledgement to ship/process the order (**Y** or **N**).
`SESS`	Unique Session ID[1]. Used to link to Kount's browser device data collector.
`ANID`	Automatic Number Identification (ANI) submitted with order.
`CASH`	Total cash amount in currency submitted.
`ORDR` (`merchantOrderRef`)	Merchant's Order Number.
`UNIQ` (`merchantCustomerRef`)	Merchant assigned account number for Customer.
`EPOC`	Date Customer account was created by merchant.
`NAME` (`customerName`)	Customer's name (or name submitted with the order).
`GENDER` (`customerGender`)	Customer's gender (**M** or **F**)
`BPREMISE` (`customerCompany`)	Customer's billing address premises name (UK only).
`BSTREET` (`customerStreet`, `customerAddress`)	Customer's billing address street (UK only).
`B2A1` (`customerAddress`)	Customer's billing address county/state.
`B2A2` (`customerAddress2`)	Customer's billing address county/state.
`B2CI` (`customerTown`)	Customer's billing address county/state.
`B2ST` (`customerCounty`)	Customer's billing address county/state.
`B2PC` (`customerPostcode`)	Customer's billing address postcode.
`B2CC` (`customerCountryCode`)	Customer's billing address country code.

Option Name	Description
EMAL (**customerEmail**)	Cardholder's email address.
B2PN (**customerPhone**)	Cardholder's phone number.
DOB (**customerDateOfBirth**, **recipientDateOfBirth**)	Cardholder's date of birth.
S2NM (**deliveryName**)	Name of person receiving the delivery.
SPREMISE (**deliveryCompany**)	Delivery premises name (UK only).
SSTREET (**deliveryStreet**, **deliveryAddress**)	Delivery street address (UK only).
S2A1 (**deliveryAddress**)	Delivery address line 1.
S2A2 (**deliveryAddress2**)	Deliver address line 2.
S2CI (**deliveryTown**)	Delivery town/city.
S2ST (**deliveryCounty**)	Delivery county/state.
S2PC (**deliveryPostcode**)	Delivery postcode.
S2CC (**deliveryCountryCode**)	Delivery country code.
S2EM (**deliveryEmail**)	Delivery email address.
S2PN (**deliveryPhone**)	Phone number of delivery location.
SHTP (**shippingType**, **shippingMethod**)	Shipping type.
PROD_TYPE[*XX*] (**items.XX.description**)	Type for the *XX*th item purchased.
PROD_ITEM[*XX*] (**items.XX.productCode**)	SKU for the *XX*th item purchased.
PROD_DESC[*XX*] (**items.XX.description**)	Description *XX*th item purchased.

Option Name	Description
PROD_QUANT[*XX*] (**items.XX.quantity**)	Quantity of *XX*th item purchased.
PROD_PRICE[*XX*] (**items.XX.amount**)	Unit amount for *XX*th item purchased.
UDF[*XXXX*]	User defined field *XXXX*.

[1] SESS, the unique session id, is automatically sent from the Kount Device Collector loaded in the Hosted Payment Page.

For further information on the options, refer to the Kount Integration documentation: https://kount.github.io/docs/ris-data-submission/.

6.5 Response Fields

These fields will be returned in addition to the risk check request fields in section 6.4 and the basic response fields in section 2.2.

Field Name	Returned?	Description
riskCheckEnabled	Always	Is risk checking enabled for this Merchant Account? Possible values are: **N** – Merchant account is not enabled. **Y** – Merchant account is enabled.
riskCheck	If checked	The result of the risk check. Possible values are: **approve** – ok, recommend proceed to authorisation. **decline** – probably fraudulent, recommend decline. **review** – possibly fraudulent, recommend review. **escalate** – possibly fraudulent, recommend review.
riskCheckDetails	If checked	Record containing the raw response received from Kount minus any sensitive data.
riskCheckResponseCode	If checked	Response code for the risk processing stage.
riskCheckResponseMessage	If checked	Response message for the risk processing stage.

7 Payment Facilitators

7.1 Background

If you are a Payment Facilitator (PayFac/PF) or Independent Sales Organisation (ISO), then you must send additional fields to identify yourself and your sub-merchants.

These fields must be sent with every new transaction; however, they can be cloned from an existing transaction if using an **`xref`** as described in appendix A-16.

Payment Facilitator support is not available with all Acquirers. Please contact support to find out if your Acquirer supports it and what fields are required.

7.2 Request Fields

Field Name	Mandatory?	Description
facilitatorID	Yes	Your facilitator identifier as assigned by the Scheme.
facilitatorName	No[1]	Your trading name as registered with the Scheme.
isoID	No[1]	Your ISO identifier as assigned by the Scheme.
subMerchantID	No[1]	Unique identifier assigned to this Sub Merchant.
merchant*XXXX*	No[1]	Sub Merchant details as documented in section 17.2.
statementNarrative*X*	No[1]	Statement details as documented in section 9.1.2.

[1] Which additional fields are mandatory will depend on your Acquirer.

8 UK MCC 6012 Merchants

8.1 Background

Every Merchant Account has a category code, also known as the MCC code, attached to it. This category code identifies the market that the payment is related to, allowing issuing banks to identify what product or service is, or was, being provided.

The merchant category code 6012 is related to payments taken for financial institutions, primarily those merchants that deal with loan payments or other credit-related activities. According to Visa, this is the most fraudulent merchant category in the UK market due to compromised debit card details' being used to pay or transfer balances to other cards. Acquirers are therefore unable to confirm whether a payment is genuine, despite matching the full CVV2 with AVS.

To address this situation, issuing banks have requested additional payment information to be provided with payment requests in order to verify that the cardholder is knowingly entering into a credit-related contractual agreement with the merchant.

If you are a Merchant who has been assigned the MCC 6012 you must collect the following data for the primary recipient for each UK domestic Visa or Mastercard transaction[1].

- Unique account identifier for the loan or outstanding balance funded. For example, the loan account number or the PAN (Primary Account Number) if it is a credit card balance.
- Last name (family name)
- Date of Birth (D.O.B)
- Postcode

Primary recipients are the entities (people or organisations) that have a direct relationship with the financial institution. Also, these primary recipients have agreed to the terms and conditions of the financial institution.

The new fields are not currently mandatory. However, some Acquirers are now declining transactions that are missing this information and so we recommend the information is always provided, even if your Acquirer doesn't currently mandate them.

If you are not a UK MCC 6012 Merchant or the payment is not a UK domestic one, then you need not provide these additional authentication details though the Gateway will accept them if you do.

[1] The additional details are currently only required by Visa and Mastercard however we recommend sending for all card types in order to be prepared for when other card Schemes follow suite.

8.2 Request Fields

To comply with the rules, an MCC6012 Merchant must send these additional fields:

Field Name	Mandatory?	Description
merchantCategoryCode	Yes[1]	Merchant's VISA MCC (should be 6012).
receiverName	Yes	Surname only - up to 6 letters allowed.
receiverAccountNo	Yes	Account number. If a PAN is supplied only the first 6 and last 4 digits will be used.
receiverDateOfBirth	Yes	Primary recipient's date of birth.
receiverPostcode	Yes	Primary recipient's postcode. (Only the district is required but full postcodes are accepted, therefore 'W12 8QT' or just 'W12' are acceptable values).

[1] Only required if the Merchants Category Code is not configured on your Gateway account.

9 Billing Descriptor

9.1 Background

The Billing Descriptor is how your details appear on the Cardholder's statement. It is set up with the Acquirer when the Merchant Account is opened. It is used by the Cardholder to identify who a payment was made to on a particular transaction.

Selecting a clear Billing Descriptor is important for you to avoid a chargeback when the Cardholder does not recognise the name on the transaction.

9.1.1 Static Descriptor

The Static Descriptor is the descriptor agreed between yourself and your Acquirer when the Merchant Account is opened. The descriptor used is typically your trading name, location and contact phone number.

9.1.2 Dynamic Descriptor

The Dynamic Descriptor is a descriptor sent with the transaction that includes details on the goods purchased or service provided, this is often used by large companies that provide many services and where the brand of the service is more familiar than the company name. The Dynamic Descriptor usually replaces any Static Descriptor on a per transaction basis.

Not all Acquirers accept Dynamic Descriptors and, for those that do, the required format varies. Often, your Merchant name is shortened to three (3) letters, followed by an asterisk (*), followed by a short description of the service or product that the business provides. This field typically has a limit of twenty-five (25) characters including the phone number.

For more information on whether your Acquirer allows Dynamic Descriptor and the format in which they should be sent, please contact customer support.

9.2 Request Fields

The Dynamic Descriptor is built using one or more of the following narrative fields.

Field Name	Mandatory?	Description
`statementNarrative1`	No	Merchant's name.
`statementNarrative2`	No	Product, service or other descriptive info.

10 Surcharges

10.1 Background

Surcharges are an additional charge that you can apply to the transactions that are processed through your Merchant Account.

Transactions that are sent for authorisation are subject to processing charges from your Acquirer and surcharges enable you to pass the processing charges that you incur on to your Customers.

You may, for example, be charged a fixed amount for debit card transactions and a percentage amount for credit card transactions. Consequently, the Gateway gives you the option to add both a fixed amount and percentage amount when applying a surcharge.

Surcharges should only be added to cover the processing charges that are incurred by your business. There is no Gateway imposed limit to the value of the surcharges that can be added to your transaction, although there are legal requirements. As a rule, the surcharge must not exceed the processing costs that you pay.

Some businesses apply surcharges to cover all the costs that they incur; while others use the surcharges to subsidise the charges.

Surcharge amounts may be limited or illegal in your jurisdiction. For example, surcharging is illegal in the European Union and many US states. It is up to you to check with your Acquirer and comply with any laws.

Surcharges is an advanced feature and must be enabled on your Merchant Account before it can be used. Please contact support if you wish to have it enabled.

10.2 Implementation

10.2.1 Surcharge Rules

The **`surchargeRules`** field allows you to provide multiple rules specifying what surcharges should be applied to a transaction. If a transaction matches multiple rules, then the most specific rule will be used; or the first in the list.

Each surcharge rule contains the following fields:

Field Name	Mandatory?	Description
`cardType`	Yes	One or more 2-letter card type codes for which this rule applies (see) The following two card type codes are also supported, in addition to the codes listed in appendix A-10: **CC** – matches any credit card. **DD** – matches any debit card.
`currency`	No	Zero or more 3-letter ISO-4217 currency codes.
`surcharge`	Yes	Surcharge amount in minor (N) or major (N.N) units or a percentage (N%).

The surcharge rules should be passed in a sequential array of records, either as nested records or serialised records as described in section 1.7.8. The record field names are case sensitive.

10.2.2 Surcharge Amounts

The Gateway doesn't usually validate that any **`amount`** and **`grossAmount`** fields are the same and that any **`netAmount`**, **`taxAmount`** and **`taxRate`** tally. However, in order to update them when a surcharge is applied, the amount and **`grossAmount`** must match and the correct **`taxRate`** must be provided or be able to be calculated from one or more of the other fields. Failure in this respect can cause the Gateway to return one of the following **`responseCode`** values; **66360** (**INVALID_GROSSAMOUNT**), **66361** (**INVALID_NETAMOUNT**), **66338** (**INVALID_TAXAMOUNT**), **66362** (**INVALID_TAX_RATE**).

If the request contains a **`surchargeAmount`** field, then the Gateway will assume that surcharging has already been performed externally and will not attempt to apply any further surcharges.

10.3 Request Fields

Field Name	Mandatory?	Description
surchargeRequired	No[1]	Is surcharging required for this transaction? Possible values are: **N** – Surcharging is not required. **Y** – Surcharging is required.
surchargeRules	No[1]	Surcharge rules as documented in section 10.2.1.
surchargeAmount	No	Surcharge amount already added. A further surcharge will not be added.

[1] Overrides any Merchant Account setting configured via the Merchant Management System (MMS).

10.4 Response Fields

These fields will be returned in addition to the Surcharge request fields in section 10.3 and the basic response fields in section 2.2.

Field Name	Returned?	Description
surchargeEnabled	Always	Is surcharging enabled on this Merchant Account?
surchargeAmount	Always	Surcharge amount added.
amount	Always	Original request value with additional surcharge.
grossAmount	Conditional	Original request value adjusted for new **amount**.
netAmount	Conditional	Original request value adjusted for new **amount**.
taxAmount	Conditional	Original request value adjusted for new **amount**.

11 Receipts and Notifications

11.1 Background

The Gateway can be configured to email transaction receipts automatically to the Customer and notifications to the Merchant.

11.2 Customer Email Receipts

The Customer can be emailed a transaction receipt automatically each time a transaction is processed by the Gateway. Receipts are sent at the time the transaction is authorised and only for transactions where the Acquirer has approved the authorisation. Receipts are not sent for declined or referred authorisations or aborted transactions.

This functionality is enabled globally on a per Merchant Account basis using the Merchant Management System (MMS). This global setting can also be overridden per transaction if required, using the **`customerReceiptsRequired`** field.

Customer receipts require the Customer to provide an email address; if no email address is provided, using the Hosted Payment Page or the **`customerEmail`** field, then no receipt will be sent.

11.2.1 Merchant Email Notifications

You can be automatically emailed a transaction notification each time a transaction is processed by the Gateway. Notifications are sent at the time the transaction is authorised and only for transactions where the Acquirer approved, declined or referred the authorisation. Notifications are not sent for aborted transactions.

This functionality is enabled globally on a per Merchant Account basis, using the Merchant Management System (MMS). This global setting can also be overridden per transaction if required, using the **`notifyEmailRequired`** field.

Merchant notifications require you to provide an email address; if no email address is provided, using the Merchant Management System (MMS) or the **`notifyEmail`** field, then no notification will be sent.

11.3 Request Fields

Field Name	Mandatory?	Description
`customerReceiptsRequired`	No[1]	Send a Customer receipt if possible. Possible values are: **N** – Don't send a receipt. **Y** – Send if Customer's email provided.
`customerEmail`	No	Customer's email address.
`notifyEmailRequired`	No[1]	Send a notification email if possible. Possible values are: **N** – Don't send a notification. **Y** – Send if notification email provided.
`notifyEmail`	No[1]	Merchant's notification email address.

[1] Overrides any Merchant Account setting configured via the Merchant Management System (MMS).

11.4 Response Fields

The request fields for the required receipts and notifications are returned together with the appropriate fields from the following:

Field Name	Returned?	Description
`customerReceiptsResponseCode`	If required	Result of sending email to Customer. Refer to appendix A-1 for details.
`customerReceiptsResponseMessage`	If required	Description of above response code.
`notifyEmailResponseCode`	If required	Result of sending email to Merchant. Refer to appendix A-1 for details.
`notifyEmailResponseMessage`	If required	Description of above response code.

12 Credentials on File

12.1 Background

You may need to repeat a payment or perform a new payment using payment details previously requested from the Customer and stored by either yourself or the Gateway. These payment details could be stored in previous transaction or in the Gateway wallet. These repeat payments could be one off payments, scheduled recurring payments, or repeats due to authorisation problems or industry requirements. All must be correctly flagged to allow the payment network to process them.

The stored details, or credentials, are termed Credentials on File (COF) and refer to any payment details (including, but not limited to, an account number or payment token) that the Consumer has authorised you to store to perform future transactions without the need for them to enter their payment details again.

These transactions must always be identified with the reason for storing or using the stored credentials and who initiated the transaction – the Consumer (CIT) or the Merchant (MIT).
You may store the credentials and send them with the future transaction, or you may store the details in the Gateway's Wallet as described in section 19 or by taking advantage of the Payment Tokenisation feature of the Gateway as described in appendix A-15[1]. Either way you must tell the Gateway of your intentions, it will not assume that just because you have asked, for example, to store credentials in the Wallet that those are legitimate stored credentials and follow all the requirements laid out below.

If you store credentials on file, then you must:

- Disclose to consumers how those credentials will be used.
- Obtain consumers' consent to store the credentials.
- Notify consumers when any changes are made to the terms of use.
- Inform the card issuer via a transaction that payment credentials are now stored on file.
- Identify transactions with appropriate **`rtAgreementType`** when using stored credentials.
- Perform a PREAUTH, SALE or VERIFY transaction during the initial credential setup.

Note: Credentials stored to complete a single transaction (or a single purchase) for a Consumer, including multiple authorisations related to that particular transaction or future refunds are not considered stored credentials and can be stored and used without the following the above rules.

Note: A new recurring transaction will be a clone of the cross-referenced transaction, including any stored credentials details except for any new data provided in the new transaction. If the new transaction provided different payment details, then any stored credentials in the cross-referenced transaction cannot be used. The **`cloneFields`** request field can also be used to control which fields in the cross-referenced transaction are used in the repeat transaction (refer to appendix A-16).

[1] Card numbers will be stored as part of any transaction details for a period of 13 months or less at your request. Once the card number has been removed a transaction **`xref`** can no longer be used to provide the original card number. If you wish to store credentials for longer then you may store them in the Gateway's Wallet.

12.2 Consumer Initiated Transactions (CIT)

Consumer Initiated Transactions (CIT) are any transaction where the Consumer is actively participating in the transaction. This can be either through a checkout experience online, via a mail order or telephone order, with or without the use of an existing stored credential.

A Consumer Initiated Transaction is one whose **`action`** field is one of **PREAUTH**, **SALE** or **VERIFY** and whose **`type`** is one of **1** (ECOM) or **2** (MOTO).

To indicate that the card details are to be stored as, or were stored as, Credentials on File then send the **`rtAgreementType`** field as one of the following values:

- **cardonfile** – card details stored as Credential on File
- **recurring** – initial payment as the start of a recurring payment agreement.
- **instalment** – initial payment as the start of an instalment payment agreement.

If the card details are cloned from an existing transaction or loaded from a Gateway Wallet which also stored the Credentials on File, then the transaction will be flagged as subsequent use of stored credentials rather than first use of them[1].

If the transaction is the first in a recurring or instalment sequence then the optional **`rtSequenceCount`** field can be used to specify how many transactions will be taken in total, with a value greater than 1, and any optional **`rtSequenceNumber`** field specifying the which transaction it is in the sequence will be expected to have a value of 0.

[1] For flagging of subsequent use, the existing credentials will usually need to have been stored with the same Acquirer.

12.3 Merchant Initiated Transactions (MIT)

Merchant Initiated Transactions (MIT) are any transaction where you have performed the transaction without the active participation of the Consumer. This would normally always be as a follow-up to a previous Consumer Initiated Transaction (CIT).

Refer to section 13 for information on how the Gateway can be instructed to take Merchant Initiated recurring transactions automatically, according to a pre-determined schedule.

Merchant Initiated Transactions are broken down in to two categories as follows.

12.3.1 Standing Instruction MITs

Merchant Initiated Transactions defined under this category are performed to address pre-agreed standing instructions from the Consumer for the provision of goods or services.

The following transaction types are standing instructions transactions:

- Instalment Payments: A transaction in a series of transactions that use a stored credential and that represent Consumer agreement for the merchant to initiate one or more future transactions over a period for a single purchase of goods or services. An example of such a transaction is a hire purchase repayment.

- Recurring Payments: A transaction in a series of transactions that use a stored credential and that are processed at fixed, regular intervals (not to exceed one year between transactions), representing Consumer agreement for the merchant to initiate future transactions for the purchase of goods or services provided at regular intervals. An example of such a transaction is a gym membership subscription.

- Unscheduled Credential on File (UCOF): A transaction using a stored credential for a fixed or variable amount that does not occur on a scheduled or regularly occurring transaction date, where the Consumer has provided consent for the merchant to initiate one or more future transactions. An example of such a transaction is an account auto-top up transaction.

The Gateway classifies the first two types of instalment and recurring payments as Continuous Authority (CA) payments and the third type as a normal Cardholder not present payments. Different Acquirers may use different classifications, but the Gateway will handle this and send the classification expected by the Acquirer.

The maximum period between each transaction is 13 months, however individual Card Schemes or Acquirers may impose shorter periods.

12.3.2 Industry-Specific Business Practice MIT

Merchant Initiated Transactions defined under this category are performed to fulfil a business practice as a follow-up to an original Consumer-Merchant interaction that could not be completed with one single transaction. Not every industry practice Merchant Initiated Transaction requires a stored credential, for example, if you store card details for a single transaction or a single purchase, it is not considered as a stored credential transaction.

The following transaction types are industry specific transactions[1]:

- Incremental[2]: Incremental authorizations can be used to increase the total amount authorised if the authorised amount is insufficient. An incremental authorization request may also be based on a revised estimate of what the Consumer may spend.

- Resubmission: You can perform a resubmission in cases where it requested an authorization but received a decline due to insufficient funds; however, the goods or services were already delivered to the Consumer. In such scenarios, you can resubmit the request to recover outstanding debt from Consumers.

- Reauthorization: You can initiate a reauthorization when the completion or fulfilment of the original order or service extends beyond the authorization validity limit set by the Card Scheme.

- Delayed Charges: Delayed charges are performed to make a supplemental account charge after original services have been rendered and payment has been processed.

- No Show: Consumers can use their payment credentials to make a guaranteed reservation with certain merchant segments. A guaranteed reservation ensures that the reservation will be honoured and allows you to perform a No Show transaction to charge the Consumer a penalty according to your cancellation policy. If no payment is made to guarantee a reservation, then it is necessary to perform a VERIFY Consumer Initiated Transaction at the time of reservation to be able perform a No Show transaction later.

[1] Not all Acquirers support all transaction types.
[2] The Gateway does not currently support incremental authorisations.

A Merchant Initiated Transaction is one whose **`action`** field is one of **PREAUTH**, **SALE** or **VERIFY** and whose **`type`** is one of **2** (MOTO) or **9** (CA) depending on the category.

To indicate the type of MIT, send the **`rtAgreementType`** field as one of the following values:

- **recurring** – subsequent payment as the start of a recurring payment agreement (CA).
- **instalment** – subsequent payment as the start of an instalment payment agreement (CA).
- **unscheduled** – subsequent payment not to a fixed schedule (MOTO)
- **incremental** – subsequent payment to increment initial amount authorised (MOTO)
- **resubmission** – subsequent payment due to failed initial payment (MOTO)
- **reauthorisation** – subsequent payment to refresh expired initial payment (MOTO)
- **delayedcharges** – subsequent payment for additional charges (MOTO)
- **noshow** – subsequent payment as penalty for missed reservation (MOTO)

In addition, the optional **`rtSequenceCount`** field can be used to specify how many transactions will be taken in total, with a value greater than 1, and the optional **`rtSequenceNumber`** field can be used to specify which transaction this is in the sequence, with a value greater than 0.

The **`xref`** of the initial Consumer Initiated Transaction, or previous Merchant Initiated Transaction must be provided as follows:

- For standing order MITs the initial authorisation must have been a successful Consumer Initiated Transaction with Credentials on File. This MIT will be a subsequent use of those Credentials on File. For **recurring** and **instalment** MITs the initial authorisation must have used the same **`rtAgreementType`**. The **`xref`** can be to the previous MIT in which case the Gateway will follow the chain of transactions back to the initial CIT. An **`xref`** is only valid for 13 months and so there cannot be longer between recurring payments, however the Card Scheme or Acquirer may impose a shorter period.

- For industry practice MITs the initial authorisation must be successful (apart from for a **resubmission**) but need not have Credentials on File. For example, it may not be known at the time of the initial authorisation that the MIT would be required and so the initial authorisation would not necessarily have stored the Credentials on File. This is an example of when an industry practice Merchant Initiated Transaction does not require a stored credential

Note: For compatibility with existing practices, Instalment Payments and Recurring Payments MITs use Continuous Authority (CA) **`type`** transactions while other MITs Mail Order/Telephone Order (MOTO) **`type`** transactions. This use of MOTO is different to its use with a Consumer Initiated Transaction (CIT).

12.4 Request Fields

Field Name	Mandatory?	Description
xref	Yes	Reference to initial CIT or previous MIT transaction.
type	Yes[1]	The type of transaction. Possible values are: **1** – E-commerce (ECOM) (CIT). **2** – Mail Order/Telephone Order (MOTO (CIT). **9** – Continuous Authority (CA) (MIT).
rtAgreementType	Yes[2]	Consumer/Merchant agreement type. Possible values are: **cardonfile** – credential storage agreed (CIT/MIT). **recurring** – recurring type agreed (CIT/MIT). **instalment** – instalment type agreed (CIT/MIT). **unscheduled** – ad hoc COF payment (MIT). **incremental**[3] – authorisation amount increment (MIT). **resubmission** – failed authorisation retry (MIT). **reauthorisation** – expired authorisation refresh (MIT). **delayedcharges** – post authorisation charges (MIT). **noshow** – missed reservation penalty (MIT).
rtSequenceCount	No[4]	Number of transactions in a recurring sequence if known.
rtSequenceNumber	No[4]	Transaction number in a recurring sequence if known.
initiator	No[5]	Indicate who initiated the transaction. Possible values are: **consumer** – consumer initiated (CIT) **merchant** – merchant initiated (MIT)

[1] Required differentiate between initial and subsequent **recurring** or **instalment** transaction.
[2] Not required on the initial transaction for a subsequent Industry-Specific Business Practice MIT to be used.
[3] MIT type **incremental** is not currently supported but reserved for future use.
[4] Mandatory for some Acquirers, its use is highly recommended with recurring transactions.
[5] If not provided the Gateway will attempt to determine the initiator from the other request values.

For backwards compatibility, the Gateway will try to automatically identify if a transaction is CIT or MIT from the values provided for the **action**, **type** and **rtAgreementType** fields.

You may also pass the **initiator** field in the request to force a classification. This can be used if the Gateway is unable to correctly determine classify the transaction. If, however, the requested classification is incompatible with the provided request fields then the transaction will fail with a **responseCode** of **66944** (INVALID INITIATOR).

The **initiator** field will be returned in the response with either the value passed in the request or the automatically identified value.

13 Recurring Transaction Agreements

13.1 Background

A Recurring Transaction Agreement (RTA) is used to request that the Gateway should perform repeat payments on your behalf, using pre-agreed amounts and schedules.

An RTA can be configured easily and quickly, using the Merchant Management System (MMS). An RTA can also be set up while performing the initial transaction request, by including the integration request fields described in section 13.3. The RTA is only set up in the transaction results in a successful payment authorisation.

The initial transaction should be either SALE or VERIFY transaction and the **`rtAgreementType`** field should be provided to indicate whether the transactions are part of a recurring or instalment.

Merchants who use this system to implement billing or subscription type payments must use **recurring** or **instalment** type Continuous Authority (CA) transactions to comply with Card Payment Scheme practices. *Your Acquirer may refuse to accept the recurring transactions if they are not subject to an agreement between yourself and your Customer.*

The maximum period between recurring transactions is 13 months, however individual Acquirers may impose a shorter period.

Refer to section 12 for more information on the different types of repeat or recurring transactions.

13.2 Scheduling

There are two different types of scheduling available when requesting the Gateway to take recurring transactions automatically on the Merchant's behalf. In addition, a start date can be provided to allow for a recurring subscription with an initial free trial period.

13.2.1 Fixed Scheduling

Fixed scheduling causes the subsequent transaction to be taken at fixed intervals of time and for fixed amounts. A different initial date and amount or final date and amount can be provided for use when the agreed payment term or amount doesn't exactly divide by the fixed time intervals.

Fixed scheduling is specified by providing an **`rtScheduleType`** field with a value of 'fixed' and providing the **`rtCycleDuration`**, **`rtCycleAmount`** and **`rtCycleCount`** fields to define the interval at which transactions should be taken and the number of transactions to take.

An **`rtCycleCount`** field value of 0 can be provided to indicate that transactions should be taken ad-infinitum until the RTA is stopped.

13.2.2 Variable Scheduling

Variable scheduling causes the subsequent transaction to be taken on prespecified dates and for prespecified amounts.

Variable scheduling is specified by providing an **`rtScheduleType`** field with a value of 'variable' and providing the **`rtSchedule`** field with a value containing an array of one or more schedule records.

Each schedule record must contain the following fields:

Field Name	Mandatory?	Description
`date`	Yes	Date on which to take a payment.
`amount`	Yes	Amount to take on the provided date.

The schedule records should be passed in a sequential array of records, either as nested records or serialised records as described in section 1.7.8. The record field names are case sensitive.

13.3 Request Fields

Field Name	Mandatory?	Description
rtName	No	Free format short name for the agreement.
rtDescription	No	Free format longer description for the agreement.
rtPolicyRef	No	Merchant Policy Reference Number (MPRN).
rtAgreementType	No	Recurring transaction agreement type. Indicates the type of Continuous Payment Authority or Repeat Billing agreement made with the Cardholder. Possible values are: **recurring** – recurring type CPA agreed. **instalment** – instalment type CPA agreed.
rtMerchantID	No	Merchant Account ID to use for the recurring transactions (defaults to **merchantID**).
rtStartDate	No	Start date of agreement (defaults to date received).
rtScheduleType	No	Schedule type. Possible values are: **fixed** – fixed interval schedule (default). **variable** – variable interval schedule.
rtSchedule	Yes[1]	Nested array or serialised string containing payment schedule information as per section 13.2.2.
rtInitialDate	No[2]	Date of initial payment (defaults to **rtStartDate**).
rtInitialAmount	No[2]	Amount of initial payment (defaults to **rtCycleAmount**).
rtFinalDate	No[2]	Date of final payment.
rtFinalAmount	No[2]	Amount of final payment (defaults to **rtCycleAmount**).
rtCycleAmount	No[2]	Amount per cycle (defaults to **amount**).
rtCycleDuration	Yes[2]	Length of each cycle in **rtCycleDurationUnit** units.
rtCycleDurationUnit	Yes[2]	Cycle duration unit. One of: **day**, **week**, **month** or **year**.
rtCycleCount	Yes[2]	Number of cycles to repeat (zero to repeat forever).
rtMerchantData	No	Free format Merchant data field.
rtCloneFields	No	Fields to clone from one recurring transaction to the next. Refer to appendix A-16.

[1] For use with variable schedules only.
[2] For use with fixed schedules only.

13.4 Response Fields

Field Name	Returned?	Description
rtID	Always	Recurring Transaction Agreement ID.
rtResponseCode	Always	Result of setting up RT Agreement. Refer to appendix A-1 for details.
rtResponseMessage	Always	Description of above response code.

14 Dynamic Currency Conversion

14.1 Background

Dynamic Currency Conversion is a process where the amount of a credit card transaction is converted by the Gateway into the Cardholder's local currency as determined by the card's country of issue.

It allows your Customer to pay in their currency, but you will receive the funds in the original transaction currency. You can profit from a mark-up on the exchange rate used, as negotiated with your Acquirer and DCC Provider.

Dynamic Currency Conversion is not available with all Acquirers and must be enabled on your Merchant Account before it can be used. Please contact support to find out whether your Acquirer supports it and if it can be enabled on your Merchant Account.

Dynamic Currency Conversion is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

14.2 Benefits and Limitations

14.2.1 Benefits

- You may see fewer abandoned transactions because the Cardholder can choose to pay in a familiar currency and need not carry out any mental conversions.
- You can gain income from an exchange rate mark-up.
- Currency conversion can be viewed in your transaction reports.
- Fully configurable within the Merchant Management System (MMS).

14.2.2 Limitations

- Cardholders can be sceptical of use due to previously unfavourable exchange rates' being offered.
- There may be extra costs involved, though you may also find that you can recover this from your conversion mark-up.
- Not all card types are supported.

14.3 Hosted Implementation

If your Merchant Account is set up for Dynamic Currency Conversions, then the Hosted Payment Page will offer the Cardholder the option of paying in their native currency.

14.4 Direct Implementation

If your Merchant Account is set up for Dynamic Currency Conversions, then the Gateway will need you to present the conversion offer to the Cardholder and ask whether they wish to accept or refuse the offer.

14.4.1 Initial Request (Fetch DCC Conversion details)

If no DCC conversion details are provided in the initial request, the Gateway will determine whether the transaction is eligible for DCC and whether any suitable conversion rate is available.

If the Gateway determines that the transaction is not eligible for DCC or no conversion rate is available, then it will continue and process it as a normal transaction without DCC.

If a rate was checked for but none was available, then the transaction response will contain DCC conversion details but the **`dccRate`** field will be zero.

If the Gateway determines that the transaction is eligible, it will respond with a **`responseCode`** of **65890 (DCC REQUIRED)** and the response will include details of the DCC offer to ask the Cardholder whether they wish to accept, together with a **`dccRef`** transaction continuation reference.

14.4.2 Continuation Request (Accept/Refuse DCC offer)

When the DCC offer has been presented to the Cardholder and their acceptance or refusal obtained, then a new request should be sent to the Gateway containing their decision in the **`dccAccepted`** and the original **`dccRef`** field received above.

14.4.3 External Authentication Request

You can choose to obtain the DCC details from a third party, in which case you should provide their details as part of a standard request. If the Gateway receives valid third-party DCC details, then it will use those and not attempt to fetch rates from its own DCC Provider.

14.5 Request Fields

14.5.1 Initial Request (Hosted and Direct Integration)

These fields should be sent in addition to basic request fields described in section 2.1.

Field Name	Mandatory?	Description
dccRequired	No[1]	Is 3DS required for this transaction? Possible values are: **N** – DCC is not required. **Y** – DCC is not required.

[1] Any other value will use the preference set in the Merchant Management System (MMS).

14.5.2 Continuation Request (Direct Integration)

These fields may be sent alone[1].

Field Name	Mandatory?	Description
dccRef	Yes	The value of the **dccRef** field in the initial Gateway response.
dccAccepted	Yes	The Cardholder's acceptance to pay using the DCC conversion offered. If they don't accept then the original transaction currency is used. Possible values are: **N** – Cardholder refused the DCC offer. **Y** – Cardholder accepted the DCC offer.

[1] It is only necessary to send the **dccRef** and the **dccAccepted** in the continuation request, because the **dccRef** will identify the Merchant Account and initial request. However, you can send any of the normal request fields to modify or supplement the initial request. Any card details and transaction amount sent in the second request must match those used in the first request, else the second request will fail with a **responseCode** of **64442 (REQUEST MISMATCH)**.

14.5.3 External Authentication Request (Direct Integration)

These fields should be sent in addition to basic request fields from section 2.1.

Field Name	Mandatory?	Description
dccAccepted	No	The Cardholder's acceptance to pay using the DCC conversion offered. If they don't accept then the original transaction currency is used. Possible values are: **N** – Cardholder refused the DCC offer. **Y** – Cardholder accepted the DCC offer.
dccProvider	Yes	DCC provider code name.
dccMargin	Yes	Percentage markup (0-9999.999999)
dccCommission	Yes	Percentage commission (0-9999.999999)
dccSource	Yes	Source of exchange rate
dccCreated	Yes	Time rate sourced (YYYY-MM-DD HH:MM:SS)
dccExpires	Yes	Time rate expires (YYYY-MM-DD HH:MM:SS)
dccCurrencyCode	Yes	Cardholder's currency
dccAmount	No	The amount of the transaction in the Cardholder's minor currency converted at the rate provided.

Note: If DCC is not enabled for the Merchant Account then any DCC conversion fields sent in the request are ignored and the transaction is processed as normal without DCC. If DCC is enabled but the transaction is otherwise not eligible for DCC conversion, then any DCC field sent in the request will result in an error response.

14.6 Response Fields

14.6.1 Initial Response (Direct Integration)

These fields will be returned in addition to the request fields from section 5.5.1 and the basic response fields in section 2.2.

Field Name	Returned?	Description
dccEnabled	Always	Is DCC enabled for this Merchant Account? Possible values are: **N** – Merchant Account is not enabled. **Y** – Merchant Account is enabled.
dccRequired	Always	Is DCC required for this transaction? Possible values are: **N** – DCC is not required. **Y** – DCC is required.
dccProvider	Yes	DCC provider code name.
dccRate	Always	Conversion rate (0-999.999999)
dccMargin	Yes	Percentage markup (0-9999.999999)
dccCommission	Yes	Percentage commission (0-9999.999999)
dccSource	Yes	Source of exchange rate
dccCreated	Yes	Time rate sourced (YYYY-MM-DD HH:MM:SS)
dccExpires	Yes	Time rate expires (YYYY-MM-DD HH:MM:SS)
dccCurrencyCode	Yes	Cardholder's currency
dccAmount	Yes	The amount of the transaction in the Cardholder's minor currency converted at the rate provided.
dccResponseCode	Yes	Any response code generated while performing the DCC processing.
dccResponseMessage	Yes	Any response description corresponding to the above **dccResponseCode**.

14.6.2 Continuation Response (Direct Integration)

No further fields will be returned in addition to the request fields from section 14.5.1; the initial response fields in section 14.6.1; and the basic response fields in section 2.2.

14.6.3 External Authentication Response (Direct Integration)

No further fields will be returned in addition to the request fields from section 14.5.3 and the basic response fields in section 2.2.

15 Purchase Data

15.1 Background

The Gateway can be sent advanced purchase information with each transaction, where required.

The Gateway provides a number of fields that you can use to store advanced purchase information about the transaction, including details on individual items purchased. These fields are only sent to the Acquirer if needed. The stored data can be obtained by sending a QUERY request.

The details may also be used for advanced purposes, such as displaying shopping cart information on the PayPal checkout.

15.1.1 American Express Purchases

Purchases using American Express cards will send a subset of this information to the Card Scheme as appropriate.

With American Express, you can provide tax ***or*** discount reason (but not both). If **`taxAmount`** is provided, then **`taxReason`** is used; if **`discountAmount`** is provided, then **`discountReason`** is used. If both are provided, then **`taxReason`** is used.

Only the description, quantity, and amount of the first six line item details are sent to American Express.

15.1.2 Purchase Orders

These fields together with other advanced fields, as detailed in section 16, can be used to send full information relating to a purchase order and related invoice indicating types; quantities; and agreed prices for products or services. Details on the supplier; shipping; delivery can also be included.

At present, this information is not sent to the Acquirer, unless needed, but future enhancements to the Gateway may include sending such information as Level 2 or 3 Purchasing data as defined by the relevant Card Schemes.

15.2 Request Fields

The following request fields may be sent to provide more information on the breakdown of the purchase amount:

Field Name	Mandatory?	Description
`grossAmount`	No	Total gross amount of sale.
`netAmount`	No	Total net amount of sale.
`taxRate`	No	Total tax rate (percentage).
`taxAmount`	No[1]	Total tax amount of sale.
`taxReason`	No[1]	Reason for above tax (eg VAT).
`discountAmount`	No[1]	Total discount amount of sale.
`discountReason`	No[1]	Reason for above discount.
`handlingAmount`	No	Handling costs.
`insuranceAmount`	No	Insurance costs.

[1] Amex/Diners require either tax or discount, not both.

The following request fields may be sent to provide more information on the purchased items:

Field Name	Mandatory?	Description
`itemXXAmount`[1]	No	Amount for *XX*th item purchased.
`itemXXDescription`[1]	No	Description of *XX*th item purchased.
`itemXXQuantity`[1]	No	Quantity of *XX*th item purchased.
`itemXXGrossAmount`[1]	No	Gross amount for *XX*th item purchased.
`itemXXNetAmount`[1]	No	Net amount for *XX*th item purchased.
`itemXXTaxAmount`[1]	No	Tax amount for *XX*th item purchased.
`itemXXTaxRate`[1]	No	Total tax rate for *XX*th item purchased.
`itemXXTaxReason`[1]	No	Tax reason for *XX*th item purchased.
`itemXXDiscountAmount`[1]	No	Total discount for *XX*th item purchased.
`itemXXDiscountReason`[1]	No	Discount reason for *XX*th item purchased.
`itemXXProductCode`[1]	No	Product code for *XX*th item purchased.
`itemXXProductURL`[1]	No	Shopping cart URL for XXth item purchased.
`itemXXCommodityCode`[1]	No	Commodity code for *XX*th item purchased.

Field Name	Mandatory?	Description
itemXXUnitOfMeasure[1]	No	Unit of measure for *XX*th item purchased.
itemXXUnitAmount[1]	No	Unit amount for *XX*th item purchased.
itemXXImageUrl[1]	No	Image of XXth item purchased.
itemXXSize[1]	No	Size of XXth item purchased in the format 'LengthxWidthxHeight Unit'.
itemXXWeight[1]	No	Weight of XXth item purchased in the format 'Weight Unit'.
items	No	Nested line item records (see below).

[1] XX is a number between 1 and 99.

The purchased items can be passed as either individual **item*XXField*** fields; or as a single **items** field whose value is a sequential array of nested records as described in section 1.7.8.

Both formats cannot be used together. The presence of an **items** field will cause the Gateway to ignore any individual fields.

The Gateway does not currently support **items** to be formatted as a serialised array of records.

Note: no attempt is made to check that any gross, net and tax amounts are correct with respect to each other. It is the sender's responsibility to ensure alternative amount formats are correct.

Line item fields can either be sent 'flat' using field names containing the item row number as a sequential number from 1 to 99; or be sent using nested arrays of the form **items[*XX*][*field*]** where ***XX*** is the row number from 1 to 99 and ***field*** is the field name from the above table without the **item*XX*** prefix and starting with a lowercase first letter. For example, the tax rate for item 5 can be sent either as **item5TaxRate;** or as **items[5][taxRate]**. The two formats should not be mixed. If a request field of **items** is seen, then the 'flat' fields are ignored.

16 Custom Data

You may send arbitrary data with the request by appending extra fields, which will be returned unmodified in the response. These extra fields are merely 'echoed' back and not stored by the Gateway.

Caution should be made to ensure that any extra fields do not match any currently documented fields or possible future fields. One way to do this is to prefix the field names with a value unique to you, the Merchant.

If the request contains a field that is also intended as a response field, then any incoming request value will be overwritten by the correct response value.

The Gateway may add new request and response fields at any time and so your integration must take care not to send request fields that may conflict with future Gateway fields and be able to ignore response fields which it doesn't yet understand.

You can also use the **`merchantData`** field to store custom data with the transaction. This stored data can then be retrieved at a later date, using a QUERY request. Complex data can be stored as per the details for nested records in section 1.7.8, however the Gateway does not currently support this fields to be provided in the serialised record format. Alternatively, you can serialise the data before storing and unserialise it on retrieval.

16.6 Request Fields

Field Name	Mandatory?	Description
`merchantData`	No	Arbitrary data to be stored together with this transaction.

17 Advanced Data

The Gateway provides a number of fields that you can use to store information about the transaction. These fields are only sent to the Acquirer if needed. The stored data can be obtained by sending a QUERY request.

17.1 Customer Request Fields

These fields can be used to store details about the Customer and any relationship between the Customer and Merchant such as any purchase order raised.

If AVS checks are in use, then the Customer and Cardholder are assumed to be the same person and the address and postcode fields are taken as being the registered billing address of the card.

Field Name	Mandatory?	Description
customerName	No	Customer's name.
customerCompany	No	Customer's company (if applicable).
customerAddress	No[1]	Customer's address.
customerPostcode	No[1]	Customer's postcode.
customerTown	No	Customer's town/city.
customerCounty	No	Customer's county/province.
customerCountryCode	No	Customer's country.
customerPhone	No	Customer's phone number.
customerMobile	No	Customer's mobile phone number.
customerFax	No	Customer's fax number.
customerEmail	No	Customer's email address.
customerDateOfBirth	No	Customer's date of birth.
customerOrderRef	No	Customer's reference for this order (Purchase Order Reference).
customerMerchantRef	No	Customer's reference for the Merchant.
customerTaxRef	No	Customer's tax reference number.

[1] Mandatory if AVS checking required.

17.2 Merchant Request Fields

These fields can be used to store details about the Merchant and any relationship between the Merchant and Customer such as any invoice reference.

Field Name	Mandatory?	Description/Value
merchantName	No	Merchant's contact name.
merchantCompany	No	Merchant's company name.
merchantAddress	No	Merchant's contact address.
merchantTown	No	Merchant's contact town/city.
merchantCounty	No	Merchant's contact county.
merchantPostcode	No	Merchant's contact postcode.
merchantCountryCode	No	Merchant's contact country.
merchantPhone	No	Merchant's phone number.
merchantMobile	No	Merchant's mobile phone number.
merchantFax	No	Merchant's fax number.
merchantEmail	No	Merchant's email address.
merchantWebsite	No	Merchant's website. The website must be a fully qualified URL and include at least the scheme and host components.
merchantOrderRef	No	Merchant's reference for this order (Invoice/Sales Reference).
merchantCustomerRef	No	Merchant's reference for the Customer.
merchantTaxRef	No	Merchant's tax reference number.
merchantOriginalOrderRef	No	Reference to a back order.
merchantCategoryCode	No	Scheme assigned Merchant Category Code (MCC).
merchantType	No	Acquirer assigned Merchant type code.
merchantAccountNo	No	Merchant's bank account number

17.3 Supplier Request Fields

These fields can be used to store details about the Supplier. This is where any purchased goods are being supplied by a third-party and not directly from the Merchant.

Field Name	Mandatory?	Description/Value
supplierName	No	Supplier's contact name.
supplierCompany	No	Supplier's company name.
supplierAddress	No	Supplier's contact address.
supplierTown	No	Supplier's contact town/city.
supplierCounty	No	Supplier's contact county.
supplierPostcode	No	Supplier's contact postcode.
supplierCountryCode	No	Supplier's contact country.
supplierPhone	No	Supplier's phone number.
supplierMobile	No	Supplier's mobile phone number.
supplierFax	No	Supplier's fax number.
supplierEmail	No	Supplier's email address.
supplierOrderRef	No	Supplier's reference for this order (Invoice/Sales Reference).
supplierAccountNo	No	Supplier's bank account number

17.4 Delivery Request Fields

These fields can be used to store details about the delivery address. This is where any purchased goods are being delivered to if different from the Customer's address.

Field Name	Mandatory?	Description/Value
deliveryName	No	Name of person receiving the delivery.
deliveryCompany	No	Name of company receiving the delivery.
deliveryAddress	No	Delivery address.
deliveryTown	No	Delivery town/city.
deliveryCounty	No	Delivery county.
deliveryPostcode	No	Delivery postcode.
deliveryCountryCode	No	Delivery country.
deliveryPhone	No	Phone number of delivery location.
deliveryMobile	No	Mobile phone number of delivery location.
deliveryFax	No	Fax number of delivery location.
deliveryEmail	No	Delivery email address.

17.5 Receiver Request Fields

These fields can be used to store details about the recipient of the purchased goods where different from the Customer's and Delivery details. It is most commonly used by Financial Institutions (MCC 6012 Merchants) who need to record the primary recipient of a loan.

Field Name	Mandatory?	Description/Value
receiverName	No	Receiver's contact name.
receiverCompany	No	Receiver's company name.
receiverAddress	No	Receiver's contact address.
receiverTown	No	Receiver's contact town/city.
receiverCounty	No	Receiver's contact county.
receiverPostcode	No	Receiver's contact postcode.
receiverCountryCode	No	Receiver's contact country.
receiverPhone	No	Receiver's phone.
receiverMobile	No	Receiver's mobile phone number.
receiverFax	No	Receiver's fax number.
receiverEmail	No	Receiver's email address.
receiverAccountNo	No	Receiver's account number.
receiverDateOfBirth	No	Receiver's date of birth.

17.6 Shipping Request Fields

These fields can be used to store details about the shipping method and costs.

Field Name	Mandatory?	Description/Value
shippingTrackingRef	No	Shipping tracking reference.
shippingMethod	No	Shipping method (eg Courier, Post, etc.).
shippingAmount	No	Cost of shipping.
shippingGrossAmount	No	Gross cost of shipping.
shippingNetAmount	No	Net cost of shipping.
shippingTaxRate	No	Tax rate as percentage to 2 decimal places.
shippingTaxAmount	No	Tax cost of shipping.
shippingTaxReason	No	Tax reason (eg VAT).
shippingDiscountAmount	No	Discount on shipping.
shippingDiscountReason	No	Reason for discount.

Note: no attempt is made to check that any gross, net and tax amounts are correct with respect to each other. It is the sender's responsibility to ensure alternative amount formats are correct.

17.7 Device Information Fields

These fields can be used to provide details of the device from which the transaction is being made. Although not strictly mandatory, they may be required for fraud checking or 3-D Secure authentication, in which case it is highly recommended that they be provided.

Field Name	Mandatory?	Description/Value
deviceType	No	Type of Consumer's device. One of the following values: **desktop**, **laptop**, **tablet**, **phone**, **other**.
deviceChannel	No	Communications channel used by the Consumer's device. One of the following values: **browser**, **app**, **other**.
deviceIdentity	No[1]	Content of the HTTP User-Agent header received from the Consumer's device. Truncated to 2048 characters maximum.
deviceTimeZone	No[1]	Time zone offset in minutes between UTC and the Consumer's device. The offset is positive if the local time zone is behind UTC and negative if it is ahead.
deviceCapabilities	No[1]	Comma separated list of capabilities supported by the Consumer's device. One or more of the following values: **java**, **javascript**.
deviceAcceptContent	No[1]	Content of HTTP Accept header received from the Consumer's device. Truncated to 2048 characters maximum.
deviceAcceptCharset	No[1]	Content of HTTP Accept-Charset header received from the Consumer's device. Truncated to 2048 characters maximum.
deviceAcceptEncoding	No[1]	Content of HTTP Accept-Encoding header received from the Consumer's device. Truncated to 2048 characters maximum.
deviceAcceptLanguage	No[1]	Content of HTTP Accept-Language header received from the Consumer's device. Truncated to 2048 characters maximum.

Field Name	Mandatory?	Description/Value
deviceScreenResolution	No[1]	Screen resolution of the Consumer's device. Formatted as [HxWxD] where: • H – screen height in pixels • W – screen width in pixels • D – colour depth in bits Screen height and width must be between 1 and 999999 pixels. Colour depth must be one of the following values: **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48**.
deviceOperatingSystem	No	Operating system used by the Consumer's device. One of the following values: **win**, **unix**, **linux**, **macos**, **ios**, **android**, **other**.

[1] This field is mandatory for 3-D Secure unless an alternative is provided via the **threeDSOptions** field.

18 Acquirer Data

The Gateway supports the passing of Acquirer specific data where needed by an individual Acquirer to provide additional or non-standard features.

When supported, this data can be passed in the **`acquirerOptions`** request field, which must be provided using the *record* or *serialised record* format detailed in section 1.7.8.

Please contact our customer support team if you need information about what options can be provided to your Acquirer.

The Gateway also supports the returning of Acquirer specific details in the request in situations where the Gateway considers the data to be of value.

When supported, this data will be returned in the **`acquirerResponseDetails`** response field, which will be returned using the *record* format detailed in section 1.7.8.

In addition to this the Gateway will return the original response code and message received from the Acquirer and any transaction referenced provided by the Acquirer. This later reference can help you identify the transaction when you have access to the Acquirers merchant management portal or need to contact them to query a transaction.

The original Acquirer response code may not be numeric and information on these codes will need to be requested from the Acquirer.

The Gateway may support new acquirer options and return new acquirer details at any time and so your integration must be able to handle such changes and not reject unknown fields.

18.1 Request Fields

Field Name	Mandatory?	Description
acquirerOptions	No	Record containing Acquirer specific options.

18.2 Response Fields

Field Name	Mandatory?	Description
acquirerResponseCode	No	Response code supplied by the Acquirer, maybe prefixed with 'G:' if the Acquirer is itself a payment Gateway.
acquirerResponseMessage	No	Response message supplied the Acquirer.
acquirerResponseDetails	No	Record containing Acquirer specific response details.
acquirerTransactionID	No	Transaction identifier/reference used to identify the transaction in the Acquirer's system.

19 Gateway Wallet

19.1 Background

The Gateway supports an internal digital Wallet that is available to all Merchants using the Gateway.

The Gateway allows you to store your Customer's payment card, billing and delivery address details and other information securely encrypted in its internal Wallet. You can then allow your Customer to select from stored payment cards to check out faster on your website.

Management of this Wallet is done using the Gateway's REST API. However, you can use the Hosted, Direct or Batch Integrations to perform transactions, using cards and addresses stored in the Wallet; or to store new cards and address used with successful transactions.

19.2 Benefits and Limitations

19.2.1 Benefits

- Details can be used from or added to the Wallet with just a few extra integration fields.
- Customers can select from previously stored details, making the checkout process more streamlined, resulting in fewer abandoned carts and thus increasing sales.
- Compatible with existing card base fraud solutions such as Address Verification Service (AVS), 3-D Secure and third-party fraud providers.
- There are no extra costs to use the internal Gateway Wallet.
- The Wallet transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.
- Stored cards are assigned a Card Token which is fully LUHN checkable PAN ending in the same last 4 digits as stored card and thus can be used to replace the PAN in any system that is expecting to store PANs and not arbitrary card identifiers.

19.2.2 Limitations

- The payment details are stored internally by the Gateway and not available for use with other Gateway Merchants or other payment gateways.

19.3 Hosted Implementation

If a transaction is sent to the Hosted Integration, then with the addition of a few extra integration fields, the Customer can be asked whether they wish to save their details in the internal Wallet for future use.

Customers who have payment details already saved will have the option to select from those details rather than having to renter them. Customers will also have the option to delete stored details[1].

The details are only saved if the transaction is successful, ensuring that the Wallet is not filled up with invalid payment details.

The details requiring to be stored in the Wallet are validated when the transaction is performed, prior to any authorisation with the Acquirer. If any of the details are invalid, then the transaction will be aborted with a **`responseCode`** of **66304** (**INVALID_REQUEST**) and a **`responseMessage`** indicating which data could not be stored in the Wallet. Any failure that occurs post authorisation will not abort the transaction but will be available in the appropriate ***`xxxx`*`StoreResponseCode`** response fields.

The **`walletOwnerRef`** field can be used to assign a unique Customer reference to the Wallet, allowing you to identify which of your Customers owns the Wallet. This could be the Customer reference you use within your own Customer accounts or Shopping Cart software. You must ensure that this value is less than 50 characters, or the transaction will be aborted with a **`responseCode`** of **65xxx** (**INVALID_WALLETCUSTOMERREF**).

[1] There is no option to modify stored payment details. However, the Customer can delete them and then restore them.

19.4 Direct Implementation

If a transaction is sent to the Direct Integration, then with the addition of a few extra integration fields, it can be instructed to use payment details stored in the Wallet and/or store the used payment details.

Using stored payment details is similar to performing cross-referenced transactions where the payment details are cloned from a previous transaction[1]. However, in this case the payment details are taken from the Wallet and not a previous transaction.

The details are only saved if the transaction is successful, ensuring that the Wallet is not filled up with invalid payment details.

The details requiring to be stored in the Wallet are validated when the transaction is performed prior to any authorisation with the Acquirer. If any of the details are invalid, then the transaction will be aborted with a **`responseCode`** of **66304** (**INVALID_REQUEST**) and a **`responseMessage`** indicating which data could not be stored in the Wallet. Any failure that occurs post authorisation will not abort the transaction but will be available in the appropriate ***`xxxx`*`StoreResponseCode`** response fields.

The **`walletOwnerRef`** field can be used to assign a unique Customer reference to the Wallet allowing you to identify which of your Customers owns the Wallet. This could be the Customer reference you use within your own Customer accounts or Shopping Cart software. You must ensure that this value is less than 50 characters, or the transaction will be aborted with a **`responseCode`** of **65xxx** (**INVALID_WALLETCUSTOMERREF**).

[1] Refer to appendices A-15 through to A-17 for details on cross referenced transactions and cloning.

19.5 Request Fields

Field Name	Mandatory?	Description
walletID	No	Identifier for an existing Wallet to use.
walletName	No	Name for any new Wallet created.
walletDescription	No	Description for any new Wallet created.
walletOwnerRef	No	Owner Reference for any new Wallet created.
walletData	No	Merchant Data for any new Wallet created.
walletStore	No	Request that all payment details be stored in the Wallet. A new Wallet will be created if needed. Possible values are: **Y**- store all payment details. **N**- store details according to their **xxxStore** value.
cardID	No	Identifier for an existing card stored in a Wallet.
cardToken	No	Identifier for an existing card stored in a Wallet represented as valid PAN ending with same last 4 digits as the stored PAN.
cardName	No	Name for any new card stored.
cardDescription	No	Description for any new card stored.
cardData	No	Merchant Data for any new card stored.
cardStore	No	Request that the payment card details be stored in the Wallet. A new Wallet will be created if needed. Possible values are: **Y**- store the card details. **N**- do not store the card details.
customerAddressID	No	Identifier for an existing address stored in a Wallet.
customerAddressName	No	Name for any new address stored.
customerAddressDescription	No	Description for any new address stored.
customerAddressData	No	Merchant Data for any new address stored.
customerAddressStore	No	Request that the customer address details be stored in the Wallet. A new Wallet will be created if needed. Possible values are: **Y**- store the customer address details. **N**- do not store the customer address details.
deliveryAddressID	No	Identifier for an existing address stored in a Wallet.

deliveryAddressName	No	Name for any new address stored.
deliveryAddressDescription	No	Description for any new address stored.
deliveryAddressData	No	Merchant Data for any new address stored.
deliveryAddressStore	No	Request that the delivery address details be stored in the Wallet. A new Wallet will be created if needed. Possible values are: Y- store the delivery address details. N- do not store the delivery address details.

19.6 Response Fields

These fields will be returned in addition to the request fields from section.

Field Name	Mandatory?	Description
walletStoreResponseCode	No	Result of creating or updating the Wallet details. Refer to appendix A-1 for details.
walletStoreResponseMessage	No	Description of above response code.
cardStoreResponseCode	No	Result of creating or updating the card details. Refer to appendix A-1 for details.
cardStoreResponseMessage	No	Description of above response code.
customerAddresStoreResponseCode	No	Result of creating or updating the address details. Refer to appendix A-1 for details.
customerAddressStoreResponseMessage	No	Description of above response code.
deliveryAddressStoreResponseCode	No	Result of creating or updating the address details. Refer to appendix A-1 for details.
deliveryAddressStoreResponseMessage	No	Description of above response code.

If new items are stored in the Wallet, then their identifiers will be returned in the appropriate **walletID**, **cardID**, **customerAddressID** and **deliveryAddressID** together with any values provided for or assigned by default to the other item fields.

Failure to store any of the details in the Wallet will be reported using the appropriate ***xxxx*StoreResponseCode** response field.

20 Masterpass Wallet Transactions

20.1 Background

Masterpass is a digital wallet from Mastercard that is available to all Merchants using the Gateway. It is an implementation of the Secure Remote Commerce (SRC) standard, aka Click to Pay, which was developed by EMVCo as a neutral solution to replace the original proprietary Masterpass and Visa Checkout wallets.

It allows customers to store their payment and shipping information in one central, secure location. With Masterpass, customers can shop, click, and check out faster on your website.

Customers can store cards from Visa, Mastercard, American Express, and Discover to enable Click to Pay simply and securely. SRC an enhanced online checkout experience and supports all network brands participating in SRC.

Masterpass transactions process and settle just like credit card transactions. You can identify Masterpass transactions in the Merchant Management System by their unique payment type logo, which includes the credit card brand name at the bottom.

There are no additional fees for processing Masterpass transactions – pricing for Masterpass is the same as your other credit card transactions.

Masterpass used to be a proprietary wallet which has now been upgraded to be a Click to Pay SRC compliant wallet and Customers can no longer sign up to or access the original Masterpass wallet.

Existing Merchants using Masterpass v7 will have been automatically migrated to the Click to Pay wallet. Masterpass v6 has been decommissioned is no longer available.

Masterpass is an advanced feature and must be enabled on your Merchant Account before it can be used. Please contact support if you wish to have it enabled.

Masterpass is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

20.2 Benefits and Limitations

20.2.1 Benefits

- The Wallet details are stored externally to the Gateway and available with any third-party Checkout that supports a Click to Pay SRC wallet.

- Customers can select from previously stored details, making the checkout process more streamlined, resulting in fewer abandoned carts and thus increasing sales.

- Compatible with existing card base fraud solutions such as Address Verification Service (AVS), 3-D Secure and third-party fraud providers.

- There are no extra costs to add Masterpass to your Gateway account.

- The Masterpass transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

20.2.2 Limitations

- Your Customer will need a Click to Pay SRC wallet with some stored card details in order to make full use of this payment method. Many banks are automatically enrolling their cards in such a wallet.

- Repeat transactions using the retrieved payment details are supported but may require permission from the wallet.

20.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** that has Masterpass enabled, then the Hosted Payment Page will display a Click to Pay payment button that, when clicked, will open the Masterpass Wallet and allow the Customer to select their payment card and address details.

To customise the Masterpass Wallet experience, you may send various options in the `masterPassCheckoutOptions` field in your initial request.

Additional information available from the Masterpass Wallet will be made available in the `masterPassCheckoutDetails` response field.

Note: Custom Hosted Payment Pages might not support the displaying of the Click to Pay button. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

20.4 Direct Implementation

Masterpass transactions require you to display the Masterpass Pay Wallet to your Customer as part of the transaction flow. The transaction must be done in two stages, with the Wallet being displayed between the stages. They can optionally also be done in three stages, allowing you to display an order confirmation after the Wallet and before authorising the transaction. You can change the amount at this stage to allow for shipping costs when you know the confirmed delivery address the Customer selected from the Wallet.

To customise the Masterpass Wallet experience, you may send various options in the `masterPassCheckoutOptions` field in your initial request.

Additional information available from the Masterpass Wallet will be made available in the `masterPassCheckoutDetails` response field.

20.4.1 Initial Request (Checkout Preparation)

To request that a transaction be processed using details selected from the Customer's Masterpass Wallet, the request must contain a **`paymentMethod`** of 'masterpass' and a **`masterPassCallbackURL`** containing the URL of a page on your server to return to when the Wallet is closed. In addition, you **must** send **`masterPassCheckoutOptions`** to specify that Masterpass v7 with Click to Pay is used (refer to section 20.5.3). When the Gateway receives these two fields, assuming there are no other errors with the request, it will attempt to find a suitable Masterpass enabled Merchant Account in the current account mapping group (refer to Appendix A-6).

If the Gateway is unable to find a suitable account, then the transaction will be aborted and it will respond with a **`responseCode`** of **65569 (MASTERPASS_NOT_SUPPORTED)**.

Otherwise, the Gateway will respond with a **`responseCode`** of **65572 (MASTERPASS_CHECKOUT REQUIRED)** and the response will include a **`masterPassCheckoutURL`** field containing the URL required to load the Masterpass Wallet and a **`masterPassCheckoutOptions`** containing any data required to be sent to the Wallet. The response will also contain a unique **`masterPassData`** field that must be echoed back in the continuation requests. No transaction will have been created by the Gateway at this stage and this request will not appear in the Merchant Management System.

At this point your server must redirect the Customer's browser to the Masterpass Wallet at the provided **`masterPassCheckoutURL`**. Alternatively, the **`masterPassCheckoutURL`** can be used in conjunction with the Masterpass JavaScript code to implement a lightbox style Wallet that allows the Merchants website to remain visible in the background. Further details on how to use the Masterpass JavaScript SDK can be obtained from Mastercard.

20.4.2 Continuation Request (Checkout Details and Authorise)

On completion of the Masterpass Wallet, it will redirect the Customer's browser to the **`masterPassCallbackURL`** provided, including an OAuth token, OAuth verifier and status URL

parameters. If the checkout was successful, the status will be 'success'. Alternatively, if the checkout was cancelled the status will be 'cancel'.

These URL parameters should be sent to the Gateway in the **masterPassToken**, **masterPassVerifier**, **masterPassStatus** fields of a new request. The new request must contain the **masterPassData** received in the initial response. This new request will retrieve the Customer's chosen payment and delivery details from Masterpass and then send the transaction to the Acquirer for authorisation, returning the result similarly to a normal card-based authorisation transaction.

If the continuation request contains any details that would normally be read from the Masterpass y Wallet, then these will take precedence and overwrite the Wallet details. Note: in such cases, the transaction will no longer class as being a Masterpass transaction and will be treated by the Acquirers as if the Wallet was not used.

If the chosen details cannot be retrieved or if the **masterPassStatus** field indicated that the Wallet was cancelled, then the Gateway will return a **responseCode** of **65570 (MASTERPASS_CHECKOUT_FAILURE).**

20.4.3 Separate Checkout Details and Authorisation Requests

You can choose to obtain the Wallet details before sending the transaction for authorisation by sending the **masterPassOnly** field in the above continuation request. If this field is sent with a value of 'Y', then the Gateway will load the Wallet details and then return them to you without sending the request for authorisation. You can then display them and/or adjust the amount, for example, according to delivery charges appropriate to the received delivery address. You should then send a new request, containing the **masterPassData** received, to continue the transaction and authorise it.

If the continuation request contains any details that would normally be read from the Masterpass Wallet, then these will be ignored, and the Wallet details returned. Note: this is different from usual processing, where incoming fields usually take precedence.

The outcome of this request will depend on the value of the **masterPassStatus** field and the ability to communicate with Masterpass. On success, the Gateway will return a **responseCode** of **65571 (MASTERPASS_CHECKOUT_SUCCESS)** and response will include the chosen payment card and address details. If the Wallet was cancelled or if the chosen details cannot be retrieved, then the Gateway will return a **responseCode** of **65570 (MASTERPASS_CHECKOUT_FAILURE).**

Note: this stage can be repeated multiple times by including the **masterPassOnly** field with a value of 'Y' each time. To complete the transaction, the final request must not contain the **masterPassOnly** field or it must not have a value of 'Y'.

20.5 Request Fields

20.5.1 Initial Request (Hosted and Direct Integrations)

These fields should be sent in addition to basic request fields in section 2.1 excluding any card details.

Field Name	Mandatory?	Description
paymentMethod	Yes[1]	Must contain the value 'masterpass' in lower case letters only.
masterPassCallbackURL	Yes[2]	URL on Merchant's server to return to when the Masterpass Wallet is closed.
masterPassCheckoutOptions	No	Record containing options used to customise the Masterpass Wallet. Refer to section 21.5.3 for values.
masterPassCheckoutID	No	Merchant's unique checkout identifier as provided by Masterpass.

[1] Optional for Hosted Integration.
[2] Not required for Hosted Integration.

20.5.2 Continuation Request (Direct Integration)

Field Name	Mandatory?	Description
masterPassData	Yes	Unique reference returned in the initial response.
masterPassStatus	Yes[1]	The Wallet status returned to the Merchant.
masterPassToken	Yes[1]	The OAuth token returned to the Merchant.
masterPassVerifier	Yes[1]	The OAuth verifier returned to the Merchant.
masterPassResourceURL	Yes[1]	An optional resource URL (Masterpass v6 only).
masterPassOnly	No	Pass **Y** to complete the processing as far as the next Wallet stage and then return with the loaded Wallet details.

[1] These fields should be initialised with values received by your website when the wallet redirects to your **masterPassCallbackURL** URL. If the checkout was cancelled, then only the **masterPassStatus** field need be sent to the Gateway.

20.5.3 Wallet Options (Hosted and Direct Integrations)

The following options may be set in the **masterPassCheckoutOptions** field to customise the Masterpass Wallet. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Field Name	Description
version	Masterpass version required. Must be set to **v7**. Possible values are: **v6** – use version 6 of the Masterpass API (default / deprecated). **v7** – use version 7 of the Masterpass API (required).
suppressShippingAddress	Suppress the requirement for the Customer to select a shipping address as well as payment card details. If passed as 'true' then no attempt will be made to return the delivery address fields. Any delivery address fields passed in the transaction will be echoed back unaltered. Possible values are: **false** – shipping address must be selected. **true** – shipping address need not be selected.
merchantCheckoutId	Merchant's unique checkout identifier as provided by Masterpass. Either as passed in the initial request or as configured on the Gateway for your account.
clickToPay	Must be set to **Y** if the **merchantCheckoutId** is included in the options.
allowedCardTypes	List of Masterpass card types to allow selection from within the Wallet. if not provided in the request it will be determined from the card types configured for your Merchant Account and returned in the response.

Important: For backward compatibility with older integrations the Gateway will default to the decommissioned Masterpass v6 without support for Click to Pay.

*To enable the correct support for Click to Pay the **masterPassCheckoutOptions** field must be sent requesting that **version v7** be used with **clickToPay** set to **Y**.*

20.5.4 Response Fields

20.5.5 Initial Response (Direct Integration)

These fields will be returned in addition to the request fields from section 20.5.1 and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
masterPassData	Yes	Unique reference required to continue this transaction when the Masterpass Wallet has completed.
masterPassCheckoutURL	Yes	URL required to load the Masterpass Wallet.
masterPassCheckoutOptions	No	Record containing options used to customise the Masterpass Wallet. Refer to section 21.5.3 for values.

20.5.6 Continuation Response (Direct Integration)

These fields will be returned in addition to the request fields from section 20.5.2; the initial response fields in section 20.5.5; and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
masterPassData	No	Provided, if **masterPassOnly** was used in the continuation response to indicate that a further request will be sent to finalise the transaction.
masterPassWalletID	No	Masterpass Wallet ID.
masterPassCheckout	Yes	Masterpass Wallet details in original retrieved XML format minus any card numbers.
card*XXXX*	Yes	Card details chosen in the Masterpass Wallet as documented in section 2.2.
customer*XXXX*	No[1]	Customer details if provided by the Masterpass Wallet as documented in section 17.1
delivery*XXXX*	No[1]	Delivery details if provided by the Masterpass Wallet as documented in section 17.4

[1] The response will include Customer/billing address and delivery address details if provided by the Masterpass Wallet.

21 PayPal Transactions

21.1 Background

PayPal is an additional payment method that is available to all Merchants using the Gateway who have a PayPal account.

To use PayPal, you will be supplied with a separate PayPal Merchant Account that can be grouped with your main Merchant Account using the account mapping facility as documented in appendix A-6. This allows transactions to be sent using your main Merchant Account and then routed automatically to the PayPal Merchant Account in the same mapping group.

It allows you to offer payment via PayPal in addition to normal card payments.

PayPal transactions will appear in the Merchant Management System (MMS) alongside any card payments and can be captured, cancelled and refunded in the same way as card payments.

PayPal transactions can also be used for recurring billing but require you to indicate in the initial transaction that it will be the basis for recurring billing and that a billing agreement will be entered into between your Customer and PayPal when they agree to the payment.

PayPal transactions cannot be used for ad-hoc Credentials on File (COF) repeat transactions unless a billing agreement has been set up.

For more information on how to accept PayPal transactions, please contact customer support.

PayPal is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

21.2 Benefits and Limitations

21.2.1 Benefits

- Provides your customers with the flexibility of paying by using their PayPal account when this is more suitable to them than using a traditional credit or debit card.
- The in-context PayPal Express Checkout helps improve conversion rates with an easier way to pay without customers leaving your website.
- There are no extra costs for adding a PayPal Merchant Account. However, you will still be liable for the PayPal transaction fees.
- The full PayPal transaction information is available and returned as part of the transaction.
- Transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

21.2.2 Limitations

- You will need a PayPal account.
- Recurring transactions are not supported unless as part of a prearranged billing agreement.
- Independent refunds that are not tied to a previous sale transaction are not supported without prior agreement.
- Transactions require a browser in order to display the PayPal Checkout.
- The PayPal Checkout cannot be opened from within a browser IFRAME and so care must be taken to ensure that any PayPal Checkout button is not placed within such an IFRAME.

21.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** that is part of a routing group containing a PayPal Merchant, then the Hosted Payment Page will display a PayPal payment button that, when clicked, will open the PayPal Checkout and allow the Customer to pay using their PayPal account.

To customise the PayPal Checkout experience, you may send various options in the **`payPalCheckoutOptions`** field in your initial request.

Additional information available from the PayPal Checkout will be made available in the **`checkoutDetails`** response field.

Note: Custom Hosted Payment Pages might not support the displaying of the PayPal Checkout button. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

21.4 Direct Implementation

PayPal transactions require you to display the PayPal Checkout to your Customer as part of the transaction flow. The transaction must be done in two stages, with the Checkout being displayed between the stages. They can also be optionally done in three stages allowing you to display an order confirmation after the Checkout and before authorising the transaction. You can change the amount at this stage to allow for shipping costs when you know the confirmed delivery address the Customer selected as part of the PayPal Checkout.

PayPal supports the normal payment and management actions. This section explains how to make payment requests. Management requests are performed as detailed in section 3.

To customise the PayPal Checkout experience, you may send various options in the **`checkoutOptions`** field in your initial request.

Additional information available from the PayPal Checkout will be made available in the **`checkoutDetails`** response field.

The direct integration uses two complex fields to pass data between the PayPal Checkout and the and the Gateway. The **`checkoutRequest`** field will be provided by the Gateway and is a record whose name/value properties should be used with the PayPal JavaScript SDK to open the Checkout. The corresponding **`checkoutResponse`** field should be returned to the Gateway and must be a record containing name/value properties received from the Checkout when it redirects the Cardholder's browser back to the URL provided using the **`checkoutRedirectURL`** on completion.

The contents of the **`checkoutOptions`**, **`checkoutDetails`**, **`checkoutRequest`** and **`checkoutResponse`** fields are formatted using the *record* format detailed in section 1.7.8, the **`checkoutOptions`** field also supports being provided using the *serialised record* format.

21.4.1 Initial Request (Checkout Preparation)

To request that a transaction be processed via PayPal the request must contain a **`paymentMethod`** of 'paypal' and a **`checkoutRedirectURL`** containing the URL of a page on your server to return to when the Checkout is closed. In addition, you may send **`checkoutOptions`** to customise the Checkout experience. When the Gateway receives this **`paymentMethod`**, assuming there are no other errors with the request, it will attempt to find a suitable PayPal Merchant Account in the current account mapping group (refer to appendix A-6).

If the Gateway is unable to find a suitable account, then the transaction will be aborted and it will respond with a **`responseCode`** of **66364 (INVALID PAYMENTMETHOD)**.

Otherwise, the Gateway will respond with a **`responseCode`** of **65826 (CHECKOUT REQUIRED)** and included in the response will be a **`checkoutURL`** field containing the URL required to load Checkout and a **`checkoutRequest`** containing any data required to be sent to the Checkout. The response will also contain a unique **`checkoutRef`** which must be echoed back in the continuation requests.

At this point your server must redirect the Customer's browser to the provided **checkoutURL**. Alternatively, the **checkoutURL** can be used in conjunction with the PayPal In-Context JavaScript code to implement an In-context Checkout which allows the Merchants website to remain visible in the background. Further details on how to use the In-Context Checkout are provided in the PayPal guide at https://developer.paypal.com/docs/classic/express-checkout/in-context/enable_in_context_checkout/.

21.4.2 Continuation Request (Checkout Details and Authorise)

On completion of the PayPal Checkout it will redirect the Customer's browser to the **checkoutRedirectURL** provided including a **token** and **status** URL parameters. If the checkout was successful, the status will be 'success' alternatively if the Checkout was cancelled the status will be 'cancel'. The received redirect request parameters inclusive of these **token** and **status** parameters should then be sent to the Gateway in the **checkoutResponse** fields of a new request. The **checkoutResponse** field can be sent either as the original URL query string received or as an array of the decoded query parameters. This new request will load the Checkout details including any delivery address if required and send the transaction to PayPal for authorisation, returning the result as per a normal authorisation transaction. The new request must contain the **checkoutRef** received in the initial response.

21.4.3 Separate Checkout Details and Authorisation Requests

You can choose to obtain the Checkout details before actually sending the transaction for authorisation by sending the **checkoutOnly** field in the above continuation request. If this field is sent with a value of 'Y' then the Gateway will load the Checkout details and then return them to you without sending the request for authorisation. You can then display them and/or adjust the amount, for example, according to delivery charges appropriate to the received delivery address. You should then send a new request containing the **checkoutRef** received to continue the transaction and authorise it.

Note: this stage can be repeated multiple times by including the **checkoutOnly** field with a value of 'Y' each time. To complete the transaction, the final request must not contain the **checkoutOnly** field or it must not have a value of 'Y'.

21.5 Request Fields

21.5.1 Initial Request (Hosted and Direct Integrations)

These fields should be sent in addition to basic request fields in section 2.1 excluding any card details.

Field Name	Mandatory?	Description
paymentMethod	Yes[1]	Must contain the value 'paypal' in lower case letters only.
checkoutRedirectURL	Yes[2]	URL on Merchant's server to return to when the PayPal Checkout is closed.
checkoutOptions	No[3]	Record containing options used to customise the PayPal Checkout. Refer to section 21.5.3 for values.
payPalCheckoutOptions	No[4]	Record containing options used to customise the PayPal Checkout. Refer to section 21.5.3 for values.

[1] Optional for Hosted Integration.
[2] Not required for Hosted Integration.
[3] Direct Integration Only
[4] Hosted Integration Only

21.5.2 Continuation Request (Direct Integration)

These fields may be sent alone[1].

Field Name	Mandatory?	Description
checkoutRef	Yes	Unique reference return in the initial response.
checkoutResponse	Yes	The GET and or POST data received by the **checkoutRedirectURL** page.
checkoutOnly	No	Pass **Y** to complete the processing as far as the next Checkout stage and then return with the loaded Checkout details.

[1] It is only necessary to send the **checkoutRef** and the **checkoutResponse** in the continuation request because the **checkoutRef** will identify the Merchant Account and initial request. The message does not need to be signed. You can send any of the normal request fields to modify or supplement the initial request – however, in this case the request should be signed. The **checkoutRedirectURL** and **checkoutOptions** fields sent in the initial request cannot be modified and any sent in the second request must match those used in the first request, or the second request will fail with a **responseCode** of **64442** (**REQUEST MISMATCH**).

21.5.3 Checkout Options (Hosted and Direct Integrations)

The following options may be set in the **`payPalCheckoutOptions`** Hosted Integration field or the **`checkoutOptions`** Direct Integration field to customise the PayPal Checkout. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Option Name	Description
`inContext`	Use the in-context PayPal Checkout rather than the full screen Checkout when possible. Possible values are: **0** – use the full screen Checkout. **1** – use the in-context Checkout if possible.
`userAction`	Determines whether buyers complete their purchases on PayPal or on your website. Possible values are: **commit** – sets the submit button text to 'Pay Now' on the PayPal Checkout. This text lets buyers know that they complete their purchases if they click the button. **continue**– sets the submit button text to 'Continue' on the PayPal Checkout. This text lets buyers know that they will return to the Merchant's cart to complete their purchases if they click the button.
`maxAmount`	The expected maximum total amount of the order, including shipping and taxes. PayPal refer to this field as MAXAMT.
`reqBillingAddress`	Determines whether the buyer's billing address on file with PayPal is returned. **This feature must be enabled by PayPal**.
`reqConfirmShipping`	Determines whether the buyer's shipping address on file with PayPal must be a confirmed address. Possible values are: **0** – does not need to be confirmed **1** – must be confirmed
`noShipping`	Determines whether PayPal displays shipping address. Possible values are: **0** – display the shipping address **1** – do not display shipping address and remove shipping information **2** – If no **`delivery`*`XXXX`*** fields passed, PayPal obtains them from the buyer's account profile.
`addrOverride`	Determines whether the PayPal Checkout displays the shipping address sent using the **`delivery`*`XXXX`*** fields and not the shipping address on file with PayPal for this buyer. Displaying the PayPal street address on file does not allow the buyer to edit that address. Possible values are: **0** – PayPal should not display the address. **1** – PayPal should display the address.
`localeCode`	Locale of the pages displayed by PayPal during Express Checkout. It is either a two-letter country code or five-character locale code supported by PayPal.

Option Name	Description
allowNote	Enables the buyer to enter a note to the merchant on the PayPal page during Checkout. The note is returned in the **checkoutDetails** response field.
pageStyle	Name of the Custom Payment Page Style used for the PayPal Checkout. It is the same name as the Page Style Name used when adding styles in the PayPal Account.
payflowColor	The HTML hex colour code for the PayPal Checkout's background colour. By default, the colour is white (FFFFFF).
cardBorderColor	The HTML hex colour code for the PayPal Checkout's principal identifying colour. The colour will be blended to white in a gradient fill that borders the cart review area.
hdrImg	URL for the image you want to appear at the top left of the payment page. The image has a maximum size of 750 pixels wide by 90 pixels high. PayPal requires that you provide an image that is stored on a secure (https) server. If you do not specify an image, the business name displays.
logoImg	A URL to your logo image. Use a valid graphics format, such as .gif, .jpg, or .png. Limit the image to 190 pixels wide by 60 pixels high. PayPal crops images that are larger. PayPal places your logo image at the top of the cart review area.
landingPage	Type of PayPal Checkout to display. Possible values are: **Billing** – Non-PayPal account **Login** – PayPal account login
channelType	Type of channel. Possible values are: **Merchant** – Non-auction seller **eBayItem** – eBay auction
solutionType	Type of Checkout flow. Possible values are: **Sole** – Buyer does not need to create a PayPal account to check out. This is referred to as PayPal Account Optional. **Mark** – Buyer must have a PayPal account to check out.
totalType	Type declaration for the label to be displayed in MiniCart for UX. Possible values are: **Total** **EstimatedTotal**
brandName	A label that overrides the business name in the PayPal account on the PayPal Checkout.
customerServiceNumber	Merchant Customer Service number displayed on the PayPal Checkout.

Option Name	Description
buyerEmailOptInEnable	Enables the buyer to provide an email address on the PayPal pages to be notified of promotions or special events. Possible values are: **0** – Do not enable buyer to provide email. **1** – Enable the buyer to provide email.
noteToBuyer	A note from the merchant to the buyer that will be displayed in the PayPal Checkout.
paymentAction	Defines how to obtain payment. This can be used to override any `captureDelay` setting that can also be used to indicate a **Sale** or **Authorisation** only. Possible values are: **Sale** – sale with immediate capture. **Authorization** – authorisation subject to later capture (note spelling). **Order** – order subject to later authorisation and capture.
allowedPaymentMethod	The payment method type. Specify the value InstantPaymentOnly
insuranceOptionOffered	Indicates whether insurance is available as an option that the buyer can choose on the PayPal Review page. Possible values are: **true** – The Insurance option displays 'Yes' and the `insuranceAmount`. If true, the total shipping insurance for this order must be a positive number. **false** – The Insurance option displays 'No'.
multiShipping	Indicates whether this payment is associated with multiple shipping addresses. Possible values are: **0** – Single/No shipping address. **1** – Multiple shipping addresses.
noteText	Note to the Merchant.
bucketCategoryType	The category of a payment. Possible values are: **1** – International shipping **2** – Local delivery **3** – BOPIS, Buy online pick-up in store **4** – PUDO, Pick-up drop-off
locationType	Type of merchant location. Required if the items purchased will not be shipped, such as, BOPIS (Buy Online Pick-up In Store) or PUDO (Pick-Up Drop-Off) transactions. Possible values are: **1** – Consumer. **2** – Store, for BOPIS transactions. **3** – PickupDropoff, for PUDO transactions.
locationID	Location ID specified by the merchant for BOPIS (Buy Online Pick-up In Store) or PUDO (Pick-Up Drop-Off) transactions.

Option Name	Description
sellerPayPalAccountID	Unique identifier for the Merchant. For parallel payments, this field is required and must contain the Payer Id or the email address of the Merchant.
invNum	Merchant's invoice or tracking number.
custom	Custom field for your own use.
buyerID	The unique identifier provided by eBay for this buyer. The value may or may not be the same as the username. In the case of eBay, it is different.
buyerUsername	The username of the user at the marketplaces site.
buyerRegistrationDate	Date when the user registered with the marketplace. In UTC/GMT format, for example, 2013-08-24T05:38:48Z.
allowPushFunding	Indicates whether the Merchant can accept push funding. Possible values are: **0** – Merchant cannot accept push funding. **1** – Merchant can accept push funding.
userSelectedFundingSource	This element could be used to specify the preferred funding option for a guest user. However, the `landingPage` Checkout option must also be set to **Billing**, otherwise, it is ignored. Possible values are: **ChinaUnionPay.** **CreditCard.** **ELV.** **QIWI.**
billingType	Type of billing agreement for reference transactions. You must have permission from PayPal to use this field. Possible values are: **MerchantInitiatedBilling** – PayPal creates a billing agreement for each transaction associated with buyer. **MerchantInitiatedBillingSingleAgreement** – PayPal creates a single billing agreement for all transactions associated with buyer. Use this value unless you need per-transaction billing agreements.
billingAgreementDescription	Description of goods or services associated with the billing agreement. This field is required for each recurring payment billing agreement. PayPal recommends that the description contain a brief summary of the billing agreement terms and conditions. For example, buyer is billed at "9.99 per month for 2 years".

Option Name	Description
paymentType	Type of PayPal payment you require for the billing agreement. Possible values are: **Any** – The merchant accepts any payment method for the billing agreement, even if it could take a few working days for the movement of funds to the merchant account. This includes echeck, in addition to credit or debit cards and PayPal balance. **InstantOnly** – The payment options accepted by the merchant are credit cards, debit cards or PayPal balance only because the merchant expects immediate payment.
taxIDType	Buyer's tax ID type. This field is required for Brazil and used for Brazil only. For Brazil use only: The tax ID type is BR_CPF for individuals and BR_CNPJ for businesses.
taxID	Buyer's tax ID. This field is required for Brazil and used for Brazil only. For Brazil use only: The tax ID is 11 single-byte characters for individuals and 14 single-byte characters for businesses
returnFMFDetails	Flag to indicate whether you want the results returned by Fraud Management Filters when doing a recurring/reference transaction. Possible values are: **0** – Do not receive FMF details (default). **1** – Receive FMF details.
riskSessionCorrelationID	The ID of the risk session for correlation purposes when doing a recurring/reference transaction.
merchantSessionID	The buyer session identification token when doing a recurring/reference transaction.
buttonSource[1]	PayPal Partner's BN Code (if required).

[1] The BN code is the unique button source code provided by PayPal to its partners and added by its partners to the PayPal buttons used by merchants to offer the PayPal Services that are enabled through Partner Product. The button source code provides a means of identifying and tracking referred merchants' payments.

For further information on the options, refer to the PayPal Express Checkout documentation: https://developer.paypal.com/docs/classic/api/merchant/SetExpressCheckout_API_Operation_NVP/.

21.5.4 Purchase details (Hosted and Direct Integrations)

The following request fields may be sent to provide information on the purchased items and to populate the cart on the PayPal Checkout.

Field Name	Mandatory?	Description
shippingAmount	No	Shipping costs.
shippingDiscountAmount	No	Discount applied to shipping costs.
handlingAmount	No	Handling costs.
insuranceAmount	No	Insurance costs.
item*XX*Description	No	Description of *XX*th item purchased.
item*XX*Quantity	No	Quantity of *XX*th item purchased.
item*XX*Amount	No	Gross amount for *XX*th item purchased.
item*XX*TaxAmount	No	Tax amount for *XX*th item purchased.
item*XX*ProductCode	No	Product code for *XX*th item purchased.
item*XX*ProductURL	No	Shopping cart URL for *XX*th item purchased.
item*XX*Size	No	Size of *XX*th item purchased in the format 'LengthxWidthxHeight Unit'.
item*XX*Weight	No	Weight of X*X*th item purchased in the format 'Weight Unit'.
items	No	Nested array of line items.

Refer to section 15.2 for more information on these fields.

Note: The shopping cart items must total to the amount specified in the transaction. If they do not, cart items will not be sent to the PayPal Checkout.

21.6 Response Fields

21.6.1 Initial Response (Direct Integration)

These fields will be returned, in addition to the request fields from section 21.5.1 and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
checkoutRef	Yes	Unique reference required to continue this transaction when the PayPal Checkout has completed.
checkoutName	Yes	Unique name of the Checkout. For PayPal this is the value **paypal**.
checkoutURL	Yes	URL required to load the PayPal Checkout.
checkoutRequest	No	Record containing details to send to provide to the PayPal Checkout.
checkoutOptions	No	Record containing any Checkout options passed in the request.
acquirerResponseDetails	Yes	Record containing details about the PayPal response containing any error messages and codes. This can be used together with the normal **responseCode** and **responseMessage** response fields to determine further the reason for any failure.

21.6.2 Continuation Response (Direct Integration)

These fields will be returned, in addition to the request fields from section21.5.2, the initial response fields in section 21.6.1 and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
checkoutRef	Yes	Provided if **checkoutOnly** was used in the continuation response to indicate that a further request will be sent to finalise the transaction.
checkoutName	Yes	Unique name of the Checkout. For PayPal, this is the value **paypal**.
checkoutDetails	Yes	Record containing options used to customise the PayPal Checkout. Refer to section 21.6.3 for values.
customer*XXXX*	No[1]	Customer details if provided by the PayPal Checkout as documented in section 17.1
delivery*XXXX*	No[1]	Delivery details if provided by the PayPal Checkout as documented in section 17.4
acquirerResponseDetails	Yes	Record containing details about the PayPal response containing any error messages and codes. This can be used together with the normal **responseCode** and **responseMessage** response fields to determine further the reason for any failure.

[1] The response will include Customer/billing address and delivery address details if provided by the PayPal Checkout.

21.6.3 Checkout Details (Hosted and Direct Integration)

The following details may be provided in the **`checkoutDetails`** field included in the response. The details will be returned using the *record* format detailed in section 1.7.8

Sub-Field Name	Mandatory?	Description
`correlationID`	No	Correlation ID, which uniquely identifies the transaction to PayPal.
`checkoutStatus`	No	Status of the Checkout session. If payment is completed, the transaction identification number of the resulting transaction is returned. Possible values are: **PaymentActionNotInitiated** **PaymentActionFailed** **PaymentActionInProgress** **PaymentActionCompleted**
`invNum`	No	Merchant's invoice or tracking number, as set sent in **`checkoutDetails.invNum`** or assigned by the Gateway.
`custom`	No	Merchant's invoice or tracking number, as set sent in **`checkoutDetails.custom`** or assigned by the Gateway.
`payPalAdjustment`	No	A discount or gift certificate offered by PayPal to the buyer. This amount is represented by a negative amount. If the buyer has a negative PayPal account balance, PayPal adds the negative balance to the transaction amount, which is represented as a positive value.
`buyerMarketingEmail`	No[1]	Buyer's marketing email address.
`note`	No[2]	Buyer's note to the Merchant.
`cartChangeTolerance`	No	Indicates whether a cart's contents can be modified. If this parameter is not returned, then assume the cart can be modified. This will return **NONE** if financing was used in Germany. Possible values are: **NONE** – The cart cannot be changed. **FLEXIBLE** – The cart can be changed.
`payerID`	No	Buyer's PayPal Customer Account ID.
`payerStatus`	No	Buyer's PayPal status. Possible values are: **verified** **unverified**

Sub-Field Name	Mandatory?	Description
billingName	No[3]	Buyer's name. Also returned in **customerName.**
firstName	No[4]	Buyer's first name. Also returned in **customerName.**
middleName	No[2]	Buyer's middle name. Also returned in **customerName.**
lastName	No[2]	Buyer's last name. Also returned in **customerName.**
suffix	No[2]	Buyer's name suffix. Also returned in **customerName.**
business	No	Buyer's business name. Also returned in **customerCompany.**
street	No	Buyer's street first line. Also returned in **customerAddress.**
street2	No	Buyer's street second line. Also returned in **customerAddress.**
city	No	Buyer's city Also returned in **customerTown.**
state	No	Buyer's state. Also returned in **customerCounty.**
zip	No	Buyer's postal code. Also returned in **customerPostcode.**
countryCode	No	Buyer's country code. (ISO 2 char. code) Also returned in **customerCountryCode.**
countryName	No	Buyer's country name.
phoneNum	No	Buyer's contact phone number. Also returned in **customerPhone.**
email	No	Buyer's email address. Also returned in **customerEmail.**
shipToName	No	Name of person/entity to ship to. Also returned in **deliveryName.**
shipToStreet	No	Ship to street first line. Also returned in **deliveryAddress.**
shipToStreet2	No	Ship to street second line. Also returned in **deliveryAddress.**
shipToCity	No	Ship to city. Also returned in **deliveryTown.**

Sub-Field Name	Mandatory?	Description
shipToState	No	Ship to state. Also returned in **deliveryCounty.**
shipToZip	No	Ship to postal code. Also returned in **deliveryPostcode.**
shipToCountryCode	No	Ship to country code. (ISO 2 char. code) Also returned in **deliveryCountryCode.**
shipToCountryName	No	Ship to country name.
shipToPhoneNum	No	Ship to phone number. Also returned in **deliveryPhone.**
shipToAddressStatus	No	Status of shipping address on file with PayPal. Possible values are: **none** **Confirmed** **Unconfirmed**
addressNormalizationStatus	No[5]	The PayPal address normalisation status for Brazilian addresses. Possible values are: **None** **Normalized** **Unnormalized** **UserPreferred**
amount	No	Total amount for this order.
itemAmount	No	Total item amount for this order.
taxAmount	No	Tax amount for this order.
exchangeRate	No	Exchange rate for this order.
shippingAmount	No	Shipping amount for this order.
handlingAmount	No	Handling amount for this order.
insuranceAmount	No	Insurance amount for this order.
shipDiscountAmount	No	Shipping discount amount for this order.
desc	No	Description of items the buyer is purchasing.
currencyCode	No	ISO 3-letter currency code.

Sub-Field Name	Mandatory?	Description
`isFinancing`	No	Indicates whether the Customer ultimately was approved for and chose to make the payment using the approved instalment credit. Possible values are: **FALSE** – financing not in use **TRUE** – financing approved and used
`financingFeeAmount`	No	The transaction financing fee associated with the payment. This will be set to the instalment fee amount that is the same as the estimated cost of credit or the interest/fees amount the user will have to pay during the lifetime of the loan. This field will only be included in instalment credit orders. In the case of “same as cash” or “no interest” offers, this will be set to 0.
`financingTerm`	No	The length of the financing term, in months. Example values are 6, 12, 18 and 24 months.
`financingMonthlyPayment`	No	This is the estimated amount per month that the Customer will need to pay including fees and interest.
`financingTotalCost`	No	This is the estimated total payment amount including interest and fees that the user will pay during the lifetime of the loan.
`financingDiscountAmount`	No	Discount amount for the buyer if paid in one instalment.
`regularTakeFeeAmount`	No	Fee of the regular take rate on the transaction amount. It could be equal to `financingDiscountAmount` in the case of non-instalment transactions.
`noteText`	No	Note to Merchant.
`transactionID`	No	PayPal transaction ID.
`allowedPaymentMethod`	No	The payment method type as specified in the initial request.
`paymentRequestID`	No	A unique identifier of the specific payment request.
`bucketCategoryType`	No	The category of a payment as specified in the initial request.
`instrumentCategory`	No	Identifies the category of the promotional payment instrument. Possible values are: **1** – PayPal Credit® (formerly Bill Me Later®). **2** – A Private Label Credit Card (PLCC) or co-branded payment card.

Sub-Field Name	Mandatory?	Description
instrumentID	No	An instrument ID (issued by the external party) corresponding to the funding source used in the payment.
shippingCalculationMode	No	Describes how the options that were presented to the buyer were determined. Possible values are: **API – Callback** **API – Flatrate**
insuranceOptionSelected	No	The option that the buyer chose for insurance. Possible values are: **Yes –** opted for insurance. **No –** did not opt for insurance.
shippingOptionIsDefault	No	Indicates whether the buyer chose the default shipping option. Possible values are: **true –** chose the default shipping option. **false –** did not choose the default shipping option.
shippingOptionAmount	No	The shipping amount that the buyer chose.
shippingOptionName	No	The name of the shipping option, such as Air or Ground.
scheduledShippingDate	No	The scheduled shipping date is returned only if scheduled shipping options are passed in the request.
scheduledShippingPeriod	No	The scheduled shipping period is returned only if scheduled shipping options are passed in the request.
sellerPayPalAcountID	No	Unique identifier for the merchant. For parallel payments, this field contains either the Payer Id or the email address of the merchant.
taxIDType	No	Buyer's tax ID type. This field is required for Brazil and used for Brazil only. For Brazil use only: The tax ID type is BR_CPF for individuals and BR_CNPJ for businesses.
taxID	No	Buyer's tax ID. This field is required for Brazil and used for Brazil only. For Brazil use only: The tax ID is 11 single-byte characters for individuals and 14 single-byte characters for businesses

Sub-Field Name	Mandatory?	Description
billingAgreementID	No	Identification number of the billing agreement. When the buyer approves the billing agreement, it becomes valid and remains valid until it is cancelled by the buyer.
billingAgreementAcceptedStatus	No	Indicates whether the buyer accepted the billing agreement for a recurring payment. Currently, this field is always returned in the response for agreement based products, such as subscriptions; reference transactions; recurring payments; and regular single payment transactions. **0 –** Not accepted. **1 –** Accepted.
paymentStatus	No	Status of the payment. Possible values are: **None –** No status. **Canceled-Reversal –** A reversal has been cancelled: for example, when you win a dispute and the funds for the reversal have been returned to you. **Completed –** The payment has been completed and the funds have been added successfully to your account balance. **Denied –** You denied the payment. This happens only if the payment was previously pending because of possible reasons described for the **pendingReason** element. **Expired –** The authorisation period for this payment has been reached. **Failed –** The payment has failed. This happens only if the payment was made from your buyer's bank account. **In-Progress –** The transaction has not terminated: for example, an authorisation may be awaiting completion. **Partially-Refunded –** The payment has been partially refunded. **Pending –** The payment is pending. See the **pendingReason** field for more information. **Refunded –** You refunded the payment. **Reversed –** A payment was reversed due to a chargeback or other type of reversal. The funds have been removed from your account balance and returned to the buyer. The reason for the reversal is specified in the **reasonCode** element. **Processed –** A payment has been accepted. **Voided –** An authorisation for this transaction has been voided.

Sub-Field Name	Mandatory?	Description
`refundStatus`	No	Status of the refund. Possible value are: **none –** returned if the refund fails **instant** – refund was instant **delayed** – refund was delayed
`pendingReason`	No[6]	The reason the payment is pending. Possible values are: **none –** No pending reason. **address –** The payment is pending because your buyer did not include a confirmed shipping address and your Payment Receiving Preferences is set such that you want to accept or deny each of these payments manually. To change your preference, go to the Preferences section of your Profile. **authorization[7] –** The payment is pending because it has been authorised but not settled. You must capture the funds first. **echeck –** The payment is pending because it was made by an eCheck that has not yet cleared. **intl –** The payment is pending because you hold a non-U.S. account and do not have a withdrawal mechanism. You must manually accept or deny this payment from your Account Overview. **multi-currency –** You do not have a balance in the currency sent, and you do not have your Payment Receiving Preferences set to automatically convert and accept this payment. You must manually accept or deny this payment. **order –** The payment is pending because it is part of an order that has been authorised but not settled. **payment-review –** The payment is pending while it is being reviewed by PayPal for risk. **regulatory-review –** The payment is pending while we make sure it meets regulatory requirements. You will be contacted again in from 24 to 72 hours with the outcome of the review. **unilateral –** The payment is pending because it was made to an email address that is not yet registered or confirmed. **verify –** The payment is pending because you are not yet verified. You must verify your account before you can accept this payment. **other –** The payment is pending for a reason other than those listed above. For more information, contact PayPal Customer Service.

Sub-Field Name	Mandatory?	Description
reasonCode	No	The reason for a reversal if the transaction type is reversal. Possible values are: **none –** No reason code. **chargeback –** A reversal has occurred on this transaction due to a chargeback by your buyer. **guarantee –** A reversal has occurred on this transaction due to your buyer triggering a money-back guarantee. **buyer-complaint –** A reversal has occurred on this transaction due to a complaint about the transaction from your buyer. **refund –** A reversal has occurred on this transaction because you have given the buyer a refund. **other –** A reversal has occurred on this transaction due to a reason not listed above.
protectionEligibilityType	No	The kind of seller protection in force for the transaction. Possible values are: **ItemNotReceivedEligible –** Merchant is protected by PayPal's Seller Protection Policy for Item Not Received. **UnauthorizedPaymentEligible**[7] **–** Merchant is protected by PayPal's Seller Protection Policy for Unauthorised Payments. **Ineligible –** Merchant is not protected under the Seller Protection Policy. (Multiple values are separated by commas)
feeAmount	No	PayPal fee amount charged for the transaction.
settleAmount	No	Amount deposited in your PayPal account after a currency conversion.
storeID	No	Store identifier as entered in the transaction.
terminalID	No	Terminal identifier as entered in the transaction.

[1] Only available if email option was enabled in the initial request using **checkoutOptions.buyerEmailOptInEnable** option.
[2] Only available if the leaving of notes was enabled in the initial request using **checkoutOptions.allowNote** option.
[3] Permission is needed from PayPal to support this field.
[4] These fields are used when no permission to use `billingName`.
[5] This field is passed directly to PayPal and therefore the field name and value must be spelt 'ize' and not 'ise'.
[6] **pendingReason** is returned in the response only if **paymentStatus** is **Pending**.
[7] This value is received directly from PayPal and so will use the 'ize' and not 'ise' spelling.

21.7 Transaction Lifecycle

PayPal transactions will use the normal Authorise, Capture life cycle as documented in appendix A-13.1 with the following differences. In addition, the PayPal **`paymentAction`** option can be included in the **`checkoutOptions`** field to alter the normal payment lifecycle further, to allow an Order, Authorise, Capture model or a straight Sale model to be specified.

21.7.1 Order

If a **`paymentAction`** with a value of 'Order' is sent, then PayPal will store the transaction but delay authorising it until instructed. To instruct PayPal to authorise the transaction, a further management request can be sent to the Gateway with an **`action`** of 'AUTHORISE' and the **`xref`** of the transaction to authorise. Alternatively, the AUTHORISE command can be selected in the Merchant Management System (MMS). The transaction will be left in the 'received' state.

21.7.2 Authorise

If no **`paymentAction`** is specified or a **`paymentAction`** with a value of 'Authorize' is sent, then PayPal will authorise the transaction on receipt as per a standard card transaction and you can capture it later if you used the **`captureDelay`** field. *Note that the value uses the PayPal spelling 'Authorize', and not the British spelling 'Authorise'.*

For the first three days (by default) of the authorisation, funds are reserved. This is known as the honour period. After the honour period, captures can still be attempted, but may be returned with insufficient funds.

Authorisations have a fixed expiry period of 29 days.

21.7.3 Sale

If a **`paymentAction`** with a value of 'Sale' is sent, then PayPal will immediately capture the transaction after authorisation. The transaction will be regarded as having been settled and you will not be able to capture it manually and it will not be sent for settlement that evening. The transaction will be left in either the **accepted** or **rejected** terminal states depending on whether PayPal accepted or rejected the transaction.

21.7.4 Capture

Transactions that have been authorised by PayPal and not immediately settled due to a **`paymentAction`** of 'Sale' will be able to be captured as normal.

Captures are sent to PayPal immediately and the PayPal response and the transaction will be left in either the **accepted** or **rejected** terminal state depending on whether PayPal accepted or rejected the capture request.

There is no need to wait for the nightly settlement batch to run as with normal card transactions. This means that it is not possible to change the amount to be captured or cancel the transaction one a capture has been requested.

Note: PayPal allows multiple captures where they sum the individual capture amounts – ie in a different manner from the Gateway's, where only a single capture operation can be processed.

21.7.5 Refund

PayPal transactions can be refunded in the same way as normal card transactions. However, in the same way as capture requests, these will be sent to PayPal immediately and not batched up to be sent as part of the nightly settlement process. This means that the transaction will be left in either the **accepted** or **rejected** terminal state, depending on whether PayPal accepted or rejected the refund request.

Refunds can be made for full or partial amounts, with multiple refunds allowed up to the original authorised amount.

By default, PayPal allows a Merchant up to 60 days from the original authorised transaction date to perform refunds.

21.7.6 Cancel

You should cancel any transactions that you do not wish to capture in order to prevent 'pending' transactions on the Customer's PayPal account.

Authorisations should be cancelled when an initial authorisation was created to confirm the validity of funds during checkout, but the goods will not ship for a significant amount of time (>29 days). Cancelling the transaction will mean that you will have to contact the Customer for an alternative payment method.

All transactions must be completed by being captured or cancelled.

21.7.7 Pending Payments

PayPal may put a transaction into a pending state when flagged for additional fraud review. This state is known to PayPal as payment review or IPR.

IPR transactions will be automatically cancelled by the Gateway and treated as referred transactions with a **`responseCode`** of **2** and a **`responseMessage`** indicating the reason that the transaction was put into a pending state. Unlike card referred transactions, an authorisation code cannot be obtained from PayPal verbally and then the transaction resent.

21.8 Reference Transactions

PayPal does not allow ad hoc Credentials on File (COF) type repeat or recurring transactions using the **`xref`** of a reference transaction unless that transaction has specifically started a PayPal Billing Agreement.

If you want to be able to make future repeat or recurring transactions, then the initial transaction must include the **`billingType`** and **`billingAgreementDescription`** options in the **`checkoutOptions`** so as to identify this transaction as the start of a recurring billing sequence.

This will cause the Gateway to request PayPal to set up a Billing Agreement between you and the Customer. In this case, the PayPal Billing Agreement ID will be returned as part of the **`checkoutDetails`** and displayed on the Merchant Management System (MMS) as part of the payment details, so that you can easily see which PayPal transactions can be used for recurring billing.

22 Amazon Pay Transaction

22.1 Background

Amazon Pay is an additional payment method that is available to all Merchants using the Gateway that have an Amazon Pay account.

To use Amazon Pay, you will be supplied with a separate Amazon Pay Merchant account that can be grouped with your main Merchant account using the account mapping facility as documented in appendix A-6. This allows transactions to be sent using your main Merchant Account and then routed automatically to the Amazon Pay Merchant Account in the same mapping group.

It allows you to offer payment via Amazon Pay in addition to normal card payments.

Amazon Pay transactions will appear in the Merchant Management System (MMS) alongside any card payments and can be captured, cancelled and refunded in the same way as card payments.

Amazon Pay transactions can also be used for recurring billing but require you to indicate in the initial transaction that it will be the basis for recurring billing and a billing agreement will be entered into between your Customer and Amazon Pay when they agree to the payment.

Amazon Pay transactions cannot be used for ad-hoc Credentials on File (COF) repeat transactions unless a billing agreement has been set up.

For more information on how to accept Amazon Pay transactions, please contact customer support.

Amazon Pay is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

22.2 Benefits and Limitations

22.2.1 Benefits

- Provides your customers with the flexibility of paying using their Amazon Pay account when this is more suitable to them.
- The Amazon Pay Checkout can be added as an overlay on the standard checkout to help improve conversion rates with an easier way to pay without customers leaving your website.
- There are no extra costs to add an Amazon Pay Merchant Account. However, you will still be liable for the Amazon Pay transaction fees.
- The full Amazon Pay transaction information is available and returned as part of the transaction.
- Transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

22.2.2 Limitations

- You will need an Amazon Pay account.
- Recurring transactions are not supported unless part of a prearranged billing agreement.
- Independent refunds that are not tied to a previous sale transaction are not supported without prior agreement.
- Transactions require a browser in order to display the Amazon Pay Checkout widgets.

22.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** that is part of a routing group containing an Amazon Pay Merchant, then the Hosted Payment Page will display an Amazon Pay payment button which, when clicked, will open the Amazon Pay Checkout and allow the Customer to pay using their Amazon Pay account.

To customise the Amazon Pay Checkout experience, you may send various options in the **`amazonPayCheckoutOptions`** field in your initial request.

Additional information available from Amazon Pay will be made available in the **`checkoutDetails`** response field.

Note: Custom Hosted Payment Pages might not support the displaying of the Amazon Pay Checkout button. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

22.4 Direct Implementation

Amazon Pay transactions require you to display an Amazon Pay Checkout to your Customer as part of the transaction flow. The transaction must be done in two stages, with the Checkout page being displayed between the stages. They can also optionally be done in three stages, allowing you to display an order confirmation after the Checkout page and before authorising the transaction. You can change the amount at this stage to allow for shipping costs when you know the confirmed delivery address the Customer selected as part of the Amazon Pay Checkout.

Amazon Pay do not provide a ready built Checkout page and require you to create one on your servers using the JavaScript widget toolkit they provide.

Amazon Pay supports the normal payment and management actions. This section explains how to make payment requests. Management requests are performed as detailed in section 3.

To customise the Amazon Pay Checkout experience, you may send various options in the **`checkoutOptions`** field in your initial request.

Additional information available from the Amazon Pay Checkout will be made available in the **`checkoutDetails`** response field.

The direct integration uses two complex fields to pass data between the Amazon Pay JavaScript widgets and the and the Gateway. The **`checkoutRequest`** field will be provided by the Gateway and is a record whose name/value properties should be used to help initialise the widgets. The corresponding **`checkoutResponse`** field should be returned to the Gateway and must be a record containing name/value properties received from the widgets.

The contents of the **`checkoutOptions`**, **`checkoutDetails`**, **`checkoutRequest`** and **`checkoutResponse`** fields are formatted using the *record* format detailed in section 1.7.8, the **`checkoutOptions`** field also supports being provided using the *serialised record* format.

22.4.1 Initial Request (Checkout Preparation)

To request that a transaction be processed via Amazon Pay, the request must contain a **`paymentMethod`** of 'amazonpay' In addition, you may send **`checkoutOptions`** to customise the Checkout experience. When the Gateway receives this **`paymentMethod`**, assuming there are no other errors with the request, it will attempt to find a suitable Amazon Pay Merchant Account in the current account mapping group (refer to appendix A-6).

If the Gateway is unable to find a suitable account, then the transaction will be aborted, and it will respond with a **`responseCode`** of **66364 (INVALID PAYMENTMETHOD)**.

Otherwise, the Gateway will respond with a **`responseCode`** of **65826 (CHECKOUT REQUIRED)** and the response will include a **`checkoutURL`** field containing the URL required to load the Amazon Pay JavaScript Widgets; and a **`checkoutRequest`** containing any data required by those Widgets. The response will also contain a unique **`checkoutRef`** that must be echoed back in the continuation requests.

At this point, your server must create an Amazon Pay Checkout page using their JavaScript Widgets. Further details on how to use the Widgets are provided in the Amazon Pay guide at https://developer.amazonpay.com/docs/classic/express-checkout/in-context/enable_in_context_checkout/.

22.4.2 Continuation Request (Checkout Details and Authorise)

On completion of the Amazon Pay Widgets, the Merchant should send the information created by the Widgets to the Gateway together with a **`status`** value. If the Checkout was successful, the status will be 'success'; alternatively, if the Checkout was cancelled, the status will be 'cancel'. Any **`accessToken`** generated by the Amazon Pay Login Widget; **`orderReferenceID`**, generated by the Wallet or Address Widgets; and **`billingAgreementID`** generated by the optional Billing Widget, must be added to the **`checkoutResponse`** field and sent in a new request to the Gateway. The **`checkoutResponse`** field can be sent either as a URL query string; as a JSON encoded string; or as an array of parameters. This new request will load the Checkout details, including any purchaser and delivery address details as required, and send the transaction to Amazon Pay for authorisation, returning the result as in the case of a normal authorisation transaction. The new request must contain the **`checkoutRef`** received in the initial response.

22.4.3 Separate Checkout Details and Authorisation Requests

You can choose to obtain the Checkout details before actually sending the transaction for authorisation by sending the **`checkoutOnly`** field in the above continuation request. If this field is sent with a value of 'Y' then the Gateway will load the Checkout details and then return them to you without sending the request for authorisation. You can then display them and/or adjust the amount, for example, according to delivery charges appropriate to the received delivery address. You should then send a new request containing the **`checkoutRef`** received to continue the transaction and authorise it.

Note: this stage can be repeated multiple times by including the **`checkoutOnly`** field with a value of 'Y' each time. To complete the transaction, the final request must not contain the **`checkoutOnly`** field or it must not have a value of 'Y'.

22.5 Request Fields

22.5.1 Initial Request (Hosted and Direct Integration)

These fields should be sent in addition to basic request fields in section 2.1 excluding any card details.

Field Name	Mandatory?	Description
paymentMethod	Yes[1]	Must contain the value 'amazonpay' in lower case letters only.
checkoutRedirectURL	No[2]	Reserved for future use.
checkoutOptions	No[3]	Record containing options used to customise the Amazon Pay Checkout. Refer to section 22.5.3 for values.
amazonPayCheckoutOptions	No[4]	Record containing options used to customise the Amazon Pay Checkout. Refer to section 22.5.3 for values.

[1] Optional for Hosted Integration
[2] Not required for Hosted Integration.
[3] Direct Integration Only
[4] Hosted Integration Only

22.5.2 Continuation Request (Direct Integration)

These fields may be sent alone[1].

Field Name	Mandatory?	Description
checkoutRef	Yes	Unique reference return in the initial response.
checkoutResponse	Yes	The data received from the Amazon Pay Checkout Widgets together with a status value.
checkoutOnly	No	Pass **Y** to complete the processing as far as the next Checkout stage and then return with the loaded Checkout details.

[1] It is only necessary to send the **checkoutRef** and the **checkoutResponse** in the continuation request because the **checkoutRef** will identify the Merchant Account and initial request. The message does not have to be signed. You can send any of the normal request fields to modify or supplement the initial request – however, in this case the request should be signed. The **checkoutRedirectURL** and **checkoutOptions** fields sent in the initial request cannot be modified and any sent in the second request must match those used in the first request, or the second request will fail with a **responseCode** of **64442 (REQUEST MISMATCH)**.

22.5.3 Checkout Options (Hosted and Direct Integration)

The following options may be sent in the **`amazonPayCheckoutOptions`** Hosted Integration field or the **`checkoutOptions`** Direct Integration field to customise the Amazon Pay Checkout. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Sub-Field Name	Description
`billingAgreementRequired`	Can be used to specify that a billing agreement must be started. Alternatively, the **`rtAgreementType`** standard integration field can be used with a value of 'recurring' or 'instalment'.
`shippingAddressRequired`	Indication that the shipping address is required, and the Address Checkout Widget will be used.
`sellerOrderID`	The Merchant specified identifier for this order. If not sent, then any value in the **`merchantOrderRef`** standard integration field is used.
`sellerNote`	Represents a description of the order that is displayed in emails to the buyer.
`sellerAuthorizationNote`	A description for the authorisation transaction that is shown in emails to the buyer.
`sellerCaptureNote`	A description for the capture that is displayed in emails to the buyer.
`sellerBillingAgreementID`	The Merchant specified identifier for this billing agreement. If not sent, then any value in the **`rtPolicyRef`** standard integration field is used.
`customInformation`	Any additional information that you want to include with this order reference
`supplementaryData`	Supplementary data.
`softDescriptor`	The description to be shown on the buyer's payment statement
`billingAgreementRequired`	Can be used to specify that a billing agreement must be started. Alternatively, the **`rtAgreementType`** standard integration field can be used with a value of 'recurring' or 'instalment'.
`shippingAddressRequired`	Indication that the shipping address is required, and the Address Checkout Widget will be used.

For further information on the options refer to the Amazon Pay API Reference Guide:
https://pay.amazon.com/us/developer/documentation/apireference/201751630

22.5.4 Response Fields

22.5.5 Initial Response (Direct Integration)

These fields will be returned in addition to the request fields from section 21.5.1 and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
checkoutRef	Yes	Unique reference required to continue this transaction when the Amazon Pay Checkout has completed.
checkoutName	Yes	Unique name of the Checkout. For Amazon Pay this is the value **amazonpay**.
checkoutURL	Yes	URL required to load the Amazon Pay JavaScript Widgets.
checkoutRequest	No	Record containing data required for the Amazon Pay Widgets such as: • **merchantID** – Amazon Pay merchant id • **clientID** – Amazon Pay client id • **sandbox** – true if Amazon Pay sandbox • **region** – Amazon Pay API region code • **scope** – Login Widget scope parameter
checkoutOptions	No	Record containing any Checkout options passed in the request.
acquirerResponseDetails	Yes	Record containing details about the Amazon Pay response containing any error messages and codes. This can be used together with the normal **responseCode** and **responseMessage** response fields to further determine the reason for any failure.

22.5.6 Continuation Response (Direct Integration)

These fields will be returned in addition to the request fields from section 21.5.2, the initial response fields in section 21.6.1 and the basic response fields in section 2.2 minus any card details.

Field Name	Mandatory?	Description
checkoutRef	Yes	Provided if **checkoutOnly** was used in the continuation response to indicate that a further request will be sent to finalise the transaction.
checkoutName	Yes	Unique name of the Checkout. For Amazon Pay this is the value **amazonpay**.
checkoutDetails	Yes	Record containing values made available by the Amazon Pay Checkout. Refer to section 22.5.7 for values.
customer*XXXX*	No[1]	Customer details if provided by the Amazon Pay Checkout as documented in section 17.1
delivery*XXXX*	No[1]	Delivery details if provided by the Amazon Pay Checkout as documented in section 17.4
receiver*XXXX*	No	Buyer details if provided by Amazon Pay as documented in section 17.5. Amazon Pay will usually provide the buyer's name, postcode and email only, which are returned in the **receiverName**, **receiverPostcode** and **receiverEmail** fields accordingly
acquirerResponseDetails	Yes	Record containing details about the Amazon Pay response containing any error messages and codes. This can be used together with the normal **responseCode** and **responseMessage** response fields to further determine the reason for any failure.

[1] The response will include Customer/billing address and delivery address details if provided by the Amazon Pay Checkout.

22.5.7 Checkout Details (Hosted and Direct Integration)

The **`checkoutDetails`** field included in the response above will contain the following values and any further values received from Amazon Pay allowing the Merchant to see the full Amazon Pay order information. The details will be returned using the *record* format detailed in section 1.7.8

Sub-Field Name	Mandatory?	Description
`referenceID`	No	Amazon Pay reference id. Either the **`orderReferenceID`** or the **`billingReferenceID`** where appropriate.
`accessToken`	No	Amazon Pay order reference id as sent in the continuation request **`checkoutResponse`** data.
`bilingAgreementID`	No	Amazon Pay order reference id as sent in the continuation request **`checkoutResponse`** data.
`orderReferenceID`	No	Amazon Pay order reference id as sent in the continuation request **`checkoutResponse`** data.

22.6 Transaction Lifecycle

Amazon Pay transactions will use the normal Authorise, Capture life cycle as documented in appendix A-13.1 with the following differences.

22.6.1 Capture

Captures made by the Direct Integration or Merchant Management System (MMS) are sent to Amazon Pay immediately the transaction will be left in either the **accepted** or **rejected** terminal state depending on whether Amazon Pay accepted or rejected the capture request. Unlike card payments captures do not flag the transaction to be included in the nightly settlement batch and therefore, when done they cannot be redone. This means that it is not possible to change the amount to be captured or cancel the transaction when a capture has been requested.

Captures that are not explicitly performed such as normal transactions or those with a captureDelay are still done as part of the nightly settlement batch.

Transactions that are not captured within 3 days will be placed in a pending state in the Amazon Pay system which is reflected as the **tendered** state in the Gateway and will show on the Merchant Management System as being settled.

22.6.2 Refund Sale

Amazon Pay transactions can be refunded the same as normal card transactions however, like capture requests, these will be sent to Amazon Pay immediately and not batched up and sent as part of the nightly settlement process. This means the transaction will be left in either the **accepted** or **rejected** terminal state depending on whether Amazon Pay accepted or rejected the refund request.

Refunds can be made for full or partial amounts, with multiple refunds allowed up to the original authorised amount.

22.7 Reference Transactions

Amazon Pay does not allow ad hoc Credentials on File (COF) type repeat or recurring transactions using the **`xref`** of a reference transaction unless that transaction has specifically started an Amazon Pay Billing Agreement.

If you want to be able to make future repeat or recurring transactions, then the initial transaction must include an **`rtAgreementType`** of **recurring** or **instalment**. Alternatively, the **`billingAgreementRequired`** option can be included in the **`checkoutOptions`** so as to identify this transaction as the start of a recurring billing sequence.

This will cause the Gateway to request Amazon Pay setup a Billing Agreement between you and the Customer. In this case the Amazon Pay Billing Consent Widget must be used in the Checkout and the **`billingAgreementID`** it creates sent in the **`checkoutResponse`** data in the continuation request. Any billing agreement id will be displayed on the Merchant Management System (MMS) as part of the payment details so that you can easily see which Amazon Pay transactions can be used for recurring billing.

23 PPRO Transactions

23.1 Background

PPRO is an additional payment method that is available to all Merchants using the Gateway that have an PPRO account.

To use PPRO you will be supplied with a separate PPRO Merchant account that can be grouped with your main Merchant Account using the account mapping facility as documented in appendix A-6. This allows transactions to be sent using your main Merchant Account and then routed automatically to the PPRO Merchant Account in the same mapping group.

PPRO is an Acquirer that offers many Alternative Payment Methods (APM), that you can then offer to your Customers.

E-wallets, SMS payments and PSP services are some of the many payment methods PPRO support (eg Alipay, EasyPay, Bancontact). This could allow a business to facilitate overseas transactions or alternative payment methods using a different payment method suitable for that country or business plan.

All transactions created with this payment method will appear in the Merchant Management System (MMS) together with the payment method that was used to process the transaction.

PPRO transactions cannot be used for ad-hoc Credential on File (COF) repeat transactions or for recurring billing.

For more information on how to accept PPRO transactions please contact customer support.

PPRO is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

23.2 Benefits and Limitations

23.2.1 Benefits

- Multiple alternative payment methods could be used.
- Expands range of payment methods for international use.
- Supports a variety of e-wallets, SMS and PSP's.
- Ability to perform refunds on supported payment methods.
- Transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

23.2.2 Limitations

- You will need a PPRO account.
- Payment authorisation is not always instantaneous and may require additional 'QUERY' requests.
- An alternative payment method may only support one or a limited set of currencies or countries.
- Alternative payment methods require a browser in order to display their Checkout.

23.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** which is part of a routing group containing a PPRO Merchant Account, then the Hosted Payment Page will show alternative payment method buttons for each payment method listed in the **`allowedPaymentMethods`** field. When clicked on the Hosted Payment Page may request further details from the Customer before opening the APM Checkout allowing the Customer to pay using that APM.

To customise the alternative payment methods checkout experience, you may send various options in the **`pproCheckoutOptions`** field in your initial request.

Additional information available from PPRO will be made available in the **`checkoutDetails`** response field.

Note: Custom Hosted Payment Pages might not support the displaying of the Alternative Payment Methods. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

23.4 Direct Implementation

PPRO transactions require you to display the alternative payment method's Checkout to your Customer as part of the transaction flow. The transaction must be done in two stages with the Checkout being displayed between the stages.

PPRO supports only supports the SALE, REFUND_SALE actions. This section explains how to make payment requests. Management requests are performed as detailed in section 3.

To customise the alternative payment method's Checkout experience, you may send various options in the **`checkoutOptions`** field in your initial request.

Additional information available from the alternative payment method's Checkout will be made available in the **`checkoutDetails`** response field.

The direct integration uses two complex fields to pass data between PPRO and the Gateway. The **`checkoutRequest`** field will be provided by the Gateway and is a record whose name/value properties represent the data provided in the **`checkoutURL`** and is provided for information purposes only. The corresponding **`checkoutResponse`** field should be returned to the Gateway and must be a record containing name/value properties received from the Checkout when it redirects the Cardholder's browser back to the URL provided using the **`checkoutRedirectURL`** on completion.

The contents of the **`checkoutOptions`**, **`checkoutDetails`**, **`checkoutRequest`** and **`checkoutResponse`** fields are formatted using the *record* format detailed in section 1.7.8, the **`checkoutOptions`** field also supports being provided using the *serialised record* format.

23.4.1 Payment Request

To request that a transaction be processed via PPRO the request must contain a **`paymentMethod`** of 'ppro.*XXXX*', where *XXXX* is the PPRO payment method tag listed in section 23.4.3 below. The request must also have a **`checkoutRedirectURL`** containing the URL of a page on your server to return to when the alternative payment method's Checkout is closed. In addition, you may send **`checkoutOptions`** to provide further custom fields requited by the alternative payment method as details in section 23.4.2 below.

When the Gateway receives these fields, assuming there are no other errors with the request, it will attempt to find a suitable PPRO Merchant Account in the current account mapping group (Refer to appendix A-6).

If the Gateway is unable to find a suitable account, then the transaction will be aborted, and it will respond with a **`responseCode`** of **66364 (INVALID PAYMENTMETHOD)**.

Otherwise, the Gateway will respond with a **`responseCode`** of **65826 (CHECKOUT REQUIRED)** and included in the response will be a **`checkoutURL`** field containing the URL that the buyer's browser should be redirected to in order to complete the payment. The response will also contain a unique **`checkoutRef`** which must be echoed back in the continuation requests.

On completion of the third-party payment the browser will be directed to the **`checkoutRedirectURL`** you provided, complete with information about the payment in a HTTP POST request. The posted data will contain a **`checkoutResponse`** field that will contain any specific response data for the payment method.

23.4.2 Payment Specific Fields

Most of the information required by the alternative payment methods can be supplied using the standard Gateway request fields. However, there may be specific mandatory fields required by a payment method which are not available using the standard fields. In these cases, these fields can be sent in the **`checkoutOptions`** data, the value of which must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

For example, most European services may require the **`nationalid`** and **`consumerref`** fields.

Recurring transactions will require the use of **`iban`** (optionally **`sequencetype`**) and in follow-up payments; **`mandatereference`**, **`mandatesignaturedate`**, and **`sequencetype`**.

These fields are documented in your PPRO integration guide as **SPECIN** fields.

Customer support will be able to guide you on any mandatory options as you will find the transaction will fail with a **`responseCode`** of **65550 (PROCESSOR_ERROR -** Invalid request data) if any are missing.

23.4.3 Payment Method Tags

To specify which alternative payment method is required you need to send the **`paymentMethod`** field with a value using the format is 'ppro.*XXXX*', where *XXXX* is the alternative payment method's tag name as assigned by PPRO.

For example, to use the alternative payment method AstroPay Card that has a tag name of "astropaycard" (all lowercase); the resulting payment method code would be "ppro.astropaycard". This allows the Gateway to know that you're attempting to use AstroPay Card using the PPRO payment method.

The table below is a guide to the tag names available. This list is fluid as PPRO add and remove methods.

If you know of a payment method that is not on this list or the payment method cannot be used; please contact customer support for advice.

Payment Method Tag	Payment Method Name
affinbank	Affin bank
alipay	AliPay
ambank	AmBank

Payment Method Tag	Payment Method Name
argencard	Argencard
astropaycard	AstroPay Card
astropaydirect	AstroPay Direct
aura	Aura
baloto	Baloto
banamex	Banamex
bancodobrasil	Banco do Brasil
bancodechile	Banco de Chile
bancodeoccidente	Banco de Occidente
bancomer	Bancomer
bankislam	Bank Islam
bcmc	Bancontact
bitpay	Bitpay
boleto	Boleto Bancario
bradesco	Bradesco
cabal	Cabal
cartaomercadolivre	Cartao Mercado Livre
carulla	Carulla
ccauth	Credit/Debit Card
ccweb	Credit/Debit Card
cencosud	Cencosud
cimbclicks	CIMB Clicks
cmr	CMR
davivienda	Davivienda
directpay	Sofortüberweisung (Direct Pay)
dragonpay	Dragonpay
easypay	EasyPay
efecty	Efecty

Payment Method Tag	Payment Method Name
elo	Elo
empresedeenergia	Emprese de Energia
enets	eNETS
entercash	Entercash
eps	EPS
estonianbanks	Estonian Banks
giropay	Giropay
hipercard	Hipercard
hongleongbank	Hong Leong Bank
ideal	iDEAL
instanttransfer	Instant Transfer
int_payout	International Pay-Outs
itau	Itau
latvianbanks	Latvian Banks
lithuanianbanks	Lithuanian Banks
magna	Magna
maxima	Maxima
maybanktwou	Maybank2u
multibanco	Multibanco
mybank	MyBank
myclearfpx	MyClear FPX
naranja	Naranja
narvesen	Narvesen
nativa	Nativa
oxxo	OXXO
p24	Przelewy24
p24payout	Przelewy24 Payout
pagofacil	Pago Facil

Payment Method Tag	Payment Method Name
paypost	PayPost
paysafecard	Paysafe Card
paysbuy	Paysbuy
paysera	Paysera
payu	PayU
perlas	Perlas Terminals
poli	OLI
presto	Presto
pse	PSE
pugglepay	Pugglepay
qiwi	QIWI
qiwipayout	QIWI Payout
rapipago	Rapipago
redpagos	Redpagos
rhbbank	RHB Bank
safetypay	SafetyPay
santander	Santander
sepadirectdebit	SEPA DirectDebit
sepapayout	SEPA Payout
seveneleven	Seveneleven (7eleven)
singpost	SingPost
skrill	Skrill
surtimax	Surtimax
tarjetashopping	Tarjeta Shopping
trustly	Trustly
trustpay	TrustPay
unionpay	UnionPay
verkkopankki	Verkkopankki – Finish Online Banking

Payment Method Tag	Payment Method Name
webpay	Webpay
yellowpay	Yellow Pay

23.5 Request Fields

23.5.1 Initial Request (Hosted and Direct Integration)

These fields should be sent in addition to the basic request fields in section 2.1 excluding any card details.

Field Name	Mandatory?	Description
paymentMethod	Yes[1]	Payment method to be used with PPRO (eg **ppro.astropay**, **ppro.alipay**, etc.).
checkoutRedirectURL	Yes[2]	URL on Merchant's server to return to when the Alternative Payment Method's Checkout is closed.
checkoutOptions	No[3,4]	Record containing options used to customise the alternative payment methods Checkout. Refer to section 23.5.2 for values.
pproCheckoutOptions	No[5,4]	Record containing options used to customise the alternative payment methods Checkout. Refer to section 23.5.2 for values.

[1] Optional in Hosted Integration.
[2] Not required for Hosted Integration.
[3] Direct Integration only.
[4] Whilst the Gateway does not see this field as mandatory, PPRO may have payment methods that require additional configuration using checkout options.
[5] Hosted Integration Only

23.5.2 Checkout Options (Hosted and Direct Integration)

The following options may be sent in the **`pproCheckoutOptions`** Hosted Integration field or the **`checkoutOptions`** Direct Integration field to customise the Checkout. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Sub-Field Name	Description
`nationalid`	Consumer's national ID (up to 30 characters).
`consumerref`	Unique reference identifying the consumer within 1 to 20 characters and a format of A-Za-z0-9.%,&/+*$-
`siteid`	Unique site identifier. Required for clients serving multiple points of sale and forwarded onwards whilst using the qiwi payment method.
`iban`	Valid IBAN of consumer/destination account.
`sequencetype`	Sequence type of the direct debit. Possible values are: **oneOff** – The direct debit is executed once (***default***) **first** – First direct debit in a series of recurring ones
`mandatereference`	Mandate reference as returned on the first transaction in the sequence (found from mandatereference in checkoutDetails)
`mandatesignaturedate`	Date of the initial transaction.
`bic`	Valid BIC (8 or 11 alphanumeric letters) – optionally supplied to skip the bank selection page (by using the bank referenced by BIC as supplied)
`clientip`	Optional IP address of the consumer during checkout using Trustly (127.0.0.1 is not allowed!)
`address`	Customer's billing address[1]
`city`	Customer's billing city[1]
`phone`	Customer's phone[1]
`mobilephone`	Customers mobile phone[1]
`dob`	MCC 6012 Date of Birth[1]
`dynamicdescriptor`	Statement narrative[1]

[1] This information is supplied to PPRO by default using the following fields: `customerAddress`, `customerPostcode`, `customerTown`, `customerEmail`, `customerPhone`, `customerMobile`, `receiverDateOfBirth`, `statementNarrative1`.

23.6 Response Fields

23.6.1 Initial Response (Direct Integration)

The fields below will be returned in addition to the basic response fields in section 2.2 for the start of a PPRO transaction and the PPRO checkout process.

Field Name	Mandatory?	Description
checkoutRef	Yes	Unique reference required to continue this transaction when the PPRO Checkout has completed.
checkoutName	Yes	The **paymentMethod** you used to identify the PPRO payment method.
checkoutRedirectURL	Yes	The URL to redirect the Customer to, to start the checkout process.
checkoutOptions	Yes	Record containing any Checkout options provided in the request.
checkoutDetails	Yes	Record containing additional information provided from the payment method used during checkout.
checkoutRef	Yes	The unique reference required to continue the transaction when PPRO checkout is complete.
checkoutRequest	Yes	Record containing the redirect secret, checksum and request status.

23.6.2 Completion Response (Hosted and Direct Integration)

Fields from the initial response in the previous section may be present as well as the fields below and will not contain any card details.

Field name	Mandatory?	Description
checkoutResponse	Yes	Containing additional information provided by the Checkout. Any change in the payment's status will be given in **responseMessage** and **responseCode**[1]
checkoutStatus	Yes	A string containing the result of the checkout process. **This is not used to identify the transaction's payment status.**

[1] Not all payment methods give an immediate payment status. This will require a further QUERY to the Gateway to see whether this value has changed to a status of 'tendered'.

23.6.3 Notifications and “Tendered” Payments

Whilst some payment methods give an immediate payment status (ie direct card payment methods rather than SMS and e-wallet systems), some payments may come back with the status of ‘tendered’. At this time, online shopping modules will not be able to monitor the transaction status. The use of a QUERY request may be of use as seen in section 1.9.8. Please ask customer support in this matter who will be able to give more information and may be able to provide better advice for your situation.

Notifications from PPRO regarding the payment status, seconds, minutes or hours after the checkout will automatically update the transaction status.

24 Pay by Bank app (PBBA) Transactions

24.1 Background

Pay by Bank app (PBBA) is an additional payment method that is available to all Merchants using the Gateway that have a Pay by Bank app account and an OBN Global Acquiring account.

PBBA, formerly known as Zapp, is an innovative, secure, and fully digital payment option in the UK, built and operated by VocaLink, a Mastercard company. It allows your Customers to pay, in real time, using the banking app on their phone. It's designed to make online checkout easier, whilst using all the security of their bank. Payments work through secure digital tokens, meaning your Customers never reveal any of their financial details when they are shopping.

To use PBBA you will be supplied with a separate PBBA Merchant account that can be grouped with your main Merchant Account using the account mapping facility as documented in appendix A-6. This allows transactions to be sent using your main Merchant Account and then routed automatically to the PBBA Merchant Account in the same mapping group.

All transactions created with this payment method will appear in the Merchant Management System (MMS) together with the payment method that was used to process the transaction.

PBBA transactions cannot be used for ad-hoc Credentials on File (COF) repeat transactions or for recurring billing.

For more information on how to accept PBBA transactions please contact customer support.

Pay by Bank app (PBBA) transactions are only available with an OBN Global merchant account. This payment method is pre-release and these integration details maybe subject to change before general release.

PBBA is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration.

24.2 Benefits and Limitations

24.2.1 Benefits

- Increased conversion rate through a simple, quick payment process.
- Secure payment processing in real time.
- Lower transaction cost compared to card payments.
- Minimal fraud risk thanks to ‘Request to Pay’ technology.
- Quick and convenient processing of refunds and disputes.
- Integration on websites and in mobile apps.
- Transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

24.2.2 Limitations

- You will need a PBBA account and OBN Acquiring account.
- Recurring transactions are not supported.
- Independent refunds are not supported.
- Transactions require a browser or mobile device in order to display the PBBA Checkout.
- Your Customer will need a mobile device in order to display their Banking app.
- Payment authorisation is not instantaneous and will require additional ‘QUERY’ requests.
- Only available in the UK.
- Transactions require a browser or mobile device in order to display the PBBA Checkout.
- Customers require a mobile device in order to display their Banking app.

24.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** that is part of a routing group containing an PBBA Merchant, then the Hosted Payment Page will display a Pay by Bank App payment button which, when clicked, will display a Basket Reference Number (BRN) which the Customer can enter into their mobile banking application to complete the payment. When payment has been completed the Hosted Payment Page will display a payment confirmation page in the usual manner.

Only SALE transactions are supported.

Additional information available about the PBBA transaction will be made available in the **`checkoutDetails`** response field.

Note: Custom Hosted Payment Pages might not support the displaying of the Pay by Bank App payment button. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

24.4 Direct Implementation

PBBA transactions are designed to be integrated with and invoked by the Merchant Button Integration Library code made available by Mastercard for embedding the PBBA payment button on your website or mobile application[1].

PBBA supports only supports the SALE, REFUND_SALE, CANCEL and CAPTURE actions. This section explains how to make payment requests. Management requests are performed as detailed in section 3.

24.4.1 Payment Request

To request that a transaction be processed via PBBA, the request must contain a **`paymentMethod`** of 'pbba'. In addition, you must send **`checkoutOptions`** to provide information about the Customer's device. When the Gateway receives these two fields, assuming there are no other errors with the request, it will attempt to find a suitable PBBA Merchant Account in the current account mapping group (refer to appendix A-6).

If the Gateway is unable to find a suitable account, then the transaction will be aborted, and it will respond with a **`responseCode`** of **66364 (INVALID PAYMENTMETHOD)**.

Otherwise, the Gateway will initiate the PBBA transaction and respond with a **`responseCode`** of **0 (SUCCESS)** and included in the response will be a **`checkoutDetails`** field containing the Basket Reference Number (BRN) that should be displayed to the Customer.

Please note that the payment has not completed until the Customer enters the BRN into their mobile banking application and confirms the payment. The response will contain a **`state`** of received until the Gateway has received confirmation from the banking network that the payment has been confirmed by the Customer.

To determine when payment has completed you may periodically poll the Gateway using a QUERY request to check the **`state`** of the transaction. For more information on the QUERY request, please refer to section 1.9.8

[1] The current PBBA Merchant Integration Libraries are available from Mastercard.

24.5 Request Fields

24.5.1 Payment Request (Hosted and Direct Integration)

These fields should be sent in addition to the basic request fields in section 2.1 excluding any card details.

Field Name	Mandatory?	Description
`paymentMethod`	Yes[1]	Payment method to be used. Must be **pbba**.
`checkoutOptions`	No[2,3]	Record containing options used to customise the PBBA Checkout. Refer to section 24.5.2 for values.

[1] Optional in Hosted Integration.
[2] Direct Integration only.
[3] Whilst the Gateway does not see this field as mandatory, PBBA mandates that certain options are provided.

24.5.2 Checkout Options (Direct Integration)

The following options must be sent in the **`checkoutOptions`** Direct Integration field to customise the Checkout. *Unlike other payment methods these options are mandatory and must be sent*. The options must be formatted using the *record* or *serialised record* formats detailed in section 1.7.8.

Sub-Field Name	Description
`device.type`	Consumer device type. This field is mandatory and has no default value. Possible values are: **ATM** – ATM device **MOBLPHN** – Mobile phone device **PCLAPTP** – PC or Laptop device **TABLET** – tablet device **OTHERS** – other device type
`device.os`	Consumer device operating system. This field is mandatory and has no default value. Possible values are: **AND** – Android **IOS** – Apple iOS **WIN** – Microsoft Windows **WINMOB** – Microsoft Windows Mobile **OTHERS** – other operating system
`browser.userAgent`	Content of HTTP User-Agent header received from the Consumer's device. Maximum 127 characters.
`browser.timeZone`	Time zone offset in minutes between UTC and the Consumer's device. The offset is positive if the local time zone is behind UTC and negative if it is ahead.
`browser.screenResolution`	Screen resolution of the Consumer's device. Formatted as the height and width separated by a 'x'. The height and width must be between 1 and 9999 pixels.
`browser.acceptEncoding`	Content of HTTP Accept-Encoding header received from the Consumer's device. Maximum of 127 characters.
`browser.acceptLanguage`	Content of HTTP Accept-Encoding header received from the Consumer's device. Maximum of 127 characters.

The **`category.name`** format of the sub-field name indicates that option's value is a record called **`category`** containing a sub-field called **`name`**.

24.6 Response Fields

24.6.1 Payment Response (Hosted Integration)

There are no additional response fields for PBBA. The Hosted Integration will return the basic transaction response fields in section 2.2 minus any card related fields.

24.6.2 Payment Response (Direct Integration)

These fields will be returned in addition to the request fields from section 24.5.1 and the basic response fields in section 2.2 minus any card related fields.

Field Name	Mandatory?	Description
checkoutDetails	Yes	Record containing values made available by the PBBA Checkout. Refer to section 24.6.322.5.7 for values.

24.6.3 Checkout Details (Direct Integration)

The **checkoutDetails** field included in the response above will contain the following values which should be provided to the client-side PBBA Integration Library. The details will be returned using the *record* format detailed in section 1.7.8

Sub-Field Name	Mandatory?	Description
transactionID	Yes	Transaction identifier.
settlementID	Yes	Settlement retrieval identifier.
pbbaCode	Yes	Basket reference number (BRN).
secureToken	Yes	Secure token for client-size use.
retrievalExpiryInterval	Yes	Retrieval expiry time interval.
confirmationExpiryInterval	Yes	Confirmation expiry time interval.
cookieSentStatus	Yes	Cookie sent status.
payConnectID	No	Reserved for future use.
cookieExpiryDays	No	Reserved for future use.
riskScore	No	Reserved for future use.
bankName	No	Reserved for future use.

24.6.4 Refunds (Direct Integration)

Refund requests require that a reason code be included in either the **`orderRef`** or **`cancelReason`** fields.

The codes are as follows; **DUPLICATEORDER**, **GOODSRETURNED**, **ORDERCANCELLED**, **MERCHANTOUTOFSTOCK**, **GOODSNOTRECV**, **LATECONFIRMATION**, **DISPUTES**.

The code must be in uppercase and must appear as a separate word anywhere in the field's value.

For example, if refunding due to the purchased item being out of stock you could set the value to just "MERCHANTOUTOFSTOCK" or to a more complete description such as "Refund: Green polo shirt, XXL (MERCHANTOUTOFSTOCK)".

This also applies to refunds made by the Merchant Management System (MMS) where the reason code must be entered when prompted for the reason for the refund.

Only the refund code is sent to the PBBA system, any other text is stored by the Gateway for your benefit and viewable in the Merchant Management System (MMS).

25 SecurePlus Transactions

25.1 Background

SecurePlus is available if you have a Planet Merchant Account. It is a secure e-commerce payment solution designed to allow you to accept China UnionPay debit cards via the Planet Acquirer, with a reduced the risk of fraudulent transactions while providing a friction-free payment process for your Customers. SecurePlus divides the online payment process into separate authentication and authorisation transaction flows that authenticates the cardholder's identity before you submit the authorisation request.

To use SecurePlus you will be supplied with a separate Planet Merchant Account that can be grouped with your main Merchant Account using the account mapping facility as documented in appendix A-6. This allows transactions to be sent using your main Merchant Account and then routed automatically to the Planet Merchant Account in the same mapping group.

When UnionPay debit cards are issued, the Cardholder must register their mobile number with the Issuer. The SMS code authentication works at the time of checkout by submitting the payment details to UnionPay together with the Cardholder's mobile number. The Issuer verifies the card and registered mobile number and sends an SMS message to the to the Cardholder's mobile phone containing a unique 6-digit code. The Cardholder then enters this code into an authentication dialog provided by the Gateway so that it can be sent to UnionPay for verification. If approved, the final financial transaction is submitted.

The authentication process is only required for debit cards. To accommodate the use of cards stored in secure online wallet, the card authentication has some built-in flexibility when other compensating controls are employed.

SecurePlus transactions will appear in the Merchant Management System (MMS) alongside any card payments and can be captured, cancelled and refunded in the same way as card payments.

SecurePlus transactions can also be used for recurring billing with the SMS authentication not been required on recurring transactions as long as the initial transaction was successfully SMS authenticated.

.

For more information on how to accept SecurePlus transactions, please contact customer support.

SecurePlus transactions are only available with a Planet Merchant Account.

SecurePlus is supported by the Hosted and Direct Integrations. It is not supported by the Batch Integration apart from recurring billing.

25.2 Benefits and Limitations

25.2.1 Benefits

- Authorisations are available immediately and returned as part of the transaction.
- Provides your customers the flexibility of paying using their UnionPay debit card when this is more suitable to them than using other payment methods.
- Can be implemented with little or no extra integration work if you already support 3-D Secure transactions.
- There are no extra Gateway costs to use SecurePlus. Your Acquirer may charge to add this onto your business account; however, you may also find that your transaction charges are lower as a result of using 3-D Secure.
- Fully configurable within the Merchant Management System (MMS).

25.2.2 Limitations

- You will need a Planet Acquiring, however, such an account can also be used to process all your other card transactions.
- Only UnionPay debit cards are supported, however, UnionPay credit cards can be accepted without the need for SecurePlus.
- You must currently provide the Gateway with the Cardholder's mobile phone number at the time of the transaction. The Hosted Payment Page will prompt the Cardholder for this information if it detects that a UnionPay debit card is being supplied.
- The Cardholder must have access to their registered mobile phone during the payment process.
- Transactions require a browser in order to display the Customer SMS code entry dialog.

25.3 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** which is part of a routing group containing a Planet Merchant, then the Hosted Payment Page will automatically attempt to collect the Cardholder's mobile phone number if they detect that a UnionPay debit card has been entered. It will then display the SecurePlus authentication page to allow the Cardholder to enter the received SMS code.

The SecurePlus authentication page is designed and controlled by the Gateway, but you can change the Merchant name and website address that is displayed on the form by sending the **`merchantName`** and/or **`merchantWebsite`** request fields.

Any **`merchantWebsite`** must be a fully qualified URL containing at least the scheme and host components.

Note: Custom Hosted Payment Pages may not support the collection of the Cardholder's mobile number or the displaying of the SecurePlus authentication page. If you have a custom page which doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

25.4 Direct Implementation

SecurePlus transactions require you to support the 3-D Secure integration as documented in section 5.

In addition, the Cardholder's mobile phone number must be provided in the transaction request.

25.5 Request Fields

These fields should be sent in addition to basic 3-D Secure request fields in section 5.5.1.

Field Name	Mandatory?	Description
`customerMobile`	Yes	Must contain the Cardholder's mobile phone number as registered with the Issuer. The number must be international format with a leading + symbol followed by up to 3 digits then a space then up to 10 more digits.

25.6 Response Fields

There are no additional response fields for SecurePlus other those defined for 3-D Secure in section 5.6.

26 Digital Wallet Transactions

26.1 Background

The Gateway supports payments made using the following Digital Wallets, Apple Pay, Android Pay and Google Pay™. These wallets can be used to enhance mobile purchasing experiences for customers with supported devices and produce a payment token which can be passed to the Gateway instead of the Cardholder's actual card details.

You can use these wallets with any Merchant Account that has been configured to accept them.

For more information on how to accept payment tokens, please contact customer support.

Digital Wallets are currently supported by the Direct Integration only. They are not supported by the Hosted or Batch Integration.

Digital wallet support is not available with all Acquirers and must be enabled on your Merchant Account before they can be used. Please contact support to find out whether your Acquirer supports the and which can be enabled on your Merchant Account.

26.2 Benefits and Limitations

26.2.1 Benefits

- The payment details are stored externally to the Gateway and can be used with any Merchant that supports the appropriate payment tokens.
- Customers can select from previously stored payment details, making the checkout process more streamlined, resulting in fewer abandoned carts and thus increasing sales.
- Compatible with existing card base fraud solutions such as Address Verification Service (AVS), 3-D Secure and third-party fraud providers.
- There are no extra costs to add these payment methods to your Gateway account.
- The transactions are controlled within the Merchant Management System (MMS) in the same manner as normal card transactions.

26.2.2 Limitations

- Your Customer will need a digital wallet enabled device with some stored card details in order to make full use of this payment method.
- The device needs to be integrated with the Gateway using third-party provided software.
- Repeat transactions using the retrieved payment details are supported.

26.3 Configuration

The Merchant Account being used for the payments must be configured with your Digital Wallet credentials so that the Gateway can decrypt the payment token.

26.3.1 Apple Pay configuration

Apple Pay requires the Gateway to generate public/private key pair and then the public key must be shared with your Apple Pay enabled application in the guise of an Apple Pay *payment process certificate*.

To configure an Apple Pay *payment processing certificate* you must have enrolled in the Apple Developer Program and created a unique Apple Pay merchant identifier.

The *payment processing certificate* is associated with your merchant identifier and used to encrypt payment information. The certificate expires every 25 months. If the certificate is revoked, you can recreate it.

You would normally use the Merchant Management System (MMS) to configure your payment processing certificate by following the steps outlined below:

1. Open the Apple Developer Certificates, Identifiers & Profiles webpage and select 'Identifiers' from the sidebar.
2. Under 'Identifiers', select 'Merchant IDs' using the filter in the top-right.
3. On the right, select your merchant identifier.
4. Under 'Apple Pay Payment Processing Certificate', click 'Create Certificate'.
5. Download our certificate signing request (CSR) from the MMS and save to a file.
6. Click 'Choose File' and select the CSR you just downloaded.
7. Click 'Continue'.
8. Click 'Download' to download the *payment processing certificate* and save to a file.
9. Upload the payment processing certificate to the MMS.

26.3.2 Android Pay configuration

Android Pay is currently a deprecated payment method and as such you will need to contact our customer support for information on how to configure this payment method.

26.3.3 Google Pay configuration

Google Pay requires no specific configuration however you must use your Google Merchant Identifier, our Gateway identifier of '**crst**', and the correct Merchant Account identifier when configuring your Google Pay enabled application.

If you have not yet obtained your Google Merchant identifier, please create this before proceeding.

Once created, this will need to be entered in the Google Pay Preferences section in the MMS (found on Digital Wallets tab of the Preferences page).

26.4 Hosted Implementation

If a transaction is sent to the Hosted Integration using a **`merchantID`** that has a Digital Wallet enabled, then the Hosted Payment Page will display a payment button for that wallet which, when clicked, will allow the Customer to pay using cards stored in the wallet.

Please note that Apple Pay is only supported when your Customer is using the Apple Safari web browser.

You cannot send an existing payment token using the Hosted Integration, any **`paymentToken`** field will be ignored if sent.

Note: Custom Hosted Payment Pages might not support the displaying of the payment button. If you have a custom page that doesn't support this, then you would need to contact support to have your Hosted Payment Page upgraded.

26.5 Direct Implementation

Digital Wallet payments require the secure payment token generated by the wallet enabled application to be sent to the Gateway in the **`paymentToken`** field. The type of token must be specified by also sending the **`paymentMethod`** field with a value of '**applepay',** '**androidpay'** or **'googlepay'**.

26.6 Request Fields

These fields should be sent instead of the standard card details together with the fields in section 2.1.

Field Name	Mandatory?	Description
`paymentMethod`	Yes	The type of payment token sent. Value must be one of: **applepay** – to indicate an Apple Pay token **androidpay** – to indicate an Android Pay token **googlepay** – to indicate a Google Pay token
`paymentToken`	Yes	Must contain the secure payment token produced by the wallet enabled application.

26.7 Response Fields

There are no additional response fields.

26.8 Digital Wallet Tokens

Digital Wallet payments operate the same as normal card payments, the main difference is that the card details are passed from the wallet application within an encrypted payment token. Once the Gateway has extracted the card details then it can use it with 3-D Secure and Fraud checking services as normal.

26.8.1 FPAN/DPAN tokens

Apple Pay, Android Pay and Google Pay (Mobile) payment tokens contain an EMV tokenised card number also known as a device-specific number (DPAN) rather than the Cardholder's actual card number (FPAN). With these tokens the expiry date is the date the DPAN expires rather than the value printed on the Cardholder's card. The card mask returned by the Gateway will be the masked DPAN, the Gateway is not able to return the last 4 digits of the FPAN. The card issuing details return should the same as those of the original FPAN.

Google Pay (Web) payment tokens contain the Cardholder's original card number (FPAN) and expiry date. This means that the card mask and expiry date will be those of the original card.

26.8.2 AVS/CV2 Checking

Digital wallet payment tokens do not contain any address or CVV details. The Cardholder's billing address can be passed in the transaction along with the payment token so that address checking can be performed.

The Gateway and Acquirer will not perform CVV checks with these payment tokens effectively disabling CVV checks for the transaction disregarding your preferences.

26.8.3 3-D Secure Authentication

DPAN based tokens will usually contain 3-D Secure data and so the Gateway will send this data to the Acquirer to gain the benefits of an authenticated transaction without the need to challenge the Cardholder. This makes using the digital wallet a much simpler and frictionless method of payment.

FPAN based tokens can passed to the Gateway's 3-D Secure processing and undergo the normal authentication journey as a manually entered card number.

26.8.4 Risk Checking

Both FPAN and DPAN based tokens can be used with risk checking via Kount in the same manner as a normal card transaction.

26.8.5 Transaction Lifecycle and Recurring Transactions

Both FPAN and DPAN based tokens will follow the standard transaction lifecycle and can be cancelled, captured, refunded or used as the basis of subsequent transactions in the same manner as a normal card transaction.

A-1 Response Codes

The Gateway will always issue a numeric **`responseCode`** to report the status of the transaction. These codes should be used rather than the **`responseMessage`** field to determine the outcome of a transaction.

Response codes are grouped; however, the groupings are for informational purposes only and not all codes in a group are used and some codes may exist for completeness or future use.

A zero `responseCode` always indicates a successful outcome.

A-1.1 Authorisation Response Codes

The Gateway uses a set of standard response codes to indicate the status of an authorisation request to the Acquirer. These response codes are based on the 2-character ISO 8583 response codes.

The full set of ISO 8583 codes used are given in the table below, however not all are applicable to transactions currently supported by the Gateway and therefore not used and documented for reference purposes only.

Some ISO-8583 codes are not numeric and therefore to ensure all Gateway response codes are numeric these codes are mapped to an equivalent numeric code greater than 99. This equivalent numeric code is shown in the table below along with the original 2 letter code in brackets.

Not all ISO-8583 codes are applicable to the types of transactions currently available via the Gateway and therefore unapplicable codes, although documented, may not currently be returned.

If the authorising Acquirer does not return a suitable ISO 8583 code, then the Gateway will attempt to map the Acquirers response to a suitable code.

The original Acquirer authorisation response code and response message will always be returned in the **`acquirerResponseCode`** and **`acquirerResponseMessage`** fields as detailed in section 18.

The original Acquirer authorisation response code may not be numeric and information on these codes will need to be requested from the Acquirer.

Acquirer Authorisation Response codes: 0 – 9999	
Code	**Description**
0	Successful approval/completion
1	Refer to card issuer
2	Refer to card issuer, special condition
3	Invalid merchant or service provider
4	Pickup card
5	Do not honor
6	Error
7	Pickup card, special condition (other than lost/stolen card)
8	Honor with identification
9	Request in progress
10	Approval for partial amount
11	Approved VIP
12	Invalid transaction
13	Invalid amount (currency conversion field overflow), or amount exceeds maximum for card program
14	Invalid card number/account number
15	No such issuer
16	Approved, Update Track 3
17	Customer cancellation
18	Customer dispute
19	Re-enter transaction
20	Invalid response/Acquirer error
21	No action taken (unable to back out prior transaction)
22	Suspected malfunction
23	Unacceptable transaction
24	File update impossible
25	Reference number cannot be found. Unable to locate record in file, or account number is missing from the inquiry

Acquirer Authorisation Response codes: 0 – 9999	
Code	**Description**
26	Duplicate reference number
27	Error in reference number
28	File is temporarily unavailable for update
29	File action failed/Contact acquirer
30	Format error
31	Bank not supported by Switch/Unknown acquirer account code
32	Complete partially
33	Pickup card (expired)
34	Pickup card (suspected fraud)
35	Pickup card (acceptor contact acquirer)
36	Pickup card (restricted card)
37	Pickup card (acceptor call acquirer security)
38	Pickup card (PIN tries exceeded)
39	No credit account
40	Function not supported
41	Pickup card (lost card)
42	No universal account
43	Pickup card (stolen card)
44	No investment account
45	Account closed
46	Identification required
47	Identification cross-check required
48	No from account
49	No to account
50	No account
51	Insufficient funds
52	No checking account

Acquirer Authorisation Response codes: 0 – 9999	
Code	**Description**
53	No savings account
54	Expired card
55	Incorrect PIN
56	Unknown card
57	Transaction not permitted to cardholder
58	Transaction not allowed at terminal
59	Suspected fraud
60	Contact acquirer
61	Exceeds withdrawal amount limit
62	Restricted card (for example, in Country Exclusion table)
63	Security violation
64	Amount higher than previous transaction
65	SCA Required (previously, Exceeds withdrawal limit)
66	Contact acquirer
67	Hard capture - ATM
68	Time out
69	Advice received too late
70	Contact card issuer
71	Message flow error
72	Authorization centre not available for 60 seconds.
73	Authorization centre not available for 300 seconds.
74	PIN entry necessary
75	Allowable number of PIN tries exceeded
76	Unable to locate previous message (no match on Retrieval Reference number)
77	Previous message located for a repeat or reversal, but repeat or reversal data are inconsistent with original message
78	Blocked, first used. The transaction is from a new cardholder, and the card has not been unblocked
79	Already reversed

Acquirer Authorisation Response codes: 0 – 9999	
Code	**Description**
80	Visa transactions: credit issuer unavailable. Private label and check acceptance: Invalid date
81	PIN cryptographic error found (error found by VIC security module during PIN decryption)
82	Negative CAM, dCVV, iCVV, or CVV results
83	STIP cannot approve
84	Pre-auth time too great
85	No reason to decline a request for account number verification, address verification, CVV2 verification, or a credit voucher or merchandise return
86	Unable to verify PIN
87	Purchase amount only, no cash back allowed
88	Unable to authorise
89	Ineligible to receive
90	Cut-off in progress
91	Issuer unavailable or switch inoperative (STIP not applicable or available for this transaction)
92	Destination cannot be found for routing
93	Transaction cannot be completed, violation of law
94	Duplicate transaction
95	Reconcile error
96	System malfunction
97	Security Breach
98	Date and time not plausible
99	Error in PAC encryption detected
497 (B1)	Surcharge amount not permitted on Visa cards (U.S. acquirers only)
498 (B2)	Surcharge not supported
928 (N0)	Unable to authorise
931 (N3)	Cash service not available
932 (N4)	Cashback request exceeds issuer limit
933 (N5)	Resubmitted transaction over max days limit
935 (N7)	Decline for CVV2 failure

Acquirer Authorisation Response codes: 0 – 9999	
Code	**Description**
936 (N8)	Transaction amount greater than pre-authorised approved amount
1002 (P2)	Invalid biller information
1005 (P5)	PIN Change/Unblock request declined
1006 (P6)	Unsafe PIN
1037 (Q1)	Card Authentication failed
1072 (R0)	Stop Payment Order
1073 (R1)	Revocation of Authorization Order
1074 (R3)	Revocation of All Authorizations Order
1144 (T0)	Approval, keep first check
1145 (T1)	Check OK, no conversion
1146 (T2)	Invalid RTTN
1147 (T3)	Amount greater than limit
1148 (T4)	Unpaid items, failed NEG
1149 (T5)	Duplicate check number
1150 (T6)	MICR error
1151 (T7)	Too many checks
1298 (XA)	Forward to issuer
1301 (XD)	Forward to issuer
1363 (Z3)	Unable to go online

A-1.2 Gateway Response Codes

The Gateway uses a set of enhanced response codes to indicate if there is an issue with the transaction which prevented any authorisation response being received from the Acquirer. These response codes start at 65536.

The responses are grouped into categories and the codes in the 'missing' and 'invalid' field categories are designed so that that invalid field code is exactly 256 greater than the corresponding missing field code. For example, the code of a missing **`action`** field is **66055** and the corresponding code for an invalid **`action`** field is **66311** (66055 + 256).

General Error Codes: 65536 - 65791	
Code	**Description**
65536	Transaction in progress. Contact customer support if this error occurs.
65537	A general error has occurred.
65538	Reserved for future use. Contact customer support if this error occurs.
65539	Invalid Credentials: **`merchantID`** is unknown or the **`signature`** doesn't match.
65540	Permission denied: caused by sending a request from an unauthorised IP address.
65541	Action not allowed: **`action`** is not supported by the Acquirer or allowed for the transaction.
65542	Request Mismatch: fields sent while completing a request do not match initially requested values. Usually due to sending different card details to those used to authorise the transaction when completing a 3-D Secure transaction or performing a REFUND_SALE transaction.
65543	Request Ambiguous: request could be misinterpreted due to inclusion of mutually exclusive fields.
65544	Request Malformed: could not parse the request data.
65545	Suspended Merchant account.
65546	Currency not supported by Merchant.
65547	Request Ambiguous, both **`taxValue`** and **`discountValue`** provided when should be one only.
65548	Database error.
65549	Payment processor communications error.
65550	Payment processor error.
65551	Internal Gateway communications error.
65552	Internal Gateway error.
65553	Encryption error.
65554	Duplicate request. Refer to appendix A-8.

General Error Codes: 65536 - 65791	
Code	**Description**
65555	Settlement error.
65556	AVS/CV2 Checks are not supported for this card (or Acquirer).
65557	IP Blocked: Request is from a banned IP address.
65558	Primary IP blocked: Request is not from one of the primary IP addresses configured for this Merchant Account.
65559	Secondary IP blocked: Request is not from one of the secondary IP addresses configured for this Merchant Account.
65560	Reserved for future use. Contact customer support if this error occurs.
65561	Unsupported Card Type: Request is for a card type that is not supported on this Merchant Account.
65562	Unsupported Authorisation: External authorisation code **`authorisationCode`** has been supplied and this is not supported for the transaction or by the Acquirer.
65563	Request not supported: The Gateway or Acquirer does not support the request.
65564	Request expired: The request cannot be completed as the information is too old.
65565	Request retry: The request can be retried later.
65566	Invalid card number: A test card was used on a live Merchant Account. or Disallowed card number: A live card was used on a test Merchant Account.
65567	Unsupported card issuing country: Request is for a card issuing country that is not supported on this Merchant Account.
65568	Masterpass processing error.
65569	Masterpass not supported.
65570	Masterpass checkout failure.
65571	Masterpass checkout success.
65572	Masterpass checkout is required.
65573	Amounts error. Provided transaction amounts to not tally.
65574	Reserved for future use. Contact customer support if this error occurs.
65575	No data was found that match the selection criteria.
65576	Request cancelled.

3-D Secure Error Codes: 65792 – 65823	
Code	**Description**
65792	3-D Secure processing in progress.
65793	3-D Secure processing error.
65794	3-D Secure processing is required. 3-D Secure ACS challenge must be displayed.
65795	3-D Secure processing is not required.
65796	3-D Secure is not supported for this request: No versions of 3DS are supported for this request. Or 3DS processing not supported: request has threeDSRequired=Y but the Merchant Account isn't enabled for 3DS or required to do 3DS.
65797	Error occurred during 3-D Secure enrolment check.
65798	Reserved for future use.
65799	Reserved for future use.
65800	Error occurred during 3-D Secure authentication check.
65801	Reserved for future use.
65802	3-D Secure authentication is required.
65803	3-D Secure authentication results do not meet the Merchant's preferences.
65804	3-D Secure authentication was successful.

Remoted Checkout Processing Error Codes: 65824 – 65885	
Code	**Description**
65824	Remote checkout processing in progress.
65825	Remote checkout processing error.
65826	Remote checkout processing is required. Remote checkout must be displayed.
65827	Remote checkout processing is not required.
65828	Remote checkout processing is not supported.
65829	Remote checkout was successful.
65830	Remote checkout failed.

Risk Check Processing Error Codes: 65856 – 65887	
Code	**Description**
65856	Risk check processing in progress.

Risk Check Processing Error Codes: 65856 – 65887	
Code	**Description**
65857	Risk check processing error.
65858	Risk check processing required.
65859	Risk check processing is not required.
65860	Risk check processing is not supported.
65861	Risk check processor communication error.
65862	Risk check results do not meet the Merchant's preferences.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66048	Missing request. No data posted to integration URL.
66049	Missing **merchantID** field.
66050	Missing **merchantPwd** field.
66051	Reserved for internal use. Contact customer support if this error occurs.
66052	Reserved for internal use. Contact customer support if this error occurs.
66053	Reserved for internal use. Contact customer support if this error occurs.
66054	Reserved for internal use. Contact customer support if this error occurs.
66055	Missing **action** field.
66056	Missing **amount** field.
66057	Missing **currencyCode** field.
66058	Missing **cardNumber** field.
66059	Missing **cardExpiryMonth** field.
66060	Missing **cardExpiryYear** field.
66061	Missing **cardStartMonth** field. (Legacy field)
66062	Missing **cardStartYear** field. (Legacy field)
66063	Missing **cardIssueNumber** field. (Legacy field)
66064	Missing **cardCVV** field.
66065	Missing **customerName** field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66066	Missing `customerAddress` field.
66067	Missing `customerPostcode` field.
66068	Missing `customerEmail` field.
66069	Missing `customerPhone` field.
66070	Missing `countryCode` field.
66071	Missing `transactionUnique` field.
66072	Missing `orderRef` field.
66073	Missing `remoteAddress` field.
66074	Missing `redirectURL` field.
66075	Missing `callbackURL` field.
66076	Missing `merchantData` field.
66077	Reserved for internal use. Contact customer support if this error occurs.
66078	Missing `duplicateDelay` field.
66079	Missing `itemQuantity` field.
66080	Missing `itemDescription` field
66081	Missing `itemAmount` field.
66082	Missing `taxAmount` field.
66083	Missing `discountAmount` field.
66084	Missing `discountReason` field.
66085	Missing `xref` field.
66086	Missing `type` field.
66087	Missing `signature` field (field is required if message signing is enabled).
66088	Missing `authorisationCode` field.
66089	Missing `transactionID` field.
66090	Missing `threeDSRequired` field.
66091	Missing `threeDSMD` field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66092	Missing `threeDSPaRes` field.
66093	Missing `threeDSECI` field.
66094	Missing `threeDSCAVV` field.
66095	Missing `threeDSXID` field.
66096	Missing `threeDSEnrolled` field.
66097	Missing `threeDSAuthenticated` field.
66098	Missing `threeDSCheckPref` field.
66099	Missing `cv2CheckPref` field.
66100	Missing `addressCheckPref` field.
66101	Missing `postcodeCheckPref` field.
66102	Missing `captureDelay` field.
66103	Missing `orderDate` field.
66104	Missing `grossAmount` field.
66105	Missing `netAmount` field.
66106	Missing `taxRate` field.
66107	Missing `taxReason` field.
66108	Missing `surchargeRules` field.
66109	Reserved for internal use. Contact customer support if this error occurs.
66110	Missing `statementNarrative1` field.
66111	Missing `statementNarrative2` field.
66112	Missing `merchantName` field.
66113	Missing `merchantCompany` field.
66114	Missing `merchantAddress` field.
66115	Missing `merchantTown` field.
66116	Missing `merchantPostcode` field.
66117	Missing `merchantCountryCode` field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66118	Missing **merchantPhone** field.
66119	Missing **merchantMobile** field.
66120	Missing **merchantEmail** field.
66121	Missing **merchantWebsite** field.
66122	Missing **merchantOrderRef** field.
66123	Missing **merchantCustomerRef** field.
66124	Reserved for future use. Contact customer support if this error occurs.
66125	Missing **merchantType** field.
66126	Reserved for future use. Contact customer support if this error occurs.
66127	Missing **merchantCategoryCode** field.
66128	Missing **supplierName** field.
66129	Missing **supplierCompany** field.
66130	Missing **supplierAddress** field.
66131	Missing **supplierTown** field.
66132	Missing **supplierPostcode** field.
66133	Missing **supplierCountryCode** field.
66134	Missing **supplierPhone** field.
66135	Missing **supplierMobile** field.
66136	Missing **supplierEmail** field.
66137	Missing **supplierCounty** field.
66138	Missing **customerCompany** field.
66139	Missing **customerTown** field.
66140	Missing **customerCountryCode** field.
66141	Missing **customerMobile** field.
66142	Missing **customerCounty** field.
66143	Missing **merchantCounty** field.
66144	Missing **customerOrderRef** field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66145	Missing `customerMerchantRef` field.
66146	Missing `customerVatNumber` field.
66147	Missing `customerDateOfBirth` field.
66148	Missing `deliveryName` field.
66149	Missing `deliveryCompany` field.
66150	Missing `deliveryAddress` field.
66151	Missing `deliveryTown` field.
66152	Missing `deliveryPostcode` field.
66153	Missing `deliveryCountryCode` field.
66154	Missing `deliveryPhone` field.
66155	Missing `deliveryMobile` field.
66156	Missing `deliveryEmail` field.
66157	Reserved for future use. Contact customer support if this error occurs.
66158	Missing `deliveryCounty` field.
66159	Missing `deliveryFax` field.
66160	Missing `cardExpiryDate` field.
66161	Missing `cardStartDate` field. (Legacy field)
66162	Reserved for future use. Contact customer support if this error occurs.
66163	Reserved for future use. Contact customer support if this error occurs.
66164	Missing `items` field.
66165	Missing `itemGrossAmount` field.
66166	Missing `itemNetAmount` field.
66167	Missing `itemProductCode` field.
66168	Missing `itemCommodityCode` field.
66169	Missing `itemUnitOfMeasure` field.
66170	Missing `itemUnitAmount` field.
66171	Missing `itemDiscountAmount` field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66172	Missing `itemDiscountReason` field.
66173	Missing `itemTaxRate` field.
66174	Missing `itemTaxAmount` field.
66175	Missing `itemTaxReason` field.
66176	Missing `shippingTrackingRef` field.
66177	Missing `shippingMethod` field.
66178	Missing `shippingAmount` field.
66179	Missing `shippingNetAmount` field.
66180	Missing `shippingGrossAmount` field.
66181	Missing `shippingTaxRate` field.
66182	Missing `shippingTaxAmount` field.
66183	Missing `shippingTaxReason` field.
66184	Missing `shippingDiscountAmount` field.
66185	Missing `shippingDiscoutReason` field.
66186	Reserved for future use. Contact customer support if this error occurs.
66187	Reserved for future use. Contact customer support if this error occurs.
66188	Reserved for future use. Contact customer support if this error occurs.
66189	Reserved for future use. Contact customer support if this error occurs.
66190	Reserved for future use. Contact customer support if this error occurs.
66191	Reserved for future use. Contact customer support if this error occurs
66192	Missing `walletID` field.
66193	Missing `walletName` field.
66194	Missing `walletDesc` field.
66195	Missing `walletData` field.
66196	Missing `cardID` field.
66197	Missing `cardName` field.
66198	Missing `cardDesc` field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66199	Missing **cardData** field.
66200	Missing **customerAddressID** field.
66201	Missing **customerAddressName** field.
66202	Missing **customerAddressDesc** field.
66203	Missing **customerAddressData** field.
66204	Missing **deliveryAddressID** field.
66205	Missing **deliveryAddressName** field.
66206	Missing **deliveryAddressDesc** field.
66207	Missing **deliveryAddressData** field.
66208	Missing **walletOwnerRef** field.
66209	Missing **cardToken** field.
66210	Missing **masterPassData** field.
66211	Reserved for future use. Contact customer support if this error occurs.
66212	Missing **masterPassCheckoutOptions** field.
66213	Missing **masterPassCallbackURL** field.
66214	Missing **masterPassCheckoutURL** field.
66215	Missing **masterPassToken** field.
66216	Missing **masterPassVerifier** field.
66217	Missing **masterPassResourceURL** field.
66218	Missing **masterPassStatus** field.
66219	Missing **masterPassWalletID** field.
66220	Missing **handlingAmount** field.
66221	Missing **insuranceAmount** field.
66222	Missing **paymentMethod** field.
66223	Missing **paymentToken** field.
66256	Missing **receiverName** field.
66257	Missing **receiverCompany** field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66258	Missing **receiverAddress** field.
66259	Missing **receiverTown** field.
66260	Missing **receiverPostCode** field.
66261	Missing **receiverCountryCode** field.
66262	Missing **receiverPhone** field.
66263	Missing **receiverMobile** field.
66264	Missing **receiverEmail** field.
66265	Missing **receiverDateOfBirth** field.
66266	Missing **receiverAccountNo** field.
66267	Missing **receiverCounty** field.
66268	Missing **receiverFax** field.
66269	Missing **customerFax** field.
66270	Missing **rtScheduleType** field.
66271	Missing **rtSchedule** field.
66272	Missing **rtID** field.
66273	Missing **rtName** field.
66274	Missing **rtDescription** field.
66275	Missing **rtPolicyRef** field.
66276	Missing **rtMerchantID** field.
66277	Missing **rtStartDate** field.
66278	Missing **rtInitialDate** field.
66279	Missing **rtInitialAmount** field.
66280	Missing **rtFinalDate** field.
66281	Missing **rtFinalAmount** field.
66282	Missing **rtCycleAmount** field.
66283	Missing **rtCycleDuration** field.
66284	Missing **rtCycleDurationUnit** field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66285	Missing `rtCycleCount` field.
66286	Missing `rtMerchantData` field.
66287	Missing `rtCloneFields` field.
66288	Missing `checkoutRef` field.
66289	Missing `checkoutRedirectURL` field.
66290	Missing `checkoutOptions` field.
66291	Missing `checkoutRequest` field.
66292	Missing `checkoutResponse` field.
66293	Missing `rtAgreementType` field.
66294	Missing `rtSequenceNumber` field.
66295	Missing `rtSequenceCount` field.
66296	Missing `itemProductURL` field.
66297	Missing `itemImageURL` field.
66298	Missing `itemSize` field.
66299	Missing `itemWeight` field.
66300	Missing `riskCheckPref` field.
66301	Missing `riskCheckOptions` field.
66302	Missing `cloneFields` field.
66303	Missing `customField` field.
66560	Missing `dccRef` field.
66561	Missing `dccProvider` field.
66562	Missing `dccRate` field.
66563	Missing `dccMargin` field.
66564	Missing `dccCommission` field.
66565	Missing `dccSource` field.
66566	Missing `dccCreated` field.
66567	Missing `dccExpires` field.

Missing Request Field Error Codes: 66048 – 66303 and 66560 – 66815	
Code	**Description**
66568	Missing **dccCurrencyCode** field.
66569	Missing **dccAmount** field.
66570	Missing **dccAccepted** field.
66592	Missing **threeDSRef** field.
66593	Missing **threeDSVersion** field.
66594	Missing **threeDSRedirectURL** field.
66595	Missing **threeDSOptions** field.
66596	Missing **threeDSDetails** field.
66597	Missing **threeDSURL** field.
66598	Missing **threeDSRequest** field.
66599	Missing **threeDSResponse** field.
66600	Missing **threeDSPolicy** field.
66608	Missing **scaExemption** field.
66628	Missing **riskCheckPref** field.
66656	Missing **deviceType** field.
66657	Missing **deviceChannel** field.
66658	Missing **deviceIdentity** field.
66659	Missing **deviceTimeZone** field.
66660	Missing **deviceCapabilities** field.
66661	Missing **deviceAcceptContent** field.
66662	Missing **deviceAcceptCharset** field.
66663	Missing **deviceAcceptEncoding** field.
66664	Missing **deviceAcceptLanguage** field.
66665	Missing **deviceScreenResolution** field.
66666	Missing **deviceOperatingSystem** field.
66688	Missing **initiator** field.
66689	Missing **acquirerOptions** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66304	Invalid request. No data posted to integration URL.
66305	Invalid **merchantID** field.
66306	Invalid **merchantPwd** field.
66307	Reserved for internal use. Contact customer support if this error occurs.
66308	Reserved for internal use. Contact customer support if this error occurs.
66309	Reserved for internal use. Contact customer support if this error occurs.
66310	Reserved for internal use. Contact customer support if this error occurs.
66311	Invalid **action** field.
66312	Invalid **amount** field.
66313	Invalid **currencyCode** field.
66314	Invalid **cardNumber** field.
66315	Invalid **cardExpiryMonth** field.
66316	Invalid **cardExpiryYear** field.
66317	Invalid **cardStartMonth** field. (Legacy field)
66318	Invalid **cardStartYear** field. (Legacy field)
66319	Invalid **cardIssueNumber** field. (Legacy field)
66320	Invalid **cardCVV** field.
66321	Invalid **customerName** field.
66322	Invalid **customerAddress** field.
66323	Invalid **customerPostcode** field.
66324	Invalid **customerEmail** field.
66325	Invalid **customerPhone** field.
66326	Invalid **countyCode** field.
66327	Invalid **transactionUnique** field.
66328	Invalid **orderRef** field.
66329	Invalid **remoteAddress** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66330	Invalid `redirectURL` field.
66331	Invalid `callbackURL` field.
66332	Invalid `merchantData` field.
66333	Reserved for internal use. Contact customer support if this error occurs.
66334	Invalid `duplicateDelay` field.
66335	Invalid `itemQuantity` field.
66336	Invalid `itemDescription` field
66336	Invalid `itemAmount` field.
66338	Invalid `taxAmount` field.
66339	Invalid `discountAmount` field.
66340	Invalid `discountReason` field.
66341	Invalid `xref` field.
66342	Invalid `type` field.
66343	Invalid `signature` field (field is required if message signing is enabled).
66344	Invalid `authorisationCode` field.
66345	Invalid `transactionID` field.
66346	Invalid `threeDSRequired` field.
66347	Invalid `threeDSMD` field.
66348	Invalid `threeDSPaRes` field.
66349	Invalid `threeDSECI` field.
66350	Invalid `threeDSCAVV` field.
66351	Invalid `threeDSXID` field.
66352	Invalid `threeDSEnrolled` field.
66353	Invalid `threeDSAuthenticated` field.
66354	Invalid `threeDSCheckPref` field.
66355	Invalid `cv2CheckPref` field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66356	Invalid `addressCheckPref` field.
66357	Invalid `postcodeCheckPref` field.
66358	Invalid `captureDelay` field.
66359	Invalid `orderDate` field.
66360	Invalid `grossAmount` field.
66361	Invalid `netAmount` field.
66362	Invalid `taxRate` field.
66363	Invalid `taxReason` field.
66364	Invalid `surchargeRules` field.
66365	Reserved for internal use. Contact customer support if this error occurs.
66366	Invalid `statementNarrative1` field.
66367	Invalid `statementNarrative2` field.
66368	Invalid `merchantName` field.
66369	Invalid `merchantCompany` field.
66370	Invalid `merchantAddress` field.
66371	Invalid `merchantTown` field.
66372	Invalid `merchantPostcode` field.
66373	Invalid `merchantCountryCode` field.
66374	Invalid `merchantPhone` field.
66375	Invalid `merchantMobile` field.
66376	Invalid `merchantEmail` field.
66377	Invalid `merchantWebsite` field.
66378	Invalid `merchantOrderRef` field.
66379	Invalid `merchantCustomerRef` field.
66380	Reserved for internal use. Contact customer support if this error occurs.
66381	Invalid `merchantType` field.
66382	Reserved for internal use. Contact customer support if this error occurs

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66383	Invalid **merchantCategoryCode** field.
66384	Invalid **supplierName** field.
66385	Invalid **supplierCompany** field.
66386	Invalid **supplierAddress** field.
66387	Invalid **supplierTown** field.
66388	Invalid **supplierPostcode** field.
66389	Invalid **supplierCountryCode** field.
66390	Invalid **supplierPhone** field.
66391	Invalid **supplierMobile** field.
66392	Invalid **supplierEmail** field.
66393	Invalid **supplierCounty** field.
66394	Invalid **customerCompany** field.
66395	Invalid **customerTown** field.
66396	Invalid **customerCountryCode** field.
66397	Invalid **customerMobile** field.
66398	Invalid **customerCounty** field.
66399	Invalid **merchantCounty** field.
66400	Invalid **customerOrderRef** field.
66401	Invalid **customerMerchantRef** field.
66402	Invalid **customerVatNumber** field.
66403	Invalid **customerDateOfBirth** field.
66404	Invalid **deliveryName** field.
66405	Invalid **deliveryCompany** field.
66406	Invalid **deliveryAddress** field.
66407	Invalid **deliveryTown** field.
66408	Invalid **deliveryPostcode** field.
66409	Invalid **deliveryCountryCode** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66410	Invalid **deliveryPhone** field.
66411	Invalid **deliveryMobile** field.
66412	Invalid **deliveryEmail** field.
66413	Reserved for future use. Contact customer support if this error occurs
66414	Invalid **deliveryCounty** field.
66415	Invalid **deliveryFax** field.
66416	Invalid **cardExpiryDate** field.
66417	Invalid **cardStartDate** field. (Legacy field)
66418	Invalid line **items** count, either too few or too many for processor.
66419	Invalid line **items** sequence, either not sequential or not 0 to 99.
66420	Invalid **items** field.
66421	Invalid **itemGrossAmount** field.
66422	Invalid **itemNetAmount** field.
66423	Invalid **itemProductCode** field.
66424	Invalid **itemCommodityCode** field.
66425	Invalid **itemUnitOfMeasure** field.
66426	Invalid **itemUnitAmount** field.
66427	Invalid **itemDiscountAmount** field.
66428	Invalid **itemDiscountReason** field.
66429	Invalid **itemTaxRate** field.
66430	Invalid **itemTaxAmount** field.
66431	Invalid **itemTaxReason** field.
66432	Invalid **shippingTrackingRef** field.
66433	Invalid **shippingMethod** field.
66434	Invalid **shippingAmount** field.
66435	Invalid **shippingNetAmount** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66436	Invalid **shippingGrossAmount** field.
66437	Invalid **shippingTaxRate** field.
66438	Invalid **shippingTaxAmount** field.
66439	Invalid **shippingTaxReason** field.
66440	Invalid **shippingDiscountAmount** field.
66441	Invalid **shippingDiscoutReason** field.
66442	Reserved for future use. Contact customer support if this error occurs.
66443	Reserved for future use. Contact customer support if this error occurs.
66444	Reserved for future use. Contact customer support if this error occurs.
66445	Reserved for future use. Contact customer support if this error occurs.
66446	Reserved for future use. Contact customer support if this error occurs.
66447	Reserved for future use. Contact customer support if this error occurs
66448	Invalid **walletID** field.
66449	Invalid **walletName** field.
66450	Invalid **walletDesc** field.
66451	Invalid **walletData** field.
66452	Invalid **cardID** field.
66453	Invalid **cardName** field.
66454	Invalid **cardDesc** field.
66455	Invalid **cardData** field.
66456	Invalid **customerAddressID** field.
66457	Invalid **customerAddressName** field.
66458	Invalid **customerAddressDesc** field.
66459	Invalid **customerAddressData** field.
66460	Invalid **deliveryAddressID** field.
66461	Invalid **deliveryAddressName** field.
66462	Invalid **deliveryAddressDesc** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66463	Invalid **deliveryAddressData** field.
66464	Invalid **walletOwnerRef** field.
66465	Invalid **cardToken** field.
66466	Invalid **masterPassData** field.
66467	Reserved for future use. Contact customer support if this error occurs.
66468	Invalid **masterPassCheckoutOptions** field.
66469	Invalid **masterPassCallbackURL** field.
66470	Invalid **masterPassCheckoutURL** field.
66471	Invalid **masterPassToken** field.
66472	Invalid **masterPassVerifier** field.
66473	Invalid **masterPassResourceURL** field.
66474	Invalid **masterPassStatus** field.
66475	Invalid **masterPassWalletID** field.
66476	Invalid **handlingAmount** field.
66477	Invalid **insuranceAmount** field.
66478	Invalid **paymentMethod** field.
66479	Invalid **paymentToken** field.
66512	Invalid **receiverName** field.
66513	Invalid **receiverCompany** field.
66514	Invalid **receiverAddress** field.
66515	Invalid **receiverTown** field.
66516	Invalid **receiverPostCode** field.
66517	Invalid **receiverCountryCode** field.
66518	Invalid **receiverPhone** field.
66519	Invalid **receiverMobile** field.
66520	Invalid **receiverEmail** field.
66521	Invalid **receiverDateOfBirth** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66522	Invalid **receiverAccountNo** field.
66523	Invalid **receiverCounty** field.
66524	Invalid **receiverFax** field.
66525	Invalid **customerFax** field.
66526	Invalid **rtScheduleType** field.
66527	Invalid **rtSchedule** field.
66528	Invalid **rtID** field.
66529	Invalid **rtName** field.
66530	Invalid **rtDescription** field.
66531	Invalid **rtPolicyRef** field.
66532	Invalid **rtMerchantID** field.
66533	Invalid **rtStartDate** field.
66534	Invalid **rtInitialDate** field.
66535	Invalid **rtInitialAmount** field.
66536	Invalid **rtFinalDate** field.
66537	Invalid **rtFinalAmount** field.
66538	Invalid **rtCycleAmount** field.
66539	Invalid **rtCycleDuration** field.
66540	Invalid **rtCycleDurationUnit** field.
66541	Invalid **rtCycleCount** field.
66542	Invalid **rtMerchantData** field.
66543	Invalid **rtCloneFields** field.
66544	Invalid **checkoutRef** field.
66545	Invalid **checkoutRedirectURL** field.
66546	Invalid **checkoutOptions** field.
66547	Invalid **checkoutRequest** field.
66548	Invalid **checkoutResponse** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66549	Invalid **rtAgreementType** field.
66550	Invalid **rtSequenceNumber** field.
66551	Invalid **rtSequenceCount** field.
66552	Invalid **itemProductURL** field.
66553	Invalid **itemImageURL** field.
66554	Invalid **itemSize** field.
66555	Invalid **itemWeight** field.
66556	Invalid **riskCheckPref** field.
66557	Invalid **riskCheckOptions** field.
66558	Invalid **cloneFields** field.
66559	Invalid **customField** field.
66816	Invalid **dccRef** field.
66817	Invalid **dccProvider** field.
66818	Invalid **dccRate** field.
66819	Invalid **dccMargin** field.
66820	Invalid **dccCommission** field.
66821	Invalid **dccSource** field.
66822	Invalid **dccCreated** field.
66823	Invalid **dccExpires** field.
66824	Invalid **dccCurrencyCode** field.
66825	Invalid **dccAmount** field.
66826	Invalid **dccAccepted** field.
66848	Invalid **threeDSRef** field.
66849	Invalid **threeDSVersion** field.
66850	Invalid **threeDSRedirectURL** field.
66851	Invalid **threeDSOptions** field.
66852	Invalid **threeDSDetails** field.

Invalid Request Field Error Codes: 66304 – 66559 and 66816 – 67071	
Code	**Description**
66853	Invalid `threeDSURL` field.
66854	Invalid `threeDSRequest` field.
66855	Invalid `threeDSResponse` field.
66856	Invalid `threeDSPolicy` field.
66864	Invalid `scaExemption` field.
66880	Invalid `riskCheckPref` field.
66912	Invalid `deviceType` field.
66913	Invalid `deviceChannel` field.
66914	Invalid `deviceIdentity` field.
66915	Invalid `deviceTimeZone` field.
66916	Invalid `deviceCapabilities` field.
66917	Invalid `deviceAcceptContent` field.
66918	Invalid `deviceAcceptCharset` field.
66919	Invalid `deviceAcceptEncoding` field.
66920	Invalid `deviceAcceptLanguage` field.
66921	Invalid `deviceScreenResolution` field.
66922	Invalid `deviceOperatingSystem` field.
66944	Invalid `initiator` field.
66945	Invalid `acquirerOptions` field.

A-2 AVS / CV2 Check Response Codes

The AVS/CV2 Check Response Message field **`avscv2ResponseMessage`** is sent back in the raw form that is received from the Acquiring bank and can contain the following values:

Response	Description
ALL MATCH	AVS and CV2 match
SECURITY CODE MATCH ONLY	CV2 match only
ADDRESS MATCH ONLY	AVS match only
NO DATA MATCHES	No matches for AVS and CV2
DATA NOT CHECKED	Supplied data not checked
SECURITY CHECKS NOT SUPPORTED	Card scheme does not support checks

The AVS/CV2 Response Code **`avscv2ResponseCode`** is made up of six characters and is sent back in the raw form that is received from the Acquiring bank. The first 4 characters can be decoded as below, the remaining 2 characters are currently reserved for future use:

Position 1 Value	Description
0	No Additional information available.
1	CV2 not checked
2	CV2 matched.
4	CV2 not matched
8	Reserved

Position 2 Value	Description
0	No Additional information available.
1	Postcode not checked
2	Postcode matched.
4	Postcode not matched
8	Postcode partially matched

Position 3 Value	Description
0	No Additional Information
1	Address numeric not checked
2	Address numeric matched
4	Address numeric not matched
8	Address numeric partially matched

Position 4 Value	Description
0	Authorising entity not known
1	Authorising entity – merchant host
2	Authorising entity – acquirer host
4	Authorising entity – card scheme
8	Authorising entity – issuer

A-3 3-D Secure Authentication Data

The 3-D Secure system uses various data fields to report the authentication status of the Cardholder. Each 3-D Secure version may use slightly different terminology for the fields and have slightly different values but for ease of use the Gateway uses the terminology and values as described in this appendix.

The field's values would normally be populated by the Gateway's 3DS Server component[1], however you may choose to use your own 3DS Server component and provide the values, as described in section 5.4.7.

A-3.1 3-D Secure Enrolment Status

The **`threeDSEnrolled`** field indicates if the card is enrolled in the 3-D Secure program.

The value is determined if the card number is within one of the enrolled ranges downloaded daily from the Directory Server using a Preparation Request/Response (PReq/PRes) message.

The field can contain one of the following values:

Y – Enrolled. The card is enrolled in the 3-D Secure program and the payer is eligible for authentication processing.

N - Not Enrolled. The checked card is eligible for the 3-D Secure (it is within the card association's range of accepted cards) but the card issuing bank does not participate in the 3-D Secure program. If the Cardholder later disputes the purchase, the issuer may not submit a chargeback to you.

U - Unable To Verify Enrolment. The card associations were unable to verify whether the Cardholder is registered. As the card is ineligible for 3-D Secure, you can choose to accept the card nonetheless and precede the purchase as non-authenticated and submits authorisation with ECI 07.

E - Error Verifying Enrolment. The Gateway encountered an error. This card is flagged as 3-D Secure ineligible. The card can be accepted for payment at your discretion.

[1] The 3DS Server is the Gateway/Merchant component that provides the interface with the 3DS Directory Server.

A-3.2 3DS Authentication Status

The **`threeDSAuthenticated`** field indicates if the cardholder has been authenticated by the 3-D Secure program.

The value is provided by the Directory Server either on requesting authentication in the Authentication Response (ARes) message, in the case of a frictionless flow, or after a Cardholder challenge in the Result Request (RReq) message, in the case of a challenge flow.

The field can contain one of the following values:

Y - Authentication Successful. The Issuer has authenticated the Cardholder by verifying the identity information or password. A CAVV and an ECI of 5 is returned. The card is accepted for payment and authentication data passed to authorisation processing.

A - Attempted Authentication. A proof of authentication attempt was generated. The Cardholder is not participating, but the attempt to authenticate was recorded. The card can be accepted for payment at your discretion and authentication data passed to authorisation processing.

N - Not Authenticated. The Cardholder did not complete authentication and the card should not be accepted for payment.

R – Rejected By Issuer. The Issuer rejected the transaction and must not be accepted for payment.

D – Decoupled Challenge Required. Decoupled authentication confirmed.

I – Information Only. 3DS Requestor challenge preference acknowledged.

U - Unable To Authenticate. The authentication was not completed due to technical or another problem. A transmission error prevented authentication from completing. The card can be accepted for payment at your discretion, but no authentication data will be passed on to authorisation processing.

E - Error Checking Authentication. The Gateway encountered an error. The card can be accepted for payment at your discretion, but no authentication data will be passed on to authorisation processing.

A-3.3 3-D Secure Transaction Identifier

The **`threeDSXID`** field provides a unique value to identify the transaction through the 3-D Secure system.

The value is provided by the Directory Server and is referred to as the Directory Server Transaction ID (dsTransID). It is a 36-character universally unique identifier (UUID) as defined in IETF RFC 4122.

When using the Gateway's built in 3-D Secure integration the initial value for this field in the initial request may differ from that in the final requests. This is because initially the Gateway may not have had any communications with the Directory Server and thus not received the identifier.

Note: When external 3-D Secure authentication is performed (refer to section 5.4.7) this value must be provided and contain the Directory Server Transaction ID and not any of the other ids such as the 3DS Server Transaction ID or ACS Server Transaction ID.

A-3.4 3DS Electronic Commerce Indicator

The **`threeDSECI`** field indicates the security status of the transaction after the Cardholder has been authenticated or attempted authentication.

The value is provided by the Directory Server either on requesting authentication in the Authentication Response (ARes) message, in the case of a frictionless flow, or after a Cardholder challenge in the Result Request (RReq) message, in the case of a challenge flow.

The value is always present if the **`threeDSAuthenticated`** field has a value of **Y** (successful authentication), or **A** (attempted authentication) but can be present at other times.

The field can contain one of the following 2-digit values[1]:

05/02 - Both cardholder and card issuing bank are 3DS enabled. 3DS card authentication is successful.
06/01 - Either cardholder or card issuing bank is not 3DS enrolled. 3DS card authentication is unsuccessful, in sample situations as:
1. 3DS cardholder not enrolled.
2. Card issuing bank is not 3DS ready.

07/00 - Authentication is unsuccessful or not attempted. The card is either a non-3DS card or card issuing bank does not handle it as a 3DS transaction.

[1] The values are show as pairs, the first value is for Visa and other Card Schemes and the second for Mastercard only.

A-3.5 3DS Cardholder Authentication Verification Value

The **`threeDSCAVV`** field provides proof that the Cardholder has been authenticated or attempted authentication.

The value is provided by the Directory Server either on requesting authentication in the Authentication Response (ARes) message, in the case of a frictionless flow, or after a Cardholder challenge in the Result Request (RReq) message, in the case of a challenge flow.

The value is present if the **`threeDSAuthenticated`** field has a value of **Y** (successful authentication), or **A** (attempted authentication).

The field will contain a 28-character Base-64 encoded value (32-characters for Mastercard).

A-4 3-D Secure Enrolment/Authentication Only

Usually, the Gateway will perform most of the 3-D Secure processing in the background leaving the only the actual contacting of the issuers Access Control Server (ACS) to the Merchant.

However, there may be times when you may wish to gain more control over the Enrolment and Authentication process. The following field allows the request processing to stop after the 3-D Secure enrolment check or authentication check and return;

Field Name	Mandatory?	Description
`threeDSOnly`	No	Complete the processing as far as the next 3-D Secure stage and then return with the appropriate response fields for that stage.

As this stop is requested then a **`responseCode`** is returned as **0 (Success)** however it will be recorded in the Merchant Management System (MMS) as **65792 (3DS IN PROGRESS)** indicating that the transaction has been prematurely halted expecting it to be continued to the next 3-D Secure stage when required. In order to continue the process, the **`threeDSRef`** field is returned together with any relevant 3-D Secure response fields suitable for that stage in the processing.

If this flag is used when 3-D Secure is not enabled on the account or after the 3-D Secure process has been completed for the request (ie when the authentication step has completed), then passing the flag will cause the transaction to abort with a **`responseCode`** of **65795 (3DS PROCESSING NOT REQUIRED)**. This ensures that the transaction doesn't go on to completion by accident while trying do 3-D Secure enrolment or authentication only.

3-D Secure Enrolment/Authentication Only is supported by the Direct Integration only.

A-5 Request Checking Only

Sometimes, you may wish to submit a request to the Gateway in order for it to be 'validated only' and not processed or sent to the Acquirer. In these cases, the following flag can be used that will stop the processing after the integrity verification has been performed:

Field Name	Mandatory?	Description
`checkOnly`	No	Check the request for syntax and field value errors only. Do not attempt to submit the transaction for honouring by the Merchant's financial institution.

If the request is OK, then a `responseCode` is returned as **0 (Success)**; otherwise, the code that would have prevented the request from completing is returned.

Note: *in these cases, the request is not stored by the Gateway and is not available within the Merchant Management System (MMS).*

A-6 Merchant Account Mapping

Merchant Accounts can be grouped together so that if a transaction is sent to an account that doesn't support either the requested card type or currency, then it can be automatically routed to another account in the same group that does support them.

For example, you can group a Merchant Account that only supports American Express cards with a Merchant Account that only supports Visa cards. Then, if a request using an American Express card is sent to the Visa only Merchant Account, the Gateway will automatically route it to the American Express Merchant Account.

This prevents you from needing to know the card type in advance in order to send the request to the correct Merchant Account. This is important when using the Hosted integration, because you don’t know the card type at the time you send the request.

It is usual for you to have one master account to which you direct all requests and then group all your accounts together.

Any Gateway routing of the transaction can be seen from the following additional response fields:

Field Name	Returned?	Description
requestMerchantID	Always	ID of Merchant Account request was sent to (usually same as **merchantID**).
processMerchantID	Always	ID of Merchant Account request was processed by.

A-7 Velocity Control System (VCS)

The Gateway allows you to configure velocity controls using the Merchant Management System (MMS). These can be used to email you declined transactions automatically, where they exceed these controls.

For example, you can set up a control that stops a certain card number from being used more than twice in the space of a few minutes.

If one or more of these controls are broken by a transaction, then the following response fields will show the problem.

If a transaction is declined through breach of one or more of these rules, then a **`responseCode`** of **5 (VCS DECLINE)** will be returned.

Field Name	Returned?	Description
`vcsResponseCode`	Always	VCS error code. Normally **5**. Refer to appendix A-1 for details.
`vcsResponseMessage`	Always	Description of above response code or list of rules broken by this transaction.

A-8 Duplicate Transaction Checking

Duplicate transaction checking prevents transaction requests from accidentally processing more than once. This can happen if a Customer refreshes your checkout page or clicks a button that issues a new transaction request. While duplicate checking can help prevent repeat transactions from going through, we recommend talking with your developers to see whether changes can be made to your form to reduce the likelihood of this occurring (eg disabling the Submit button after it has been clicked).

To help prevent duplicate transactions, each transaction can specify a time window during which previous transactions will be checked to see whether they could be possible duplicates.

This time window is specified using the **`duplicateDelay`** field. The value for this field can range from 0 to 9999 seconds (approx. 2 ¾ hours).

If the transaction request does not include the **`duplicateDelay`** field or specifies a value of zero, then a default delay of 300 seconds (5 minutes) is used.

The following fields are used in transaction comparison and must be the same for a transaction to be regarded as a duplicate:

- **`merchantID`**
- **`action`**
- **`type`**
- **`amount`**
- **`transactionUnique`**
- **`currencyCode`**
- **`xref`** (if provided in lieu of card details)
- **`cardNumber`** (may be specified indirectly via cross reference)

If a transaction is regarded as being a duplicate, then a **`responseCode`** of **65554 (REQUEST DUPLICATE**) will be returned.

A-9 Capture Delay

Capture Delay enables you to specify a delay between the authorisation of a payment and its capture. This allows you time to verify the order and choose whether to fulfil it or cancel it. This can be very helpful in preventing chargebacks due to fraud.

When NOT using capture delay, payments are authorised and captured immediately - funds are automatically debited from the Customer's credit or debit card at that time.

When using capture delay, the payment is authorised only at the time of payment - funds are reserved against the credit or debit card and will not be debited until the payment is captured; or not at all if you cancel.

The Customer experience with capture delay is the same as when capture delay is not used. The Customer will not know whether you are using capture delay or not.

If you choose to use capture delay, you can use the **`captureDelay`** request field to specify the number of days for which capture is delayed, within a range of **0** to **30** days. Alternatively, you can use the value **-1** or '**never**' to specify that the Gateway should never automatically capture in which case you must manually capture.

The Gateway will automatically capture the transaction after any delay specified unless you manually cancel or capture the transaction, using either the Direct Integration or via the Merchant Management System (MMS).

Note that some cards require capture within 4 to 5 days - if payment is not automatically captured within that period, the transaction will expire, and the reserved funds will be released to the Customer.

Why Use Capture Delay?

Capture delay allows you to accept online orders normally but allows you to cancel any transactions that you cannot or will not fulfil, thereby reducing the risks of chargeback. If you receive an order that appears to be fraudulent or that you cannot or do not wish to fulfil, you can simply cancel the transaction.

Note: Cancelling a transaction may not always reverse the authorisation and release the funds back to the Customer. This is dependent on the Acquirer and in these cases the authorisation will never be settled and will be left to expire releasing any reserved funds. The time taken for this varies between cards.

Some Acquirers do not support delayed capture, in which case the Hosted Integration will return a ***`responseCode`*** *of* ***66358 (INVALID CAPTURE DELAY)****.*

A-10 Card Identification

The Gateway will attempt to identify the type of each card number provided along with the associated Card Scheme (network) and, when available, the issuing institution and issuing country[1].

The Gateway primarily supports Mastercard, Visa and American Express branded cards. Some Acquirers may support JCB, Discover and Diners Club cards. Not all Acquirers support all types.

The card type and scheme codes will be returned in the **`cardTypeCode`** and **`cardSchemeCode`** response fields. When available, the issuer details will be returned in the **`cardIssuer`** and **`cardIssuerCountryCode`** response fields.

If the Gateway is unable to identify the card, then a code of 'XX' will be returned.

The following fields are returned by the Gateway to indicate a cards identity.

Field Name	Description
`cardTypeCode`	Card type identification code, eg 'VD'.
`cardType`	Card type name, eg, 'Visa Debit'.
`cardSchemeCode`	Scheme identification code, eg 'VC'.
`cardScheme`	Scheme name, eg 'Visa'.
`cardIssuer`	Card Issuer's name, eg 'HSBC UK BANK PLC'.
`cardIssuerCountryCode`	Issuing country's ISO-3166 2-letter code, eg 'GB'.
`cardIssuerCountry`	Issuing country's name, eg 'United Kingdom'.
`cardFlags`	Bitmask of flags where each bit number indicates the following: **01** – debit (not credit) card. **02** – alternative debit (not credit) card. **03** – prepaid card. **09** – corporate card (not consumer) card. **12** – tokenised card (reserved for future use). **17** – ECOM use allowed. **18** – MOTO use allowed. **19** – EPOS use allowed. **20** – CA use allowed. **21** – LUHN conforming. **22** – 3-D Secure available (doesn't indicate the card is enrolled). **23** – Contactless available. *An unset bit may mean that the information is unavailable and not necessarily that the information doesn't apply to the card.*

[1] Issuing institution and country are primarily available for Visa and Mastercard issued cards only.

The following is a list of primary card types supported by the Gateway.

Card Code	Scheme Code	Card Type
MC	**MC**	Mastercard Credit
MD	**MC**	Mastercard Debit
MA	**MC**	Mastercard International Maestro
MI	**MC**	Mastercard/Diners Club
MP	**MC**	Mastercard Purchasing
MU	**MC**	Mastercard Domestic Maestro (UK)
VC	**VC**	Visa Credit
VD	**VC**	Visa Debt
EL	**VC**	Visa Electron
VA	**VC**	Visa ATM
VP	**VC**	Visa Purchasing
AM	**AM**	American Express
JC	**JC**	JCB
DS	**DS**	Discover
DI	**DI**	Diners Club
DC	**DI**	Diners Club Carte Blanche
CU	**CU**	China UnionPay
CC	**CU**	China UnionPay Credit
CD	**CU**	China UnionPay Debit

The following is a list of secondary card types recognised by the Gateway. These cards may be returned in response to a card lookup, but they are either deprecated or most likely not supported by any current Acquirer.

Card Code	Scheme Code	Card Type
CF	**CF**	Clydesdale Financial Services
BC	**BC**	BankCard
DK	**DK**	Dankort
DE	**DI**	Diners Club Enroute
FC	**FC**	FlexCache
LS	**LS**	Laser
SO	**SO**	Solo
ST	**ST**	Style
SW	**SW**	Switch
TP	**TP**	Tempo Payments
IP	**IP**	InstaPayment

A-11 Integration Testing

You will be provided with unique test Merchant Account IDs during the onboarding process. Refer to section 1.6 for more information.

Test Merchant Accounts are not connected to our Simulator and not to an actual Acquirer. The Simulator will emulate the function of an Acquirer and provide simulated responses and authorisation codes.

A-11.1 Test Amounts

When testing the transaction **`amount`** can be used to trigger different authorisation and settlement outcomes as follows:

Min. Amount	Max. Amount	Authorisation response	Settlement outcome
100 (1.00)	2499 (24.99)	(0) AUTH CODE: XXXXXX	ACCEPTED
2500 (25.00)	4999 (49.99)	(0) AUTH CODE: XXXXXX	REJECTED
5000 (50.00)	7499 (74.99)	(1) CARD REFERRED (0) AUTH CODE: XXXXXX	ACCEPTED
7500 (75.00)	9999 (99.99)	(1) CARD REFERRED (0) AUTH CODE: XXXXXX	REJECTED
10000 (100.00)	14999 (49.99)	(5) CARD DECLINED	N/A
15000 (150.00)	19999 (199.99)	(4) CARD DECLINED – KEEP CARD	N/A
20000 (200.00)	24999 (249.99)	(65) CARD DECLINED - SCA REQUIRED (0) AUTH CODE: XXXXXX	ACCEPTED
25000 (250.00)	29999 (299.99)	(65) CARD DECLINED – SCA REQUIRED (5) CARD DECLINED	N/A

Any other amount will return a **`responseCode`** of **66311 (Invalid Test Amount)**.

The settlement outcome only applies to transactions which reach settlement due to being successfully authorised and captured and not cancelled. The amount captured is used when determining the settlement outcome rather than the amount authorised.

The range 5000 to 9999 can be used to test manual authorisations. If the transaction does not contain an **`authorisationCode`** request field, then the Simulator will return a **`responseCode`** of **1 (CARD REFERRED)**. If it does, then it will return a **`responseCode`** of **0 (SUCCESS)**, with an amount between 50000 and 7499 being accepted at settlement and an amount of 7500 to 9999 being rejected.

The range 20000 to 29999 can be used to test SCA soft declines. If the transaction is eligible[1] to request SCA then the Simulator will return a **`responseCode`** of **65 (SCA REQUIRED**). If not, then it will return a **`responseCode`** of **0 (SUCCESS)** for the range 20000 to 24999 or **5 (DO NOT HONOR)** for the range 25000 to 29999**.** Successful transactions will be approved at settlement.

[1] A cardholder-initiated e-commerce sale or verify transaction that is enabled for 3-D Secure but is not already authenticated. SCA exemptions are not supported by the simulator and so cannot be used to request that SCA is not required.

A-11.2 Test Cards

The test accounts will only accept card numbers that are designated for test purposes. These test cards cannot be used on production accounts.

To test AVS and CV2 verification then the associated CVV and billing addresses are provided for each card. If a different value is used, then the Simulator will mark the responses as 'not matched'.

Unless stated otherwise an expiry date sometime in the near future should be used.

A-11.2.1 Visa Credit

Card Number	CVV	Address
4929421234600821	356	Flat 6 Primrose Rise 347 Lavender Road Northampton NN17 8YG
4543059999999982	110	76 Roseby Avenue Manchester M63X 7TH
4543059999999990	689	23 Rogerham Mansions 4578 Ermine Street Borehamwood WD54 8TH

A-11.2.2 Visa Debit

Card Number	CVV	Address
4539791001730106	289	Unit 5 Pickwick Walk 120 Uxbridge Road Hatch End Middlesex HA6 7HJ
4462000000000003	672	Mews 57 Ladybird Drive Denmark 65890

A-11.2.3 Electron

Card Number	CVV	Address
4917480000000008	009	5-6 Ross Avenue Birmingham B67 8UJ

A-11.2.4 Mastercard Credit

Card Number	CVV	Address
5301250070000191	419	25 The Larches Narborough Leicester LE10 2RT
5413339000001000	304	Pear Tree Cottage The Green Milton Keynes MK11 7UY
5434849999999951	470	34a Rubbery Close Cloisters Run Rugby CV21 8JT
5434849999999993	557	4-7 The Hay Market Grantham NG32 4HG

A-11.2.5 Mastercard Debit

Card Number	CVV	Address
5573 4712 3456 7898	159	Merevale Avenue Leicester LE10 2BU

A-11.2.6 UK Maestro

Card Number	CVV	Address
6759 0150 5012 3445 002	309	The Parkway 5258 Larches Approach Hull North Humberside HU10 5OP
6759 0168 0000 0120 097	701	The Manor Wolvey Road Middlesex TW7 9FF

A-11.2.7 JCB

Card Number	CVV	Address
3540599999991047	209	2 Middle Wallop Merideth-in-the-Wolds Lincolnshire LN2 8HG

A-11.2.8 American Express

Card Number	CVV	Address
374245455400001	4887	The Hunts Way Southampton SO18 1GW

A-11.2.9 Diners Club

Card Number	CVV Number	Address[1]
36432685260294	**111**	**N/A**

[1] Diners Club do not support the Address Verification Service (AVS). For testing purposes, we advise that a separate Merchant Account is used with AVS is turned off.

A-11.3 3-D Secure Testing

Your test accounts are connected to our 3-D Secure Product Integration Testing (PIT) system rather than to the production 3-D Secure servers.

You can use any of the test cards provided in section A-11.2 with this PIT system, and the authentication status returned by the Directory Server (for frictionless flow simulation) can be selected using the value of the card expiry month as follows:

Card Expiry Month	Auth Status	Simulation (Frictionless)
01 - Jan	**Y**	Fully authenticated
02 - February	**N**	Not authenticated
03 - March	**U**	Unknown authentication status
04 - April	**A**	Attempted authentication
05 - May	**D**	Decoupled authentication
06 - June	**R**	Transaction rejected (do not attempt to send for authorisation)
07 – July	**E**	Unknown error performing 3-D Secure checks
08 - August	**E**	Error due to timeout communicating with the Directory Server
09 – September	**E**	Error due to corrupt response from the Directory Server.
10 – October	**I**	Information only
11 – November	**U**	Unknown authentication due to Cardholder not enrolled (error 13)
12 - December	**C**	Frictionless not possible, challenge Cardholder

An expiry month of 12 will simulate the non frictionless flow and desired authentication status (**`threeDSAuthenticated`**) can be selected on the challenge dialog shown by the PIT Access Control Server.

When using an expiry month from the above table please use a suitable expiry year to ensure the date is sometime in the near future.

A-11.4 PayPal Sandbox Accounts

Please contact customer support to have your own PayPal test Merchant account created that connects to your own PayPal sandbox account, thus enabling you to view the transactions as they are sent to PayPal.

A-11.5 Amazon Pay Sandbox Accounts

Please contact customer support to have your own Amazon Pay test Merchant account created that connects to your own Amazon Pay sandbox account, thus enabling you to view the transactions as they are sent to Amazon Pay.

A-12 Sample Signature Calculation

It is highly recommended that transactions are protected using message signing. The signing process offers a quick and simple way to ensure that the message came from an authorised source and has not been tampered with during transmission.

Signing, however, must be completed on your servers and never left for the Customer's browser to complete in JavaScript, as this would mean revealing your secret signature code to anyone who viewed the JavaScript code in the browser.

Signatures are especially important when a transaction is sent from a browser's payment form via the use of hidden form fields because the Customer can easily use tools built into their browser to modify these hidden fields and change items such as the amount they should be charged.

Care must be taken to ensure that fields are sorted before signing into ascending field name order according to their numeric ASCII value. Some languages natural sort routines do **NOT** use ASCII order by default and so need to be adjusted or alternative methods used.

Also, when signing requests with fields formatted as per the *record* format detailed in section 1.7.8, only the root integration field is included in any sorting as the sub-fields are part of the value and should not have their order changed. The sub-fields must then be sent in the same order as they were hashed if added as hidden fields in HTML forms etc.

The section below gives a step-by-step example of how to sign a transaction, complete with coding examples using the PHP language.

Example Signature Key:

```
$key = 'DontTellAnyone'
```

Example Transaction:

```
$tran = array (
  'merchantID' => '100001',
  'action' => 'SALE',
  'type' => '1',
  'currencyCode' => '826',
  'countryCode' => '826',
  'amount' => '2691',
  'transactionUnique' => '55f025addd3c2',
  'orderRef' => 'Signature Test',
  'cardNumber' => '4929 4212 3460 0821',
  'cardExpiryDate' => '1213',
)
```

The transaction used for signature calculation must not include any 'signature' field as this will be added after signing when its value is known.

Step 1 - Sort transaction values by their field name

Transaction fields must be in ascending field name order according to their **numeric ASCII value**.

```
ksort($tran);
```

```
array ( 'action' => 'SALE', 'amount' => '2691', 'cardExpiryDate' => '1213',
'cardNumber' => '4929 4212 3460 0821', 'countryCode' => '826', 'currencyCode' => '826',
'merchantID' => '100001', 'orderRef' => 'Signature Test', 'transactionUnique' =>
'55f025addd3c2', 'type' => '1' )
```

Step 2 - Create url encoded string from sorted fields

Use RFC 1738 and the application/x-www-form-urlencoded media type, which implies that spaces are encoded as plus (+) signs.

```
$str = http_build_query($tran, '', '&');
```

```
action=SALE&amount=2691&cardExpiryDate=1213&cardNumber=4929+4212+3460+0821&countryCode=
826&currencyCode=826&merchantID=100001&orderRef=Signature+Test&transactionUnique=55f025
addd3c2&type=1
```

Step 3 - Normalise all line endings in the url encoded string

Convert all CR NL, NL CR, CR character sequences to a single NL character.

```
$str = str_replace(array('%0D%0A', '%0A%0D', '%0D'), '%0A', $str);
```

```
action=SALE&amount=2691&cardExpiryDate=1213&cardNumber=4929+4212+3460+0821&countryCode=
826&currencyCode=826&merchantID=100001&orderRef=Signature+Test&transactionUnique=55f025
addd3c2&type=1
```

Step 4 - Append your signature key to the normalised string

The signature key is appended to the normalised string with no separator characters.

```
$str .= 'DontTellAnyone'
```

```
action=SALE&amount=2691&cardExpiryDate=1213&cardNumber=4929+4212+3460+0821&countryCode=
826&currencyCode=826&merchantID=100001&orderRef=Signature+Test&transactionUnique=55f025
addd3c2&type=1DontTellAnyone
```

Step 5 - Hash the string using the SHA-512 algorithm

The normalised string is hashed to a more compact value using the secure SHA-512 hashing algorithm[1].

```
$signature = hash('SHA512', $str);
```

```
da0acd2c404945365d0e7ae74ad32d57c561e9b942f6bdb7e3dda49a08fcddf74fe6af6b23b8481b8dc8895
c12fc21c72c69d60f137fdf574720363e33d94097
```

Step 6 - Add the signature to the transaction form or post data

The signature should be sent as part of the transaction in a field called 'signature'.

```
<input type="hidden" name="signature" value="<?=$signature?>">
```

or

```
$tran['signature'] = $signature;
```

A-13 Transaction Life cycle

Each transaction received by the Gateway follows a pre-determined life cycle from receipt to completion. The stages in the life cycle are determined by the type of transaction and its success or failure at different stages in its life.

A-13.1 Authorise, Capture and Settlement

The key stages in the transaction's life cycle can be grouped into the Authorisation, Capture and Settlement stages as follows:

A-13.1.1 Authorisation

An authorisation places a hold on the transaction amount in the Cardholder's issuing bank. No money actually changes hands yet. For example, let's say that you are going to ship a physical product from your website. First, you authorise the amount of the transaction; then you ship the product. You only capture the transaction after the product is shipped.

A-13.1.2 Capture

A capture essentially marks a transaction as ready for settlement. As soon as the product is shipped, you can capture an amount up to the amount of the authorisation. Usually, the full amount is captured. An example of a situation in which the whole amount is not captured is where the Customer ordered multiple items and one of them is unavailable.

The Gateway will normally automatically capture all authorisations as soon as they are approved, freeing up you from having to do this.

However, it is usually more desirable to delay the capture either for a period of time or indefinitely. The **`captureDelay`** field can be used for this purpose and will allow you to state the number of days to delay any automatic capture or never to automatically capture. For more details on delayed capture, refer to appendix A-8.

A-13.1.3 Settlement

Within 24 hours, the Gateway will instruct your Acquirer to settle the captured transaction. The Acquirer then transfers the funds between the Cardholder's and your accounts.

A-13.2 Transaction States

At any time during the transaction's life cycle, it is in one of a number of states as follows:

A-13.2.1 Received

The transaction has been received by the Gateway and stored away. This is the first stage. The Gateway will examine the transaction and pass it on to the next stage, as appropriate.

A-13.2.2 Approved

The transaction has been sent to the Acquirer for authorisation and the Acquirer has approved it and is holding the Cardholder's funds.

This is an intermediate state and follows the **received** state.

A-13.2.3 Verified

The transaction has been sent to the Acquirer for verification and the Acquirer has confirmed that the account is valid.

This is a terminal state and follows the **received** state. The transaction will never be settled and no funds will ever be transferred

A-13.2.4 Declined

The transaction has been sent to the Acquirer for authorisation and the Acquirer declined it. The Acquirer will not usually give any reason for a decline and will not have held any funds.

The transaction has now completed its life cycle and no more processing will be done on it.

This is a terminal state and follows the **received** state. The transaction will never be settled and no funds will ever be transferred. The transaction **`responseCode`** will be **5 (Declined)**.

A-13.2.5 Referred

The transaction has been sent to the Acquirer for authorisation and the Acquirer referred it for verbal approval.

You can choose not to seek verbal approval and treat these transactions the same as a normal 'declined' authorisation.

To seek verbal approval, you must phone the Acquirer and ask for an authorisation code. They will probably ask for more information about the transaction and might require you to gather other forms of identification from the Cardholder. If an authorisation code is provided, then a new transaction can be sent to the Gateway specifying the **`xref`** of this transaction and the received **`authorisationCode`**. This new request will not be sent for authorisation and will be in the 'approved' state ready for capture and settlement.

This is a terminal state and follows the **received** state. The transaction will never be settled and no funds will ever be transferred. The transaction **`responseCode`** will be **2 (Referred)**.

A-13.2.6 Reversed

The transaction was sent to the Acquirer for authorisation and the Acquirer approved it. However, the transaction has been voided and the approval reversed. The Acquirer will have been asked to reverse any approval previously received, effectively cancelling the authorisation and returning any held funds back to the Cardholder.

The Gateway will reverse an authorisation if it declines the transaction post authorisation due to any AVS/CV checking. The PREAUTH action will also automatically reverse an authorisation before return.

This is a terminal state and follows the **approved** state. The transaction will never be settled and no funds will ever be transferred.

If the transaction was reversed due to AVS/CV2 checking, then the transaction **`responseCode`** will be **5 (AVS/CV2 Declined)**.

A-13.2.7 Captured

The transaction has been captured and the Acquirer will be asked to capture the approved held funds when the settling process next runs. The settling process usually runs each evening but the Acquirer may take up to 3 days to transfer the funds.

The **capture** state can either be entered automatically if the transaction requested an immediate or delayed capture; or it can be manually requested by sending a CAPTURE request. You are free to change the amount to be captured to a value less than that initially approved by issuing one or more CAPTURE commands. When captured, there is no way to un-capture a transaction. If not explicitly cancelled, it will be sent for settlement at the next opportunity.

This is an intermediate state and follows the **approved** state.

A-13.2.8 Tendered

The transaction has been sent to the Acquirer for settlement by the settling process and is awaiting confirmation that it has been accepted.

At this point, the transaction can no longer be cancelled or re-captured.

This is an intermediate state and follows the **captured** state.

A-13.2.9 Deferred

The transaction could not be settled due to some temporary problem such as a communications loss. It will be attempted again the next time the settling process runs – usually first thing the next day.

This is an intermediate state and follows the **tendered** state. It will normally be accompanied by a transaction response that indicates why the settlement process could not settle the transaction.

A-13.2.10 Accepted

The transaction has been accepted for settlement by the Acquirer. The held funds will be transferred between the Merchant and Cardholder in due course.

The transaction has now completed its life cycle and no more processing will be done on it, unless it is subject to a rejection while the Acquirer is settling it.

This is a terminal state and follows the **tendered** state.

A-13.2.11 Rejected

The transaction has been rejected for settlement by the Acquirer. The held funds will not be transferred between the Merchant and Cardholder.

Only a few Acquirers inform the Gateway that they have rejected a transaction: they usually inform you directly. Therefore, a transaction may show as **accepted** even if was ultimately rejected or it may change from **accepted** to **rejected** if the Acquirer does inform the Gateway.

The transaction has now completed its life cycle and no more processing will be done on it.

This is a terminal state and follows the **tendered** or **accepted** states. The transaction response will normally indicate the reason the transaction was rejected.

A-13.2.12 Canceled

The transaction has been cancelled by the Merchant by sending a cancellation request to the Gateway either using the CANCEL action or via the Merchant Management System (MMS).

You can cancel any transaction that is not in a terminal state or in the ‘tendered’ state. When cancelled, any further processing that would have normally taken place will be halted. Cancelling a transaction may or may not release any funds held on the Cardholder’s card, depending on support from the Acquirer and Card Scheme. Note: the state is spelt American style, with a single ‘l’ as **canceled**.

This is a terminal state and follows any non-terminal state that occurs before the transaction reaches the **tendered** state.

A-13.2.13 Finished

The transaction has finished and reached the end of its lifespan but did not reach one of the other terminal states. Usually this indicates that a problem has occurred with the transaction that prevents it continuing with its normal life cycle.

This is a terminal state and can follow any other state. The transaction response will normally indicate the reason that the transaction failed.

A-14 Transaction types

The Gateway only supports card not present (CNP) types of transactions, made where the Cardholder does not or cannot physically present the card for your visual examination at the time that an order is placed and payment effected.

The type of transaction required is specified using the **`type`** request field when performing a new payment transaction.

A-14.1 E-commerce (ECOM)

E-commerce transactions are supported by the Gateway by using a transaction **`type`** of **1**. They are designed for you to accept payments via a website, such as a shopping cart payment. E-commerce transactions can use advance fraud detection, such as 3-D Secure.

In accordance with Mastercard stipulations, the Gateway will not allow Maestro cards to be used for new e-commerce transactions without the use of 3-D Secure.

A-14.2 Mail Order/Telephone Order (MOTO)

Mail Order/Telephone Order transactions are supported by the Gateway by using a transaction **`type`** of **2**. They are designed for you to build your own virtual terminal system to enter remote order details. MOTO transactions cannot use 3-D Secure as the cardholder is not able to perform the challenge.

Your Acquirer may need to enable MOTO capabilities on your main acquiring account, or they provide a separate acquiring account which will be available through its own Gateway Merchant Account.

A-14.3 Continuous Authority (CA)

Continuous Authority transactions are supported by the Gateway by using a transaction **`type`** of **9**. They are designed for you to make subscription transactions. For further details on how to use Continuous Authority transactions, please refer to section 12.3.1.

Your Acquirer may need to enable Continuous Authority capabilities on your main acquiring account, or they provide a separate acquiring account which will be available through its own Gateway Merchant Account.

The Gateway offers a means of automating the taking of regular CA transactions using Recurring Transaction Agreements (RTA) as detailed in section 13.

A-15 Payment Tokenisation

All new transactions stored by the Gateway are assigned a unique reference number that is referred to as the cross reference and returned in the **`xref`** response field. This cross reference is displayed on the Merchant Management System (MMS) and used whenever a reference to a previous transaction is required.

The cross reference can be sent as part of a transaction request, in the **`xref`** request field, to tell the Gateway to perform an action on an existing transaction. This is usually for management actions such as **CANCEL** or **CAPTURE**.

The cross reference can also be sent with new transactions such as **PREAUTH**, **SALE**, and **REFUND** actions, to request that the Gateway uses the values from the existing transaction if they have not been specified in the new request. For more information on how the existing values are used, please refer to appendix A-16. This allows an existing transaction to be effectively repeated without your needing to know the original card number. The only exception to this is the card's security code (CVV) which the Gateway cannot store, due to PCI DSS restrictions. Accordingly, it will have to be supplied in the new request (unless the new request is a Continuous Authority transaction, refer to appendix A-14.3).

The use of cross references to perform repeat transactions is referred to as Payment Tokenisation and should not be confused with Card Tokenisation which is a separate service offered by the Gateway Wallet, covered in section 19.

Refer to section 12 for details on how to instruct the Gateway to repeat a payment automatically.

The Gateway will make transaction details available for a maximum period of 13 months, after this time the **`xref`** to the transaction will be invalid. The card number will be available during this time, but you may request that it is removed sooner. Once the card number has been removed the **`xref`** can no longer be used to provide the number to a future a transaction.

The way each action handles any supplied **`xref`** is as follows:

A-15.1 PREAUTH, SALE, REFUND, VERIFY requests

These requests will always create a new transaction.

The **`xref`** field can be provided to reference an existing transaction, which will be used to complete any missing fields in the current transaction. The previous transaction will not be modified. For more information on how the existing values are used, please refer to appendix A-16. If the existing transaction cannot be found, then an error will be returned and recorded against the new transaction

The request is expected to contain any transaction information required.

The **`xref`** will only be used to complete any missing card and order details, relieving you from having to store card details and reducing your PCI requirements.

A-15.2 REFUND_SALE requests

These requests will always create a new transaction.

The **`xref`** field can be provided to reference an existing transaction that is going to be refunded. This existing transaction will be marked as have been fully or partially refunded and the amounts will be tallied to ensure that you cannot refund more than the original amount of this existing transaction. If the existing transaction cannot be found, then an error will be returned and recorded against the new transaction.

The request is expected to contain any transaction information required.

The **`xref`** will not only be used to find the transaction to be refunded: additionally, that transaction will be used to complete any missing card and order details, relieving you from having to store card details and reducing your PCI requirements.

A-15.3 CANCEL or CAPTURE requests

These requests will always modify an existing transaction.

The **`xref`** field must be provided to reference an existing transaction, which will be modified to the desired state. If the existing transaction cannot be found, then an error is returned but no record of the error will be recorded against any transaction.

The request must not contain any new transaction information any attempt to send any new transaction information will result in an error. The exception is that a CAPTURE request can send in a new lesser **`amount`** field when a lesser amount, than originally authorised, must be settled.

A-15.4 QUERY requests

These requests will not create or modify any transaction.

The **`xref`** field must be provided to reference an existing transaction, which will be returned as if it had just been performed. If the existing transaction cannot be found, then an error is returned but no record of the error will be recorded against any transaction.

The request must not contain any new transaction information and any attempt to send any new transaction information will result in an error.

A-15.5 SALE or REFUND Referred Authorisation requests

These will always create a new transaction.

The **`xref`** field must be provided to reference an existing transaction, which must be of the same request type and be in the **referred** state. A new transaction will be created based upon this transaction. If the existing transaction cannot be found or is not in the **referred** state, then an error will be returned and recorded against the new transaction.

The new transaction will be put in the **approved** state and captured when specified by the existing or new transaction details. It will not be sent for authorisation again first.

The request may contain new transaction details, but any card details or order amount must be the same as the existing transaction. Any attempt to send different card details or order details will result in an error.

NB: This usage is very similar to a normal SALE or REFUND request sent with an **`authorisationCode`** included. The difference is that the **`xref`** must refer to an existing **referred** transaction whose full details are used if required and not simply an existing transaction whose card details are used if required.

This means it is not possible to create a pre-authorised SALE or REFUND request and use a **`xref`** (ie to use the card and order details from an existing transaction). As a soon as the **`xref`** field is seen, the Gateway identifies that it is a **referred** transaction that you wish to authorise.

A-16 Transaction Cloning

If a new transaction request is received with the Cross Reference (**`xref`**) of an existing transaction, then the values of certain fields in the existing transaction will be used to initialise the new transaction where new values have not been provided in the new request. This copying of fields from a base transaction is termed 'transaction cloning', and the copied-over value is termed the 'cloned value'.

Appendix A-16.1 shows all the fields whose values can be copied over from the existing transaction. To allow for easy addition of future fields, the fields are grouped into logical groupings and each group is given a name, given in brackets after the group title.

Certain groups of fields, such as address fields, can only be copied as a whole entity and any new value provided in the new request will prevent the whole group from being copied from the existing transaction. These fields are marked with an asterisk after the field name. This is explained in appendix A-16.2.1.

By default, the values of all the fields listed in appendix A-16.1 are copied from the existing transaction where appropriate. However, you can control exactly which fields are copied using the **`cloneFields`** field in the new request. The value of **`cloneFields`** should be a comma separated list of field names or group names that should be copied over. Alternatively, if you wish to specify a list of fields not to copy, then prefix the list with a single exclamation mark (!).

Field Name	Mandatory?	Description
`cloneFields`	N	Comma separated list of field names or group names whose values should be cloned.

Examples

To copy over only the value of **`customerName`** and any values for the fields in the ***customerAddressFields*** group:

```
cloneFields="customerName, customerAddressFields"
```

To copy over the values of all supported fields apart from the value of **`customerName`** and **`merchantName`**:

```
cloneFields="!customerName,merchantName"
```

A-16.1 Cloned Fields

Transaction fields currently cloned are as follows:

- Order Details Fields (**orderFields**)
  - **type**
  - **countryCode**
  - **currencyCode**
  - **amount**
  - **grossAmount**
  - **netAmount**
  - **taxRate**
  - **taxAmount**
  - **taxReason**
  - **discountAmount**
  - **discountReason**
  - **handlingAmount**
  - **insuranceAmount**
  - **surchargeAmount**

- Order Reference Fields (**orderRefFields**)
  - **transactionUnique**
  - **orderRef**
  - **orderDate**

- Card Fields (**cardFields**)
  - **paymentMethod**
  - **paymentToken**
  - **cardToken**
  - **cardNumber**
  - **cardExpiryDate***
  - **cardExpiryMonth***
  - **cardExpiryYear***
  - **cardStartDate***
  - **cardStartMonth***
  - **cardStartYear***
  - **cardIssueNumber**

- Cardholder Fields (**cardholderFields**)
  - **customerName**
  - **customerAddress**
  - **customerPostcode**
  - **customerEmail**
  - **customerPhone**

- Purchase Fields (**purchaseFields**)
  - **items**

- Statement Narrative Fields (**narrativeFields**)
  - **statementNarrative1**
  - **statementNarrative2**

- AVS/CV2 Fields (**avscv2Fields**)
  - **avscv2Required**
  - **cv2CheckPref**
  - **addressCheckPref**
  - **postcodeCheckPref**
  - **customerAddress**
  - **customerPostcode**

- Risk Check Fields (**riskCheckFields**)
  - **riskCheckRequired**
  - **riskCheckPref**
  - **riskCheckOptions**

- 3-D Secure Fields (**threedsFields**)[1]
  - **threeDSRequired**
  - **threeDSPolicy**
  - **threeDSCheckPref**
  - **threeDSRedirectURL**
  - **threeDSOptions**
  - **scaExemption**

- Merchant Email Notification Fields (**notifyFields**)
  - **notifyEmailRequired**
  - **notifyEmail**

- Customer Receipt Fields (**cReceiptFields**)
  - **customerReceiptsRequired**
  - **customerEmail**

- Merchant Information Fields (**merchantFields**)
  - **merchantName**
  - **merchantCompany**
  - **merchantAddress***
  - **merchantTown***
  - **merchantCounty***
  - **merchantPostcode***
  - **merchantCountryCode***
  - **merchantPhone**
  - **merchantMobile**
  - **merchantFax**
  - **merchantEmail**
  - **merchantWebsite**
  - **merchantData**
  - **merchantOrderRef**
  - **merchantCustomerRef**
  - **merchantTaxRef**
  - **merchantOriginalOrderRef**
  - **merchantCategoryCode**
  - **merchantAccountNo**
  - **merchantType**

  Customer Information Fields (**customerFields**)
  - **customerName**
  - **customerCompany**
  - **customerAddress***
  - **customerTown***
  - **customerCounty***
  - **customerPostcode***
  - **customerCountryCode***
  - **customerPhone**
  - **customerMobile**
  - **customerFax**
  - **customerEmail**
  - **customerDateOfBirth**
  - **customerOrderRef**
  - **customerMerchantRef**
  - **customerTaxRef**

[1] 3-D Secure fields are only cloned if both the existing and new transaction support 3-D Secure.

Supplier Information Fields (**supplierFields**)
- **supplierName**
- **supplierCompany**
- **supplierAddress***
- **supplierTown***
- **supplierCounty***
- **supplierPostcode***
- **supplierCountryCode***
- **supplierPhone**
- **supplierMobile**
- **supplierFax**
- **supplierEmail**
- **supplierOrderRef**
- **supplierAccountNo**

- Receiver Information Fields (**receiverFields**)
  - **receiverName**
  - **receiverCompany**
  - **receiverAddress***
  - **receiverTown***
  - **receiverCounty***
  - **receiverPostcode***
  - **receiverCountryCode***
  - **receiverPhone**
  - **receiverMobile**
  - **receiverFax**
  - **receiverEmail**
  - **receiverAccountNo**
  - **receiverDateOfBirth**

- Delivery Information Fields (**deliveryFields**)
  - **deliveryName**
  - **deliveryCompany**
  - **deliveryAddress***
  - **deliveryTown***
  - **deliveryCounty***
  - **deliveryPostcode***
  - **deliveryCountryCode***
  - **deliveryPhone**
  - **deliveryMobile**
  - **deliveryFax**
  - **deliveryEmail**

- Shipping Information Fields (**shippingFields**)
  - **shippingMethod**
  - **shippingTrackingRef**
  - **shippingAmount**
  - **shippingGrossAmount**
  - **shippingNetAmount**
  - **shippingTaxRate**
  - **shippingTaxAmount**
  - **shippingTaxReason**
  - **shippingDiscountAmount**
  - **shippingDiscountReason**

- MCC 6012 Additional Authorisation Data (**mcc6012Fields**)
  - **receiverName**
  - **receiverPostcode**
  - **receiverAccountNo**
  - **receiverDateOfBirth**

- Payment Facilitator Data (**facilitatorFields**)[1]
  - **subMerchantID**
  - **facilitatorID**
  - **facilitatorName**
  - **isoID**

- Surcharge Data (**surchargeFields**)
  - **surchargeRequired**
  - **surchargeAmount**
  - **surchargeRules**

- Device Data (**deviceFields**)
  - **deviceType**
  - **deviceChannel**
  - **deviceIdentity**
  - **deviceTimeZone**
  - **deviceCapabilities**
  - **deviceAcceptContent**
  - **deviceAcceptCharset**
  - **deviceAcceptEncoding**
  - **deviceAcceptLanguage**
  - **deviceScreenResolution**
  - **deviceOperatingSystem**

- Acquirer Data (**acquirerFields**)
  - **acquirerOptions**

[1] Payment facilitator fields are only cloned if the existing transaction uses the same **merchantID** as the new transaction.

A-16.2 Cloned Groups

To allow for easy future addition of new fields, the existing fields are grouped into logic groupings. Each group is given a name (as shown in brackets after the group title). It is recommended that this group name be used in any **`cloneFields`** value instead of listing all the fields separately.

A-16.2.1 Compound Groups

To help maintain transaction integrity, certain groups of fields, such as address fields, can only be copied as a whole entity and any new value provided in the new request will prevent the whole group from being copied from the existing transaction.

These compound fields are marked with an asterisk in appendix A-16.1 and can be referred to in **`cloneFields`** as logical groups using the following group names; ***merchantAddressFields***, ***customerAddressFields***, ***deliveryAddressFields***, ***supplierAddressFields*** and ***receiverAddressFields***.

A-16.2.2 Line-Item Data

Any line-item data (**`items`**) is copied over in its entirety and there is no way to merge the line item from an existing transaction with any sent in a new transaction.

A-16.2.3 Amount Consistency

The Gateway does not validate that the various sub-amount fields, such as **`netAmount`**, **`grossAmount`**, all add up to the actual requested **`amount`**. Therefore, these fields are currently not treated as a compound group.

If a new **`amount`** value is passed that is different from the value in the existing transaction, then the following fields should also be passed so that they tally with the new amount.

- **`grossAmount`**
- **`netAmount`**
- **`taxRate`**
- **`discountAmount`**

A-17 Credentials on File Matrix

The following table provides a quick reference for the correct flagging for the different credentials on file scenarios.

Scenario	CIT	CNP	COF	SCA	initiator	type	rtAgreementType	xref
Cardholder opts to store their card details on Merchant's website.	CIT	Ecom	Initial	Required	consumer	1	cardonfile	*(Optional for cloning)*
Cardholder opts to store their card details provided to Merchant via mail or telephone.	CIT	MoTo	Initial	Exempt	consumer	2	cardonfile	*(Optional for cloning)*
Cardholder pays using a card they previously stored on the Merchant's website.	CIT	Ecom	Subsequent	Required	consumer	1	cardonfile	Reference to transaction that initially stored the card
Cardholder provides their card details to sign up to a subscription on the Merchant's website.	CIT	Ecom	Initial or Subsequent	Required	consumer	1	recurring	*(Optional for cloning or using previous stored card)*
Cardholder provides their card details when agreeing to purchase by instalments on the Merchant's website.	CIT	Ecom	Initial or Subsequent	Required	consumer	1	instalment	*(Optional for cloning or using previous stored card)*
Cardholder provides their card details to sign up to a subscription via mail or telephone to the Merchant.	CIT	MoTo	Initial or Subsequent	Exempt	consumer	2	recurring	*(Optional for cloning or using previous stored card)*
Cardholder provides their card details when agreeing to purchase by instalments via mail or telephone to the Merchant.	CIT	MoTo	Initial or Subsequent	Exempt	consumer	2	instalment	*(Optional for cloning or using previous stored card)*
Merchant/Gateway takes an automatic subscription payment at the interval agreed to by the Cardholder.	MIT	CNP	Subsequent	Exempt	merchant	9	recurring	Reference to initial or previous recurring payment
Merchant/Gateway makes an automatic instalment payment at the interval agreed to by the Cardholder.	MIT	CNP	Subsequent	Exempt	merchant	9	instalment	Reference to initial or previous instalment payment
Merchant makes an unscheduled transaction, such as an account top-up, as previously agreed with the Cardholder when they stored their card details.	MIT	CNP	Subsequent	Exempt	merchant	2	unscheduled	Reference to transaction that initially stored the card
Merchant resubmits a payment where the initial payment was declined due to insufficient funds, but the goods have already been provided to the Cardholder.	MIT	CNP	N/A	Exempt	merchant	2	resubmission	Reference to initial payment that was declined

Scenario	CIT	CNP	COF	SCA	initiator	type	rtAgreementType	xref
Merchant reauthorises a payment when the completion or fulfilment of the original order or service extends beyond the authorization validity limit set by the Card Scheme.	MIT	CNP	N/A	Exempt	merchant	2	reauthorisation	Reference to payment that is to be reauthorised
Merchant makes a payment to process a supplemental account charge after original services have been rendered and respective payment has been processed.	MIT	CNP	N/A	Exempt	merchant	2	delayedcharges	Reference to original payment to which the delayed charges relate
Merchant makes a payment to charge the Cardholder a penalty according to the merchant's reservation cancellation policy.	MIT	CNP	N/A	Exempt	merchant	2	noshow	Reference to an initial CIT payment or account verification payment made by Cardholder at time of booking

A-18 PSD2 SCA Compliance

Strong Customer Authentication (SCA) is a requirement of the second Payment Services Directive (PSD2) in the European Economic Area (EEA), Monaco, and the United Kingdom[1]. It aims to add extra layers of security to e-commerce payments by requiring banks to perform additional checks when Customers make payments.

PSD2 is for banks, not for merchants. This means that to comply with the law in their home country, banks must refuse non-compliant payments. To avoid the risk of the bank declining your payment, you as a merchant need to ensure that your payments comply with PSD2 SCA regulations.

You can comply by obtaining additional authentication to verify the Customer's identity or by providing a valid reason for the payment to be exempt from SCA. Any authentication must use a least two of the following three elements:

1. Something the Customer knows (eg password)
2. Something the Customer has (eg phone)
3. Something the Customer is (eg fingerprint)

[1] PSD2 countries are: Austria, Belgium, Bulgaria, Croatia, Republic of Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Liechtenstein, Lithuania, Luxembourg, Malta, Monaco, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, and the UK.

A-18.1 Obtaining Strong Customer Authentication

Strong Customer Authentication applies to Customer entered online transactions (ECOM). Mail Order/Telephone Order (MOTO) transactions and recurring transactions are considered Merchant entered transactions and do not require strong authentication.

Currently, the most common way of authenticating an online card payment is to use 3-D Secure as covered in section 5. 3-D Secure is accepted as a means of obtaining Cardholder authentication for the purposes of SCA.

Other card-based payment method such as Apple Pay and some Google Pay payments already support payment flows with a built-in layer of strong authentication (biometric or password). These can be a great way for you to offer a frictionless checkout experience while meeting the new requirements.

We also expect many alternative European payment methods such as PayPal, Amazon Pay, iDEAL, etc. to follow the new SCA rules without any major changes to their user experience.

A-18.2 SCA Soft-Declines

There are two main types of card transaction declines: hard declines and soft declines.

Hard declines happen when the issuing bank rejects the transaction. Examples include attempting to use a card that has been reported lost or stolen, or the card has expired. Hard declines are permanent, so the payment should not be retried.

Soft declines are temporary authorisation failures. Around 80% to 90% of all declines fall into this category. They occur for a host of reasons including the need to authenticate the Cardholder further or because there are issues with the technical infrastructures that process the transaction. Soft declines are temporary, meaning you can process the transaction again after meeting the requirements that led to the decline the first time around.

If you do not obtain SCA on an eligible payment the issuer may soft decline the payment insisting that SCA be obtained. In which case a **`responseCode`** of **65** will be returned by the Gateway[1] and you can choose to resubmit the payment with SCA if required.

Refer to appendix A-18.4 for details on how you can use 3-D Secure to obtain SCA and automatically retry payments that have been soft declined for this reason. If 3-D Secure has been used to provide SCA and the issuer still declines insisting that SCA be obtained, then the Gateway will return a normal decline **`responseCode`** of **5** to prevent an infinite loop of obtaining SCA and then being declined for lack of SCA.

[1] Mastercard will return an ISO-8583 code of **65**, Visa/Diners/Discover and others will return an ISO-8583 code of **1A**. The Gateway will normalise these and always return a **`responseCode`** of **65**.

A-18.3 Exemptions to Strong Customer Authentication

There are some e-commerce transactions which are out of scope of the regulation, and others that may be exempt.

Obtaining Strong Customer Authentication can add friction and increase Customer drop-off and therefore you should make use of exemptions to reduce the number of times you will need to authenticate a Customer.

However, the bank has the right to refuse any requested exemption and decline the payment insisting that SCA be obtained. Refer to appendix A-18.2 for further information on this soft decline.

The following exemptions are available:

A-18.3.4 Mail Order / Telephone Order Payments

Card details collected via mail or over the phone (MOTO) fall outside of the scope of SCA and do not require authentication.

You can flag such payments by passing a value of **'2'** in the **`type`** request field.

A-18.3.5 Merchant Initiated Transactions (including recurring transactions)

Payments made with saved cards when the Customer is not present in the payment flow may qualify as Merchant Initiated Transactions (12.3). These payments fall outside of the scope of SCA however it is still up to the bank to decide whether authentication is needed for the payment.

The initial payment that saved the card will still need to have obtained SCA or be exempt and agreement must be obtained from the Customer to charge their card at a later point.

You can flag such payments by passing a value of **'9'** (Continuous Authority) in the **`type`** request field or using an **`rtAgreementType`** that signifies the transaction as being Merchant Initiated.

A-18.3.6 Low Value Exemption

Payments below €30 are considered low value and are generally exempt from authentication. However, if the Customer initiates more than five consecutive low value payments or if the total payments value exceeds €100) then SCA will be required.

You can request this exemption by passing a value of **'lowvalue'** in the **`scaExemption`** request field, or it may be automatically applied by the Issuer.

A-18.3.7 Trusted Beneficiary Exemption

The Customer can opt to trust you as a Merchant during their first authentication, then subsequent payments with you are likely to be exempt from future SCA.

You can request this exemption by passing a value of **'trusted'** in the **`scaExemption`** request field to allow this trust to be taken into consideration.

A-18.3.8 Trusted Risk Analysis (TRA) Exemption

If the payment provider, having in place effective risk analysis tools, assesses that the fraud risk associated with the payment is low then they can allow this exemption.

You can request this exemption by passing a value of **'risk'** in the **`scaExemption`** request field if agreed to by the payment provider.

A-18.3.9 Secured Corporate Payment Exemption

Payments initiated by a business rather than a Consumer and processed through a secured dedicated payment protocol can be exempt from SCA provided alternative controls are sufficiently secure.

You can request this exemption by passing a value of **'corporate'** in the **`scaExemption`** request field to indicate such a secure transaction.

A-18.3.10 Delegated Authentication Exemption

If you already require your Customers to perform sufficient authentication on your website, such as secure account logins etc., then you can use this exemption to request that further SCA is not required.

You can request this exemption by passing a value of **'delegated'** in the **`scaExemption`** request field to indicate such a secure transaction.

A-18.4 SCA Using 3-D Secure

3-D Secure can be used to provide the necessary Strong Customer Authentication. You have a choice of how and when to use 3-D Secure to satisfy SCA:

Authentication Before Authorisation
You submit payments using 3-D Secure for authentication up front so that the authorisation will be submitted to the Acquirer with the appropriate authentication data showing that SCA was sought. You may pass an exemption indicator[1] causing the Gateway to automatically request a frictionless flow[2].

Authentication After Authorisation, when requested by the Issuer (Bypass)
You submit payments without 3-D Secure authentication but with an exemption indicator[1], if required, and the authorisation will be submitted to the Acquirer with no authentication data. If the Issuer approves the authorisation, then no further additional authentication is needed. However, if the Issuer refuses the authorisation due to SCA being required[3] then transaction can be repeated but this time using 3-D Secure[4] and no exception indicator.

The Gateway can support both choices and in the case of the second choice it can automatically perform the repeat the transaction on your behalf.

The choice of how and when authentication is performed is indicating by select a 3-D Secure Policy in the Merchant Management System or by sending the **`threeDSPolicy`** field in the request.

The policies available are:

1. Authenticate Before Authorisation or When Issuer Requests (Default)
2. Authenticate Before Authorisation Only
3. Authenticate When Issuer Requests Only (Bypass)
4. Authenticate Before Authorisation or When Issuer Requests [PSD2]
5. Authenticate Before Authorisation Only [PSD2]
6. Authenticate When Issuer Requests Only [PSD2] (Bypass)

The [PSD2] policies will perform 3-D Secure authentication only if the transaction falls within the jurisdiction of the European Union's Payment Services Directive 2, otherwise it will behave as if 3-D Secure had not been required.

[1] An exemption can be explicitly requested using the **`scaExemption`** field, refer to appendix A-18.3.
[2] The Gateway will use the correct 'requestorChallengeIndicator' unless overridden by any value passed in the request.
[3] The issuer will soft decline the transaction indicating SCA is required, refer to appendix A-18.2.
[4] You are advised to send a **`threeDSOptions`** 'requestorChallengeIndicator' value of 4 to mandate a challenge.

A-19 Hosted Payment Page Options

You can customise the appearance of the Hosted Payment Page by sending additional fields in the request.

You can control which payment methods are displayed and the default or initial value to display in the input fields.

You can also state which fields you require to be mandatory, in which case the payment form may not be submitted until the Cardholder has complete all the mandatory fields. On the standard modal Hosted Payment Page, the mandatory requirement can also control whether the field needs to be displayed. A value of 'Y' or 'N' means the field must be displayed, 'Y' indicates that a value must be supplied, while 'N' indicates the value can be blank. Omitting the request field means that the field need not be displayed if the form can provide a better user experience without it.

Not all fields may be supported if you have a customised Hosted Payment Page.

Field Name	Mandatory?	Description
cardNumber	No[1]	Default value for the Card number field.
cardCVV	No[2]	Default value for the Card security number field.
cardExpiryMonth	No	Default value for the Card expiry month field.
cardExpiryYear	No	Default value for the Card expiry year field.
cardExpiryDate	No	Alternative to **cardExpiryMonth**/**cardExpiryYear**
customerName	No	Default value for the Cardholder's name field.
customerAddress	No	Default value for the Cardholder's address field.
customerTown	No	Default value for the Cardholder's town field.
customerCounty	No	Default value for the Cardholder's county field.
customerPostcode	No	Default value for the Cardholder's postcode field.
customerCountryCode	No	Default value for the Cardholder's country field.
customerEmail	No	Default value for the Cardholder's email field.
customerPhone	No	Default value for the Cardholder's phone number field.
receiverDateOfBirth	No	Default value for the Cardholder's date of birth field.
cardCVVMandatory	No	Card security number field requirements (**Y** or **N**).
customerNameMandatory	No	Cardholder's name field requirements (**Y** or **N**).
customerFullNameMandatory	No[3]	Cardholder's name field requirements (**Y** or **N**).
customerAddressMandatory	No	Cardholder's address field requirements (**Y** or **N**).
customerTownMandatory	No	Cardholder's town field requirements (**Y** or **N**).

Field Name	Mandatory?	Description
customerCountyMandatory	No	Cardholder's county field requirements (**Y** or **N**).
customerPostcodeMandatory	No	Cardholder's postcode field requirements (**Y** or **N**).
customerCountryCodeMandatory	No	Cardholder's country field requirements (**Y** or **N**).
customerPostcodeMandatory	No	Cardholder's postcode field requirements (**Y** or **N**).
customerEmailMandatory	No	Cardholder's email field requirements (**Y** or **N**).
customerPhoneMandatory	No	Cardholder's phone field requirements (**Y** or **N**).
receiverDateOfBirthMandatory	No	Cardholder's date of birth requirements (**Y** or **N**).
formAmountEditable	No	Enables the payment amount to be edited by the consumer thus allowing the consumer to choose the amount they wish to pay.
formResponsive	No	Request the Hosted Payment Page adjust its layout according to the browser display size etc. (**Y** or **N**).
formAllowCancel	No	Request the Hosted Payment Page show a cancel button to allow the payment to be cancelled resulting in a transaction **responseCode** of **65576 (REQUEST CANCELLED)**.
paymentMethod	No	Request the Hosted Payment Page invoke an alternative payment method on display without the need for the Customer to select it.
allowedPaymentMethods	No	Comma separated list of **paymentMethod**s supported by the Merchant to show on Hosted Payment Page where supported.

[1] This should only be used by Merchants who are storing Card numbers as per PCI DSS requirements.
[2] This should only be used for test purposed as Merchants are not allowed to store Card CVV numbers.
[3] Same as **customerNameMandatory** only the name must be provided as at least two words of two or more characters.

A-20 Integration Libraries

We can provide a range of libraries to help you to integrate with the Gateway.

These libraries include simple sever-side classes in many popular programming languages through to client-side scripts to help with the integration of the Hosted Payment Page or Hosted Payment Fields.

The server-side libraries can be obtained by contacting customer support.

The client-side libraries can be downloaded directly from the Gateway server.

A-20.1 Gateway Integration Library

A simple server-side integration library is available to simplify the preparation and transmission of Hosted and Direct Integration requests.

The library is available in many popular programming languages and is based around a single class: the **`Gateway`** class.

The Gateway integration library does not currently support the preparation and transmission of Batch Integration requests.

A-20.1.1 Library Namespace

To avoid polluting the global namespace, the library uses the 'P3/SDK' namespace where supported by the language.

A-20.1.2 Gateway Configuration

Before you can use the **`Gateway`** class, you will need to configure the following properties to match your integration parameters explained in section 1.6 and authentication parameters documented in section 1.8.

Property Name	Type	Description
`hostedURL`	string	Absolute URL provided for the Hosted Integration.
`directURL`	string	Absolute URL provided for the Direct Integration.
`merchantID`	string	Your unique Merchant ID to be passed in the **`merchantID`** integration field.
`merchantPwd`	string	Any password configured on your Merchant Account as per section 1.8.1.
`merchantSecret`	string	Any secret configured on your Merchant Account as per section 1.8.2.
`proxyUrl`	string	Absolute URL to any proxy required for connections. (eg https://www.proxy.com:3128)
`debug`	boolean	True to enable debugging output.

A-20.1.3 Gateway Methods

The follow methods are made available by the **`Gateway`** class:

`string hostedRequest(mixed[] request, string[] options)`

Return an HTML fragment that can be included in your webpage to render a <form> which will send the provided request data to the Gateway's Hosted Integration when submitted.

The **`request`** parameter should be an associative array containing the request fields required to be sent. The request fields are not validated.

The following class properties are used unless alternative values are provided in the **`request`** array: **`directUrl`**, **`merchantID`**, **`merchantPwd`**, **`merchantSecret`**.

The **`options`** parameter is an optional associative array containing options that can be used to modify the returned HTML fragment as follows:

- **`formAttrs`** – string containing additional attributes to include in the form tag.
- **`submitAttrs`** – string containing additional attributes to include in the submit button tag.
- **`submitImage`** – string containing the URL to use as the submit button.
- **`submitHtml`** – string containing HTML to use as the label on the submit <button>.
- **`submitText`** – string containing text to use as the label on the submit <input>.

The **`submitImage`**, **`submitHtml`** and **`submitText`** options are mutually exclusive and will be checked for in that order. If none is provided, then a **`submitText`** value of 'Pay Now' is assumed.

If a **`merchantSecret`** is provided, then the method will add the correct **`signature`** field to the request.

An exception is thrown if the HTML fragment cannot be composed.

The **`verifyResponse()`** method can be used to validate and decode any response POSTed back to your website.

Please refer to appendix A-23.1.1 for an example of how to use this method.

Returns a string containing the HTML fragment if successful; throws an exception otherwise.

`mixed[] directRequest(mixed[] request, string[] options)`

Return the response received when sending the provided request to the Gateway's Direct Integration.

The **`request`** parameter should be an associative array containing the request fields required to be sent. The request fields are not validated.

The following class properties are used unless alternative values are provided in the **`request`** array: **`directUrl`**, **`merchantID`**, **`merchantPwd`**, **`merchantSecret`**.

The **`options`** parameter is not used and reserved for future use.

If a **`merchantSecret`** is provided, then the method will add the correct **`signature`** field to the request and check the **`signature`** field on the response.

An exception is thrown if the request cannot be sent; or the response cannot be received; or if the response's **`signature`** is incorrect.

Please refer to appendix A-23.1.2 for an example of how to use this method.

Returns an associative array containing the received response fields; otherwise, throws an exception.

`void prepareRequest(mixed[] &request, string[] &options,`
`string &secret, string &direct_url, string &hosted_url)`

Prepare a request for sending to the Gateway's Direct Integration.

The **`request`** parameter should be a reference to an associative array containing the request fields required to be sent. The request fields are not validated.

The **`merchantSecret`**, **`directUrl and hostedUrl`** configuration properties will be returned in the **`secret`**, **`direct_url`** and **`hosted_url`** method parameters. These properties can be overridden by providing them in the **`request,`** in which case they will be extracted and removed from the request.

The **`merchantID`** and **`merchantPwd`** configuration properties will be added to the **`request`**.

A few known Gateway response fields will be removed from the request, if present, to avoid confusion, notably the **`responseCode`**, **`responseMessage`**, **`responseStatus`**, **`state`** fields.

An exception will be thrown if the request does not contain an action element or a merchantID element (and none could be inserted).

`void verifyResponse(mixed[] &response, string secret)`

Verify a response received from the Gateway's Hosted or Direct Integration.

The **`response`** parameter should be a reference to an associative array containing the response received from the Gateway, either from the Direct Integration or as POSTed from the Hosted Integration.

The **`secret`** parameter should be any Merchant secret to use when checking the response's **`signature`** element. If not provided, then the value of the **`merchantSecret`** property is used.

Any **`signature`** element is removed from the **`response`**.

An exception is thrown if the response is not valid, does not contain a **`responseCode`** element or its **`signature`** is incorrect.

Please refer to appendix A-23.1.1 for an example of how to use this method.

`string sign(mixed[] request, string secret, mixed partial = false)`

Return the signature for the provided request data.

The **`request`** parameter should be a reference to an associative array containing the request fields required to be sent. The request fields are not validated.

The **`secret`** parameter should be the Merchant secret to use when signing the request.

The **`partial`** parameter should be either the boolean **false** or comma separated string; or an array of strings containing the names of the request elements to sign.

Returns a string containing the correct signature for the request.

A-20.2 Hosted Payment Page Library

A simple client-side script is available to simplify the displaying of the Hosted Payment Page in a lightbox overlaying your website.

The library is available as a JavaScript script and is based around a single class: the **`Form`** class. The script is compatible with most modern web browsers.

The script can be loaded directly from our Gateway server as follows[1].

```
1.  <script src="https://gateway.example.com/sdk/web/v1/js/hostedforms.min.js"></script>
```

If the script detects the presence of the jQuery API, then it will extend the jQuery object with its own plugin method. However, jQuery is not needed in order to use the script.

A-20.2.1 Hosted Payment Pages

Hosted Payment Pages are a prebuilt webpage residing on our server that you can use to collect sensitive payment details without those details' touching your server. The standard Hosted Payment Page is designed so that it can be displayed in a transparent overlay over your website, thus making the Customer feel as though they never left your shopping cart.

The standard Hosted Integration examples redirect the Customer's browser to the Hosted Payment Page, resulting it appearing on a new browser page and not overlaying your website. The Hosted Payment Page library provides the scripting necessary to result in the redirection, causing the Hosted Payment Page to appear in an overlay and not a new browser page, without your having to make any modifications to your website. The library can also simplify the creation of the Hosted Integration redirection FORM if required.

A-20.2.2 Library Namespace

To avoid polluting the global namespace, the library extends the global `window` object with a **`hostedForms`** object containing the following properties:

- **`forms`** – array containing all the instantiated **`Form`** objects.
- **`classes`** – array containing all the instantiable classes.
  - **`form`** – **`Form`** class prototype.

[1] Please use the correct hostname as explained in section 1.6.

A-20.2.3 Form Construction

The construction method can be used to build and prepare a HTML FORM element for use with the modal Hosted Payment Page; or to prepare an existing element. The method signature is as follows:

```
Form(element, data)
```

The **element** parameter should be either the id or DOM node of an existing FORM or DIV DOM element.

If the **element** is a DIV node, then the data is used to create a new FORM node within the **element**.

If the **element** is a FORM node, then the data is used to modify the existing FORM **element**.

The **data** parameter should be an object containing construction details and can contain the following optional properties:

- **id** – string containing the value to use as the FORM tag's id attribute.
- **url** – string containing the URL to use as the FORM tag's src attribute.
- **attrs** – object whose properties are added as additional attributes on the FORM tag.
- **modal** – boolean indicating that the HPP should open in a modal overlay.
- **data** – object whose properties are added as hidden input elements in the FORM.
- **submit** – object containing details for a submit button that should be added to the FORM.
  - **type** – type of submit button, either 'auto', 'image', 'button', 'input'
  - **id** – string containing the value to use as the submit button's id attribute.
  - **attrs** – object whose properties are added as additional attributes on the submit button.
  - **label** – string containing button label (or 'alt' attribute for 'image' buttons)
  - **src** – string containing image URL for 'image' buttons.

The constructor will submit the FORM immediately after preparation if the **data.submit.type** property is 'auto'; or if the existing FORM **element** has a `data-hostedform-autosubmit` attribute. Otherwise, an event handler will be attached to the submit button to disable it automatically when clicked, to help prevent your Customer from clicking it twice.

The constructor will prepare the FORM so that the Hosted Payment Page (HPP) will be opened in a modal overlay if the **data.modal** property is true; or if the existing FORM **element** has a `data-hostedform-modal` attribute; or has an `action` attribute containing the string 'modal/' or ending in the string 'modal'.

The modal overlay is automatically created as a semi-opaque IFRAME element that fills the browser display. The Hosted Payment Page is then loaded into this IFRAME and being semi-opaque, your shopping cart will remain visible beneath, but greyed out and noninteractive. When the Customer closes the Hosted Payment Page, then their browser will be redirected to the URL you provided using the **redirectURL** parameter. This will cause the original page and IFRAME to be replaced by your new page.

A-20.2.4 Form Methods

The follow methods are made available by the **`Form`** class:

`void destroy()`

Destroys the **`Form`**, reverting its **`element`** back to its original state.

A-20.2.5 jQuery Plugin

If the jQuery API has been loaded into the browser before the script, then it will extend the jQuery object with its own plugin method.

Construction and destruction can then be done as follows:

```
$(element).hostedForm(data);
$(element).hostedForm('destroy');
```

A-20.3 Hosted Payment Fields Library

A simple client-side script is available to support the displaying of Hosted Payment Fields in your payment form.

The library is available as a JavaScript script and is based around two classes: the **`Form`** and **`Field`** classes. The script is compatible with most modern web browsers.

The script can be loaded directly from our Gateway server as follows[1]:

```
1.  <script src="https://gateway.example.com/sdk/web/v1/js/hostedfields.min.js"></script>
```

The script requires the jQuery API, which must be loaded prior to the script.

A-20.3.1 Hosted Payment Fields

Hosted Payment Fields are a set of prebuilt JavaScript UI components that can be used by your website's HTML payment form to collect sensitive payment details without those details touching your server. They provide you with the PCI benefits of using a Hosted Payment Page, while allowing you the ability to design and implement your own payment forms.

There are 6 predefined Hosted Payment Fields available as follows:

- **cardNumber** – collects the card number.
- **cardCVV** – collects the card cvv.
- **cardExpiryDate** – collects the card expiry month and year.
- **cardStartDate** – collects the card start/issue month and year.
- **cardIssueNumber** – collects the card issue number.
- **cardDetails** – collects the card number, expiry date and cvv in a single field.

The **cardNumber** field is designed to collect a card number, including an icon used to display the card type. The field will only accept digits and spaces and validate that any entered value is a correctly formatted card number and insert spaces at the correct positions for the card type as the number is typed.

The **cardCVV** field is designed to collect a card CVV. The field will only accept digits and will validate that any entered value is a correctly formatted CVV, taking into account the card type as determined by an associated **cardNumber** field.

The **cardExpiryDate** and **cardStartDate** fields are designed to collect a card expiry date and card issue date respectively. The fields can render as a pair of select controls containing the months and a suitable range of years; or as an input control that will only allow digits to be entered and automatically formatted as a month / year entry. The field will validate that any entered value is a valid month and year combination.

The **cardIssueNumber** field is designed to collect a card issue number. The field will only accept digits and will validate that any entered value is a correctly formatted issue number.

[1] Please use the correct hostname as explained in section 1.6.

The **cardDetails** field is designed to collect all of the essential card details. It combines the **cardNumber**, **cardExpiryDate** and **cardCVV** fields into a single line compound field design to allow easy entry of the card details and to complement the look of your checkout.

These hosted fields can be used on your payment form alongside any standard HTML form fields, for example, any collecting the Cardholder's billing or delivery addresses and any other order information you require.

The field type is either: passed as the value of the **`type`** option the **`Field`** construction, provided by the HTML element's meta data; or provided via the HTML element's type attribute (prefixed with the 'hostedfield:' name space).

The following example shows all three approaches to specifying the field type:

```
1.  <input type="hostedfield:cardNumber" name="card-number">
2.  <div class="hostedfield" data-hostedfield-type="cardExpiryDate"></div>
3.  <input data-hostedfield='{"type":"cardCVV"}'>
```

It is highly recommended that you adopt a single approach as above and don't mix and match.

Each field type has its own additional configuration options, as detailed in section A-20.3.6.

A-20.3.2 Library Namespace

To avoid polluting the global namespace, the library extends the global `window` object with a **`hostedFields`** object containing the following properties:

- **`forms`** – array containing all the instantiated **`Form`** objects.
- **`classes`** – array containing all the instantiable classes.
  - **`form`** – **`Form`** class prototype.

A-20.3.3 Form Construction

The construction method can be used to prepare a HTML FORM for use with Hosted Payment Field components. The method signature is as follows:

```
Form(element, options)
```

The **element** parameter should be the DOM node of an existing FORM tag.

The **options** parameter should be object containing one of more of the following optional properties:

- **autoSetup** – boolean indicating whether setup should be handled automatically.
- **autoSubmit** – boolean indicating whether submission should be handled automatically.
- **merchantID** – string containing the **merchantID** the payment request is for.
- **stylesheet** – string containing DOM selector for any stylesheets to be used.
- **tokenise** – string/array/object specifying fields whose values should be tokenised.
- **fields** – object containing field configuration by field type.
- **locale** – string containing the desired locale.
- **classes** – object containing names of extra CSS classes to use.
- **submitOnEnter** – boolean indicating whether the enter key should cause the form to submit.
- **nativeEvents** – boolean indicating that native browser events should be fired.

Any **options** parameter will be merged with those provided via meta data supplied, using `data-hostedfield` and/or `data-hostedfield-<option>` attributes; or via existing attributes or properties of the **element**.

The **autoSetup** option can be used to disable the automatic creation of **Field** objects for the FORM child controls by calling the **autoSetup()** method during the **Form** construction. If automatic setup is disabled, then you must manually instantiate **Field** objects and attach them to the **Form** as required, using the **addField()** method. This option or manually calling the **autoSetup()** method minimises the amount of JavaScript you have to write. Automatic operation is good if you don't need to customise the operation or can't customise it by reacting to the **Form** or **Field** events. The option defaults to true and cannot be changed once the **Form** has been created.

The **autoSubmit** option can be used to disable the automatic handling of the FORM submission via the **autoSubmit()** method. If automatic submission is disabled, then you must manually retrieve the sensitive payment details by calling **getPaymentDetails()** and include them in the form submission data. This option or manually calling the **autoSubmit()** method minimises the amount of JavaScript you have to write. Automatic operation is good if you don't need to customise the operation or can't customise it by reacting to the **Form** or **Field** events. The option defaults to true and cannot be changed once the **Form** has been created.

The **merchantID** option can be used to specify the **merchantID** with which the final **paymentToken** will be used. The option defaults to the value of any child INPUT node whose name is 'merchantID' and can be changed at runtime by calling the **setMerchantID()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **stylesheet** option can be used to specify a DOM selector used to locate stylesheets that should be parsed for styles related to the Hosted Payment Fields. Refer to section A-20.3.10 for

how to style the Hosted Payment Fields using CSS stylesheets. The option defaults to the DOM selector string 'link.hostedfield[rel=stylesheet], style.hostedfield' and can be changed at runtime by calling the **`setStylesheet()`** method; or by altering the options using the jQuery **`hostedForm()`** plugin method.

The **`tokenise`** option can be used to specify addition FORM controls whose values, as returned by the `jQuery.val()` method, should be included in the final **`paymentToken`**.
The option's value must be either:

- A string containing a DOM selector used to select one or more controls.
- An array containing values used to `jQuery.filter()` down to one or more controls.
- An object whose properties are the name of fields to tokenise and whose values are objects containing a **`selector`** property used to select a control.

For the first two, the tokenised field's name will be taken from the controls `data-hostedfield-tokenise` attribute or `name` attribute. For the third, the name is property name in the **`tokenise`** object. If the field's name is of the format 'paymentToken[<name>]', then only the '<name>' part is used. The option defaults to the DOM selector string 'INPUT.hostedfield-tokenise:not(:disabled), INPUT[data-hostedfield-tokenise]:not(:disabled), INPUT[name^="paymentToken["]:not(:disabled)' and cannot be changed once the **`Form`** has been created.

Currently only the card details and customer address fields can be tokenised, that is the field's name must be one of cardExpiryMonth, cardExpiryYear, cardExpiryDate, cardStartMonth, cardStartYear, cardStartDate, cardIssueNumber, customerName, customerCompany, customerAddress, customerTown, customerCountry, customerPostcode or customerCountryCode.

The **`fields`** options can be used to specify default options for the different types of Hosted Payment Fields. The option's value should be an object whose properties are the fields type or the wildcard type 'any' and whose values are objects whose properties are the default options for fields of that type. The values can also contain a **`selector`** property containing a DOM selector that is used during the automatic setup stage to select a FORM's child element to add as a **`Field`** of the specified type automatically. The option has no default value and cannot be changed once the **`Form`** has been created.

The **`locale`** option can be used to specify the language that should be used by the Hosted Payment Fields attached to this **`Form`**. The option defaults to the value provided by any `lang` attribute on the **element** or closest ancestor and cannot be changed once the **`Form`** has been created.

The **`classes`** options can be used to specify additional CSS class names to add in addition to the default classes documented in section A-20.3.9. The value is an object whose properties are the default class name and whose values are a string containing the additional class name(s) to use. The option has no default and cannot be changed once the **`Form`** has been created.

The **`submitOnEnter`** option can be used to specify if pressing the enter key when typing a **`Field`** value should cause the **`Form`** to submit. The option defaults to false and cannot be changed once the **`Form`** has been created.

The **`nativeEvents`** option can be used to specify that any associated native event should be fired when a 'hostedField:' prefixed **`Field`** event is fired (as documented in section A-20.3.8). For

example, when enabled if the 'hostedfield:mouseover' event is fired, then the native 'mouseover' event is also fired. The option defaults to false and cannot be changed once the **`Form`** has been created.

If not explicitly constructed, a **`Form`** object will be automatically instantiated and attached to the FORM DOM node as soon as any **`Field`** object is instantiated on a child DOM node.

Example **`Form`** construction is as follows:

```
var form = new window.hostedFields.classes.Form(document.forms[0],{
    // Auto setup the form creating all hosted fields (default)
    autoSetup: true,

    // Auto validate, tokenise and submit the form (default)
    autoSubmit: true,

    // Additional fields to tokenise
    tokenise: '.add-to-token',

    // Stylesheet selection
    stylesheets: '#hostedfield-stylesheet',

    // Optional field configuration (by type)
    fields: {
        any: {
            nativeEvents: true
        },
        cardNumber: {
            selector: $('#form2-card-number'),
            stylesheet: $('style.hostedform, style.hostedform-card-number')
        }
    },

    // Additional CSS classes
    classes: {
        invalid: 'error'
    }
});
```

Or using meta data on the HTML FORM element:

```
<form data-hostedfields='{"autoSetup":true,"autoSubmit":true,"tokenise":".add-to-
token","stylesheets":"#hostedfield-
stylesheet","fields":{"any":{"nativeEvents":true},"cardNumber":{"selector":"#form2-card-
number","stylesheet":"style.hostedform, style.hostedform-card-
number"}},"classes":{"invalid":"error"}}' method="post" novalidate="novalidate" lang="en">
<script>
var form = new window.hostedFields.classes.Form{document.forms[0]);
</script>
```

A-20.3.4 Form Methods

The follow methods are made available by the **`Form`** class:

`void autoSetup()`

Automatically setup the form by scanning the Form element for child nodes to control as Hosted Payment Fields. Child nodes are selected if they:

- have a `type` attribute with a `hostedfield:<type>` value *(INPUT nodes only)*.
- have a `data` attribute with a `hostedfield.<type>` property.
- match a DOM selector provided by the `fields.<type>.selector` option.

If multiple selection criteria are present, then they must all specify the same **`Field`** type or an exception is thrown.

This method is called during the **`Form`** construction unless the **`autoSetup`** option is false.

`void autoSubmit()`

Automatically handles any attempted FORM submission by checking the FORM's controls are valid by calling the **`validate()`** method; and then requesting the **`paymentToken`** using the **`getPaymentDetails()`** method; and finally adding the token to the forms fields using the **`addPaymentToken()`** method. Failure to validate or request the payment token will cause the form submission to be stopped.

You can affect the automatic submission stages by listening for events and preventing their default actions. The full list of events is documented in section A-20.3.5.

This method is attached to the FORM submit event during the **`Form`** construction unless the **`autoSubmit`** option is false, or the **`autoSubmit`** option is null and the **`autoSetup`** option is false.

If automatic submission is disabled, then you must react to the FORM's submit event and then request the **`paymentToken`** using the **`getPaymentDetails()`** method and ensure that the token is sent as part of the form's data.

`boolean addField(Field f)`

Add a hosted **`Field`** to the `Form`.

Returns true if successful, false otherwise.

`boolean delField(Field f)`

Remove a hosted **`Field`** from the **`Form`**.

Returns true if successful, false otherwise.

`promise validate(boolean submitting)`

Validate all **`Field`** values on the **`Form`**, either during submission or not.

Returns a `promise` that will be resolved when the validation is complete.

`object[] getInvalidElements()`

Get details about all invalid FORM controls (not just invalid hosted **`Field`** elements).

Returns an array of objects containing the following properties:

- **`element`** – DOM element.
- **`message`** – DOM elements `validationMessage` property or 'Invalid value'.
- **`label`** – associated LABEL text.
- **`field`** – **`Field`** instance (if DOM element is a hosted **`Field`**).

`object getValidationErrors()`

Get the validation errors for all invalid FORM controls (not just invalid hosted **`Field`** elements).

Returns an object whose properties are the associated labels, names or id of the invalid FORM controls and whose values are the error message for that control.

`promise getPaymentDetails(object tokenData, boolean validate)`

Gets the payment details, generating a **`paymentToken`** containing the hosted Field values; any values specified by the **`tokenise`** option; and any passed **`tokenData`**. The Form will be validated first if required.

Returns a `promise` that will be resolved when the payment details have been obtained, passing the details as an object containing the following properties:

- **`success`** – boolean true if successful, false otherwise.
- **`message`** – string containing message to display if not successful.
- **`errors`** – object containing details about invalid payment data.
- **`invalid`** – object as returned by **`getValidationErrors()`** method.
- **`paymentToken`** – string containing generated **`paymentToken`**.

`void addPaymentToken(string token)`

Add the payment token as the value of a Form child INPUT whose `name` is 'paymentToken', creating the control if needed. Any created control will be given a `type` of 'hidden'.

`void setMerchantID(string merchantID)`

Set the **`merchantID`** used by the payment form.

`void setStylesheet(string selector)`

Set the DOM selector used to select the stylesheet(s) used by the **`Form`**.

`object defaultFieldOptions(string type)`

Get any default field options specified via the **`fields`** option, resulting from the merger of its optional **`any`** and ***`<type>`*** properties.

Returns an object whose properties are the default options.

`void forceSubmit()`

Forcefully submit the FORM **`element`** as if a child submit button had been clicked.

`void reset()`

Reset all the **`Form`**, setting all **`Field`** values back to their initial values.

`void destroy()`

Destroys the **`Form,`** reverting its **`element`** back to its original state.

A-20.3.5 Form Events

The following events may be fired by the **`Form`** object and you can use these to hook into and modify the object's behaviour:

Event Name[1]	Description
`create`	Fired when a **`Form`** has been created.
`destroy`	Fired when a **`Form`** has been destroyed.
`presubmit`	Fired by the **`autoSubmit()`** method prior to handling the submission. You can prevent the handling of the submission and handle it yourself by calling the Events `preventDefault()` method.
`valid`	Fired by the **`autoSubmit()`** method if the FORM contains valid data prior to requesting the payment details. You can prevent the continued handling of the submission and handle it yourself by calling the Events `preventDefault()` method or by invalidating the FORM.
`submit-invalid`	Fired by the **`autoSubmit()`** method if the FORM contains invalid data prior to displaying the validity using the DOM `reportValidity()` method. You can prevent the `reportValidity()` call and display the validity yourself by calling the Events `preventDefault()` method.
`submit`	Fired by the **`autoSubmit()`** method prior to submitting the FORM. You can prevent the FORM from submitting by calling the Events `preventDefault()` method.
`error`	Fired by the **`autoSubmit()`** method if an exception is caught prior to displaying the error, using the JavaScript `alert()` function. You can prevent the `alert()` call and display the error yourself by calling the Events `preventDefault()` method.

[1] Event names are prefixed with the 'hostedform:' namespace not shown in the table.

The **`presubmit`**, **`valid`**, **`submit-invalid`**, **`submit`** and **`error`** events fired by the **`autoSubmit()`** method the payload is an object with the following properties:

- **`success`** – boolean false.
- **`message`** – error message if **`error`** otherwise null.
- **`invalid`** – result of **`getValidationErrors()`** method if **`Form`** invalid.
- **`submitting`** – boolean true.

.

A-20.3.6 Field Construction

The construction method can be used to prepare a HTML INPUT control as a Hosted Payment Field or to create a new field in HTML DIV container. The method signature is as follows:

```
Field(element, options)
```

The **element** parameter should be the DOM node of an existing INPUT or DIV tag.

The **options** parameter should be object containing one of more of the following optional properties:

- **type** – string containing the desired field type.
- **value** – string containing the initial value.
- **placeholder** – string containing any placeholder text.
- **style** – string containing any inline CSS styles.
- **stylesheet** – string containing DOM selector for any stylesheets to be used.
- **disabled** – boolean indicating if initially disabled.
- **required** – boolean indicating if the value is required.
- **readOnly** – boolean indicating if initially read only.
- **validity** – boolean or string indicating the initial validity.
- **locale** – string containing the desired locale.
- **classes** – object containing names of extra CSS classes to use.
- **submitOnEnter** – boolean indicating if the enter key should cause the form to submit.
- **nativeEvents** – boolean indicating that native browser events should be fired.
- **validationMessages** – object containing alternative validation messages.
  - **required** – string containing validation message to use when a value is required.
  - **invalid** – string containing validation message to use when a value is invalid.
- **format** – string containing select option format for date fields.
- **minYear** – integer containing minimum year (relative to current year) for date fields.
- **maxYear** – integer containing maximum year (relative to current year) for date fields.

Any **options** parameter will be merged with those provided via meta data supplied using `data-hostedfield` and/or `data-hostedfield-<option>` attributes, or via existing attributes or properties of the **element** or provided via the **getDefaultOptions()** method of the parent **Form**.

The **type** option can be used to specify the type of Hosted Payment Field required. It defaults to the value provided by any `type` attribute on the **element** (prefixed with the 'hostedfield:' namespace). The option cannot be changed once the **Field** has been created. Valid types are **cardDetails**, **cardNumber**, **cardCVV**, **cardExpiryDate**, **cardStartDate**, **cardIssueNumber**.

The **value** option can be used to specify any initial value that should be used by the **Field**. It defaults to the value provided by any `value` attribute or property on the **element**. Obviously, due to the purpose of the Hosted Payment Fields, any initial value is not wise for card number and CVV fields. The option can be changed at runtime by calling the **setValue()** method.

The **placeholder** option can be used to specify any initial text that should be used as a placeholder by the **Field**. It defaults to the value provided by any `placeholder` attribute or property on the **element**. When used with the **CardDetails** type **Field** the placeholder contains three parts separated by a pipe character, the first part contains the **cardNumber** placeholder, the second part contains the **cardExpiry** placeholder, and the third part contains the **cardCVV** placeholder. The option can be changed at runtime by calling the **setPlaceholder()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **style** option can be used to specify any initial inline CSS style that should be used by the **Field**. It defaults to the value provided by any `style` attribute or property on the **element**. The option can be changed at runtime by calling the **setStyle()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **stylesheet** option can be used to specify a DOM selector used to locate stylesheets that should be parsed for styles related to this **Field**. Refer to section on styling fields. The option can be changed at runtime by calling the **setStylesheet()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **disabled** option can be used to specify if the **Field** should be initially disabled. It defaults to the value provided by any `disabled` attribute or property on the **element**. The option can be changed at runtime by calling the **setDisabled()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **required** option can be used to specify if the **Field** value is required. It defaults to the value provided by any `required` attribute or property on the **element**. The option can be changed at runtime by calling the **setRequired()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **readOnly** option can be used to specify if the **Field** should be initially read-only. It defaults to the value provided by any `readOnly` attribute or property on the **element**. The option can be changed at runtime by calling the **setReadOnly()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **validity** option can be used to specify if the **Field** should be initially marked as invalid. It defaults to the value provided by any `validity` property on the **element**. The option can be changed at runtime by calling the **setValidity()** method or by altering the options using the jQuery **hostedForm()** plugin method.

The **locale** option can be used to specify the language that should be used by the **Field**. It defaults to the value provided by any `lang` attribute or property on the **element** or closest ancestor. The option cannot be changed once the **Field** has been created.

The **classes** options can be used to specify additional CSS class names to add in addition to the default classes documented in section A-20.3.9. The value is an object whose properties are the default class name and whose values are a string containing the additional class name(s) to use. This option will be merged with any classes option provided to the **Form** constructor. The option cannot be changed once the **Form** has been created.

The **`submitOnEnter`** option can be used to specify if pressing the enter key when typing the **`Field`** value should cause the **`Form`** to submit. The option defaults to false and cannot be changed once the **`Field`** has been created.

The **`nativeEvents`** option can be used to specify that any associated native event should be fired when a 'hostedfield:' prefixed event is fired. Events are documented in section A-20.3.8. For example, when enabled if the 'hostedfield:mouseover' event is fired then the native 'mouseover' event is also fired. The option defaults to false and cannot be changed once the **`Field`** has been created.

The **`validationMessages`** option can be used to specify alternative validation messages that should be displayed when a value is required or invalid. The option defaults to suitable messages depending on the locale and cannot be changed once the **`Field`** has been created.

The **`dropdown`** option can be used to specify that a **cardStartDate** or **cardExpiryDate `Field`** should be displayed as a pair of select controls to select the month and year, otherwise the month and year are entered via a formatted input box instead. The option defaults to false and cannot be changed once the **`Field`** has been created.

The **`format`** option can be used in conjunction with the **`dropdown`** option to specify the format used to display the month and year in the dropdowns. The month and year parts of the format are separated by a pipe character. The option defaults to 'N – M | Y' (eg '01 – January | 2020') and cannot be changed once the **`Field`** has been created.

The following formatting characters are understood:

- **`n`** – month number (no zero prefixing).
- **`N`** – month number (zero prefixed to two digits when required).
- **`m`** – short month name (eg Jan, Feb, Mar)
- **`M`** – long month name (eg January, February, March)
- **`y`** – two digit year number.
- **`Y`** – four digit year number.

The **`minYear`** and **`maxYear`** options can be used in conjunction with the **`dropdown`** option to specify the minimum and maximum years that are included in the year dropdown. The option defaults to minus 20 to zero for a **cardStartDate `Field`** or zero to plus 20 for a **cardExpiryDate `Field`** and cannot be changed once the **`Field`** has been created.

Example **`Field`** construction is as follows:

```
var field = new window.hostedFields.classes.Field(document.forms[0].elements[0], {
    // Field type
    type: 'cardNumber',

    // Stylesheet selection
    stylesheets: '#hostedfield-stylesheet',

    // Additional CSS classes
    classes: {
        invalid: 'error'
    }
});
```

Or using meta data on the HTML INPUT element:

```
<input type="hostedfield:cardNumber" data-hostedfields='{"stylesheet":"style.hostedform, style.hostedform-card-number"}},"classes":{"invalid":"error"}}'>
<script>
var field = new window.hostedFields.classes.Field(document.forms[0].elements[0]);
</script>
```

A-20.3.7 Field Methods

The follow methods are made available by the **`Field`** class:

`promise validate()`

Validate the **`Field`** value. This will normally be called automatically when the **`Field`** loses focus or the form is submitted, or when an invalid value is modified.

Returns a `promise` that will be resolved when the validation is complete.

`boolean isEmpty()`

Check if the **`Field`** has a value.

Returns true if the field has a value, false otherwise.

`boolean isComplete()`

Check if the **`Field`** has a complete, but not necessarily valid, value. This is mainly used by compound fields such as **cardDetails**, **cardExpiryDate**, **cardStartDate**, which contain multiple input controls and are deemed complete when all their required input controls have values.

Returns true if the value is complete, false otherwise.

`void setStyle()` / **`string getStyle()`**

Set or gets the field's inline CSS style data.

Returns void when setting, or a CSS style string when getting.

`void setStylesheet(string selector)` / **`string getStylesheet()`**

Sets or gets the DOM selector used to select the stylesheet(s) used by the **`Field`**. When setting, the stylesheets are parsed and applied to the **`Field`**.

Returns void when setting, or a DOM selector string when getting.

`void setPlaceholder(string text)` / **`string getPlaceholder()`**

Sets or gets the placeholder text to be shown when the **`Field`** has no value.

When used with the **CardDetails** type **`Field`** the placeholder contains three parts separated by a pipe character, the first part contains the **cardNumber** placeholder, the second part contains the **cardExpiry** placeholder, and the third part contains the **cardCVV** placeholder.

Returns void when setting, or a text string when getting.

`void setDisabled(boolean disabled)` / **`string getDisabled()`**

Sets or gets the disabled state of the **`Field`**. When disabled, the field will be greyed out and not be focusable and thus will not react to any input events.

A disabled **`Field`** will have the ‘hf-disabled’ class added otherwise the ‘hf-enabled’ class is added.

Returns void when setting, or a boolean representing the state when getting.

`void setRequired(boolean required)` / **`string getRequired()`**

Sets or gets the required state of the **`Field`**. When required, the field will be invalid if it contains no value or a blank value.

A required **`Field`** will have the ‘hf-required’ class added otherwise the ‘hf-optional’ class is added.

Returns void when setting, or a boolean representing the state when getting.

`void setReadOnly(boolean read_only)` / **`string getRequired()`**

Sets or gets the read-only state of the **`Field`**. When read-only, the field will be not be focusable and thus will not react to any input events.

A read-only **`Field`** will have the ‘hf-readonly’ class added otherwise the ‘hf-readwrite’ class is added.

Returns void when setting, or a boolean representing the state when getting.

`void setFocused(boolean focused)`

Moves the browser's focus to the **`Field`**. When focused, the field will react input events.

A focused **`Field`** will have the 'hf-focus' class added otherwise the 'hf-blur' class is added.

Returns void when setting, or a boolean representing the state when getting.

`void setValidity(string validity)` / **`string getValidity()`**

Sets or gets the validity of the **`Field`**. When valid, the validity will be true or a blank string. When invalid, the validity will be an error message explaining the reason the value is invalid.

When used with the **CardDetails** type **`Field`** the error message contains three parts separated by a pipe character, the first part contains the **cardNumber** value, the second part contains the **cardExpiry** value, and the third part contains the **cardCVV** value.

A valid **`Field`** will have the 'hf-valid' and 'hf-user-valid' classes added otherwise the 'hf-invalid' and 'hf-user-invalid' classes are added.

Returns void when setting, or an error message string when getting.

`void setValue()` / **`string getValue()`**

Set or gets the **`Field`** value. Because Hosted Payment Fields are designed for the entry of sensitive payment details, then these methods are not normally used. There is no means to retrieve the actual sensitive data and so any returned value will be an empty string if the field has no value or a single asterisk if the field has a value.

When used with the **CardDetails** type **`Field`** the value contains three parts separated by a pipe character, the first part contains the **cardNumber** value, the second part contains the **cardExpiry** value, and the third part contains the **cardCVV** value.

Returns void when setting, or a mask string when getting.

`void getState()`

Get the current state of the **`Field`** as an object with the following boolean properties:

- **`isReady`** – the **`Field`** has been created, initialised and is ready for use.
- **`isValid`** – the value is valid (refer to the **`setValidity()`** method).
- **`isEmpty`** – the value is empty (refer to the **`isEmpty()`** method).
- **`isComplete`** – the value is complete (refer to the **`isComplete()`** method).
- **`isDisabled`** – the value is complete (refer to the **`setDisabled()`** method).
- **`isRequired`** – the value is complete (refer to the **`setRequired()`** method).
- **`isReadOnly`** – the value is complete (refer to the **`setReadOnly()`** method).

Returns an object containing the states.

`void reset()`

Reset **`Field`** value back to the initial value.

`void destroy()`

Destroys the **`Form,`** reverting its **`element`** back to its original state.

Note: A field's options or properties cannot be changed while a field is initialising, that is between construction and firing of the 'ready' event. Attempts to change field options or properties before this will be ignored.

A-20.3.8 Field Events

The following events may be fired by the **`Field`** object, and you can use these to hook into and modify the object's behaviour:

Event Name[1]	Description
`create`	Fired when a **`Field`** has been created.
`destroy`	Fired when a **`Field`** has been destroyed.
`ready`	Fired when a **`Field`** style is has finished initialising and is ready.
`style`	Fired when a **`Field`** style is changed.
`autofill`	Fired when a **`Field`** has a value auto filled by the browser.
`autofillcancel`	Fired when a **`Field`** has an auto filled value removed.
`valid`	Fired when a **`Field`** is checked for validity and passes the check.
`invalid`	Fired when a **`Field`** is checked for validity and fails the check.
`uservalid`	Fired when the valid event is fired but only after user interaction has occurred, such as focusing a **`Field`**, leaving a **`Field`** or attempting to submit a **`Form`**.
`userinvalid`	Fired when the invalid event is fired but only after user interaction has occurred, such as focusing a **`Field`**, leaving a **`Field`** or attempting to submit a **`Form`**.
`disabled`	Fired when a **`Field`** changes to disabled.
`enabled`	Fired when a **`Field`** changes from disabled.
`required`	Fired when a **`Field`** changes to required.
`optional`	Fired when a **`Field`** changes from required.
`readonly`	Fired when a **`Field`** changes to read-only.
`readwrite`	Fired when a **`Field`** changed from read-only.
`focus`	Fired when a **`Field`** receives focus.
`blur`	Fired when a **`Field`** loses focus.
`mouseenter`	Fired when a pointing device is moved into the **`Field`**.
`mouseleave`	Fired when a pointing device is moved out of the **`Field`**.
`mouseover`	Fired when a pointing device is moved into the **`Field`**.
`mouseout`	Fired when a pointing device is moved out of the **`Field`**.
`mousemove`	Fired when a pointing device is moved over the **`Field`**.
`keydown`	Fired when a key is pressed in the **`Field`**.
`keyup`	Fired when a key is released in a **`Field`**.

Event Name[1]	Description
`keypress`	Fired when a key except Shift, Fn, CapsLock is in a pressed position in a **`Field`**.
`change`	Fired when an alteration to the value of a **`Field`** is committed by the user.
`input`	Fired when the value of a **`Field`** is changed.

[1] Event names are prefixed with the 'hostedfield:' namespace not shown in the table.

A-20.3.9 Field CSS Classes

The following CSS class names will be added to a **`Field`** object depending on its state and you can use these to style the object as required:

Event Name	Description
`hostedfield`	Present on all **`Field elements`**.
`hf-autofill`	Present when the value was auto filled by the browser.
`hf-invalid`	Present when in the invalid state.
`hf-valid`	Present when in the valid state.
`hf-user-invalid`	Present when in the invalid state and user interaction has occurred, such as focusing a **`Field`**, leaving a **`Field`** or attempting to submit a **`Form`**.
`hf-user-valid`	Present when in the valid state and user interaction has occurred, such as focusing a **`Field`**, leaving a **`Field`** or attempting to submit a **`Form`**.
`hf-disabled`	Present when in the disabled state.
`hf-enabled`	Present when not in the disabled state.
`hf-required`	Present when in the required state.
`hf-optional`	Present when not in the required state.
`hf-readonly`	Present when in the read-only state.
`hf-readwrite`	Present when not in the read-only state.
`hf-focus`	Present when in the focused state.
`hf-blur`	Present when not in the focused state.
`hf-empty`	Present when in the empty state.
`hf-complete`	Present when in the complete state.
`hf-hover`	Present when a pointing device is over the **`Field`**.
`hf-placeholder-shown`	Present when the placeholder text is displayed.

In addition to these class names, the **`Field`** will add any corresponding class names provided by the **`classes`** option provided when the **`Field`** is constructed.

For example, if the **`Field`** is constructed with a **`classes`** option as follows '`{disabled: 'text-blur text-grey', enabled: 'text-normal'}`', then the 'text-blur' and 'text-grey' class names will be present whenever the 'hf-disabled' class is present and the 'text-normal' class name will be present whenever the 'hf-enabled' class name is present.

A-20.3.10 Field Styling

The Hosted Payment Fields are styled using CSS as normal.

However, styles have to be transferred from your website to the controls served from our website, therefore styles must be isolated and easily identifiable. To aid with identification, all styles intended for a **`Field`** must contain the 'hostedfield' class name in their selector or '-hostedfield' extension on any id in the selector.

As a website may contain lots of stylesheets, a **`Field`** cannot be expected to parse every stylesheet present on the page and therefore it only parses those selected using the stylesheets construction option or using the **`setStylesheet()`** method. By default, this is any stylesheet referenced via a <link> tag or <style> tag with the 'hostedfield' class name: ie any HTML node that matches the following DOM selector 'link.hostedfield[rel=stylesheet], style.hostedfield'.

CSS styles using the **`Field`** state classes, pseudo classes and pseudo elements are supported as follows:

- :focus
- .hf-focus
- :hover
- .hf-hover
- :enabled
- .hf-enabled
- :disabled
- .hf-disabled
- :valid
- .hf-valid
- :invalid
- .hf-invalid
- :user-valid
- .hf-user-valid
- :user-invalid
- .hf-user-invalid
- :required
- .hf-required
- :optional
- .hf-optional
- :empty
- .hf-empty
- :complete
- .hf-complete
- :autofill
- .hf-autofill
- :placeholder-shown
- .hf-placeholder-shown
- :readonly
- .hf-readonly
- :readwrite
- .hf-readwrite
- :-webkit-auto-fill
- .hf-icon
- ::placeholder
- ::-moz-placeholder
- ::-webkit-input-placeholder
- ::-ms-input-placeholder

The styles can contain any valid CSS rules and will be used to style both the public elements and internal private elements. For security only, styles that relate to the textual representation of the **`Field`** are passed to the internal private elements. This includes styles such as colours, font weights and text decorations. At present, it is not possible to specify custom fonts as they would require the font files to be available on our servers.

The following styles can be used to style the **`Field`** internal private elements:

- caret-color
- color
- cursor
- direction
- fill
- filter
- font
- font-family
- font-feature-settings
- font-kerning
- font-language-override
- font-size
- font-size-adjust
- font-smooth
- font-stretch
- font-style
- font-synthesis
- font-variant
- font-variant-alternates
- font-variant-caps
- font-variant-east-asian
- font-variant-ligatures
- font-variant-numeric
- font-variant-position
- font-weight
- letter-spacing
- line-height
- stroke
- text-align
- text-decoration
- text-decoration-color
- text-decoration-line
- text-decoration-style

- text-emphasis
- text-emphasis-color
- text-emphasis-position
- text-emphasis-style
- text-indent
- text-rendering
- text-shadow
- text-transform
- text-underline-position
- -moz-osx-font-smoothing
- -webkit-font-smoothing
- -webkit-text-fill-color

The '.hf-icon' class name can be used to target the icon sub element in a **`cardDetails Field`**.

Individual controls can be targeted by using DOM ids, which will have a '-hostedfield' extension added to the DOM id of the original **`element`**.

It is advisable to keep CSS selectors and rules as simple as possible to avoid styling errors caused by a failure to parse and filter the rules.

The following list are the best web safe fonts for use with any 'font' or 'font-family' style as these fonts should be available on most web browsers:

- Arial (sans-serif)
- Verdana (sans-serif)
- Helvetica (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Example stylesheet:

```
<style class="hostedfield">
    /*
     * Style hosted field internals
     * - only accept font, foreground and background styling
     */

    /* Copy of Bootstrap styles */
    .hostedfield:disabled {
        cursor: not-allowed;
        background-color: #eee;
        opacity: 1;
    }

    /* Change text to red when invalid */
    .form-control:invalid,
    .hostedfield:invalid {
        border-color: #a94442 !important;
        color: #a94442 !important;
    }

    /* Change text to light grey when readonly */
    .form-control:readonly,
    .hostedfield:readonly {
        color: lightgrey !important;
    }

    /* Emulate webkit auto fill style */
    .form-control.hf-autofill,
    .hostedfield.hf-autofill {
        background-color: rgb(250, 255, 189) !important;
        background-image: none !important;
        color: rgb(0, 0, 0) !important;
    }

    /* Add pink placeholder */
    .form-control::placeholder,
    .hostedfield::placeholder {
        color: pink;
    }

    /* Show hovering over the control */
    .form-control.hf-hover,
    .hostedfield.hf-hover {
        font-style: italic;
    }

    /* Style by id (hosted field will have '-hostedfield' appended to the id) */
    #form1-card-details.hostedfield, #form1-card-details-hostedfield {
        color: blue;
    }

</style>
```

A-20.3.11 jQuery Plugin

The script will extend the jQuery object with its own plugin methods to allow easy access to **`Form`** and **`Field`** objects attached to an **`element`** as follows:

```
$(element).hostedForm(options);
$(element).hostedForm('instance');
$(element).hostedForm('options', options);
$(element).hostedForm(method, parameters);
$(element).hostedForm('destroy');

$(element).hostedField(options);
$(element).hostedField('instance');
$(element).hostedField('options', options);
$(element).hostedField(method, parameters);
$(element).hostedField('destroy');
```

The script will also add a ':hostedfield' pseudo selector allowing **`Field`** elements to be selected using the following example notation:

```
$('INPUT:hostedfield')
```

:

A-21 Example HTTP Requests

A-21.1 Hosted Integration

A-21.1.1 Transaction Request HTTP Headers

The following HTTP headers are sent for transaction request:

HTTP Header	Mandatory	Description
`content-type`	**Y**	Content type of the request. This must be 'application/x-www-form-urlencoded', A charset parameter is optional and any non UTF-8 request will be converted to UTF-8.

A-21.1.2 Transaction Response HTTP Headers

The following HTTP headers are received for a transaction response:

HTTP header	Description
`Status`	HTTP status header. Possible value are: **200** – Transaction request processed **500** – Internal Server Error **503** – Service Temporarily Unavailable
`content-type`	Content type of the response. This will be 'application/x-www-form-urlencoded'

A-21.1.3 Submission Example

The following shows an example of a transaction request:

```
HTTP/1.1 200 OK
POST /hosted/ HTTP/1.1
Host: gateway.example.com
Accept: */*
Content-Length: 314
Content-Type: application/x-www-form-urlencoded

merchantID=100001&action=SALE&type=1&currencyCode=826&countryCode=826&amount=680&transactionUnique=5de651c7c5379&orderRef=Test+Transaction&redirectURL=https%3A%2F%2Fmyshop.com&signature=ba12b0766a3412782448f154be15e8f61eea390387b1b23d4688c82c9f28f81df593de5890426546cca365943cc7b5c4897c9721b663a0e17680e1e796f1ad55
```

The following shows an example of a transaction response:

```
HTTP/1.1 200 OK
Date: Tue, 01 Jan 2019 09:30:45 GMT
Server: Apache/2.4.23 (Win64) OpenSSL/1.0.2k-fips PHP/5.4.12
Vary: Host
X-Powered-By: PHP/5.4.12
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 4129
Content-Type: text/html

<!DOCTYPE html>
<html>
--- Hosted Payment Page HTML Removed ---
</html>
```

A-21.2 Direct Integration

A-21.2.1 Transaction Request HTTP Headers

The following HTTP headers are sent for transaction request:

HTTP Header	Mandatory	Description
`content-type`	**Y**	Content type of the request. This must be 'application/x-www-form-urlencoded', A charset parameter is optional and any none UTF-8 request will be converted to UTF-8.

A-21.2.2 Transaction Response HTTP Headers

The following HTTP headers are received for a transaction response:

HTTP header	Description
`Status`	HTTP status header. Possible value are: 200 – Hosted Payment Form returned 500 – Internal Server Error 503 – Service Temporarily Unavailable
`content-type`	Content type of the response. This will be 'text/html'

A-21.2.3 Submission Example

The following shows an example of a transaction request:

```
POST /direct/ HTTP/1.1
Host: gateway.example.com
Accept: */*
Content-Length: 397
Content-Type: application/x-www-form-urlencoded

merchantID=100001&action=SALE&type=1&currencyCode=826&countryCode=826&amount=680&transactionUnique=5de65b552499e&orderRef=Test+Transaction&cardNumber=4929+4212+3460+0821&cardCVV=356&cardExpiryDate=1219&threeDSRequired=N&avscv2CheckRequired=N&duplicateDelay=0&signature=06b01e06c8fc761943d676d5f3aa2e9264758fed72e7bcb058a2a35cf23e8e45403099537bb0363054d6bc8ea951ce1ad86e582dbf0b435855b9c97507fcf844
```

The following shows an example of a transaction response:

```
HTTP/1.1 200 OK
Date: Tue, 01 Jan 2019 09:30:45 GMT
Server: Apache/2.4.23 (Win64) OpenSSL/1.0.2k-fips PHP/5.4.12
Vary: Host
X-Powered-By: PHP/5.4.12
Content-Length: 2449
Content-Type: text/html

merchantID=100001&threeDSEnabled=Y&avscv2CheckEnabled=Y&riskCheckEnabled=N&caEnabled=Y&rtsEnabled=Y&cftEnabled=Y&threeDSCheckPref=not+known%2Cnot+checked%2Cauthenticated%2Cattempted+authentication&cv2CheckPref=matched&addressCheckPref=not+known%2Cnot+checked%2Cmatched%2Cpartially+matched&postcodeCheckPref=not+known%2Cnot+checked%2Cmatched%2Cpartially+matched&cardCVVMandatory=Y&riskCheckPref=not+known%3Dfinished%2Cnot+checked%3Ddecline2%2Capprove%3Dcontinue%2Cdecline%3Ddecline1%2Creview%3Ddecline2%2Cescalate%3Dfinished&notifyEmail=an.operator%40merchant.com&customerReceiptsRequired=Y&merchantCategoryCode=6013&surchargeEnabled=Y&surchargeRequired=N&surchargeRules%5B0%5D%5BcardType%5D=CC&surchargeRules%5B0%5D%5Bsurcharge%5D=10.1235&surchargeRules%5B1%5D%5BcardType%5D=CC&surchargeRules%5B1%5D%5Bcurrency%5D=GBP&surchargeRules%5B1%5D%5Bsurcharge%5D=2.5000&surchargeRules%5B2%5D%5BcardType%5D=VC&surchargeRules%5B2%5D%5Bsurcharge%5D=3.5000&surchargeRules%5B3%5D%5BcardType%5D=VC&surchargeRules%5B3%5D%5Bcurrency%5D=GBP&surchargeRules%5B3%5D%5Bsurcharge%5D=4.5000&surchargeRules%5B4%5D%5BcardType%5D=DD&surchargeRules%5B4%5D%5Bsurcharge%5D=5.5000&action=SALE&type=1¤cyCode=826&countryCode=826&amount=680&transactionUnique=5de65b552499e&orderRef=Test+Transaction&cardExpiryDate=1219&threeDSRequired=N&avscv2CheckRequired=N&duplicateDelay=0&requestID=5de65b562496f&responseCode=0&responseMessage=AUTHCODE%3A347414&state=captured&requestMerchantID=100001&processMerchantID=100001&paymentMethod=card&cardType=Visa+Credit&cardTypeCode=VC&cardScheme=Visa+&cardSchemeCode=VC&cardIssuer=BARCLAYS+BANK+PLC&cardIssuerCountry=United+Kingdom&cardIssuerCountryCode=GBR&cardFlags=8323072&cardNumberMask=492942%2A%2A%2A%2A%2A%2A0821&cardNumberValid=Y&xref=19120312NG55CM51QH35JRL&cardExpiryMonth=12&cardExpiryYear=19&authorisationCode=347414&transactionID=10018201&responseStatus=0×tamp=2019-12-03+12%3A55%3A52&amountApproved=680&amountReceived=680&amountRetained=680&avscv2ResponseCode=244100&avscv2ResponseMessage=SECURITY+CODE+MATCH+ONLY&avscv2AuthEntity=merchant+host&cv2Check=matched&addressCheck=not+matched&postcodeCheck=not+matched&notifyEmailResponseCode=0&notifyEmailResponseMessage=Email+successfully+queued&vcsResponseCode=0&vcsResponseMessage=Success+-+no+velocity+check+rules+applied&currencyExponent=2&signature=e5c65e5d0340e0ec0de8782affcb6caba2e4d202a6873a1677ddbf6415cb1dd52cc597e43c758b233afd121367d300a57d0faade7abf6b4b88a1a1b974e55d33
```

A-21.3 Batch Integration

A-21.3.1 Submission Request HTTP Headers

The following HTTP headers are sent for batch submission request:

HTTP Header	Mandatory	Description
`content-type`	**Y**	Content type of the batch request. This must be 'multipart/mixed' and contain a boundary parameter to separate each transaction request. A charset parameter is optional and any none UTF-8 request will be converted to UTF-8.
`content-encoding`	**N**	Optional content encoding applied to the request. The value should be a comma separated list of one or more: **x-gzip**, **gzip**, **base64**.
`authorization`	**N**	Optional username and password to authenticate the submitter

The following HTTP headers are sent on each individual part request:

HTTP Header	Mandatory	Description
`content-type`	**Y**	Content type of the individual request. This must be 'application/x-www-form-urlencoded', A charset parameter is optional and any none UTF-8 request will be converted to UTF-8.
`content-encoding`	**N**	Optional content encoding applied to the request. The value should be a comma separated list of one or more: **x-gzip**, **gzip**, **base64**.
`content-id`	**N**	Optional identifier for each individual transaction with the batch. The Gateway will return this identifier in the submission response. If not sent, the Gateway will generate a unique identifier for each transaction.

A-21.3.2 Submission Response HTTP Headers

The following HTTP headers are received for batch submission response:

HTTP header	Description
`status`	HTTP status header. Possible value are: **200** – Batch submission status response ok **201** – Batch submission received and stored **400** – Batch submission invalid **401** – Unauthorised (none or incorrect credentials) **405** – HTTP method was not POST/PUT or GET **500** – Internal Gateway error
`location`	URL to use to monitor the status of the batch. A unique batch reference number will be provided in the URL in the format: XXXX-XXXX-XXXX-XXXX (eg 1A23-B4C5-DEF6-G7HI) This reference number is used to request information about the status of a batch via HTTP GET requests to the URL endpoint as outlined in section 1.3.3.
`x-p3-token`	If user authentication was sent in the initial request, this header will contain a token that can be used for future requests for the status of the batch instead of having to use a username/password.
`content-type`	Content type of the HTTP batch request. This will be 'multipart/mixed' and contain a boundary parameter to separate each transaction request.

The following HTTP headers are received on each individual part response:

HTTP header	Description
`content-type`	Content type of the individual request. This will be 'application/x-www-form-urlencoded', A charset parameter is optional and any none UTF-8 request will be converted to UTF-8.
`content-id`	The content ID sent in the initial request as outlined in A-21.3.1. If no content-id header was sent, the Gateway will return a unique content ID per transaction.
`x-transaction-id`	The Gateway transaction ID. This will be empty if the transaction is currently pending in this stage.
`x-transaction-response`	A message containing the current status of the transaction. Possible value are: **skipped** – insufficient permissions to view transaction **pending** – queued for processing **success** – (Response Message) **failure** – (Response Message)

A-21.3.3 Status Request HTTP Headers

The following HTTP headers are used during a batch status request:

HTTP Header	Mandatory	Description
authorization	**Y**	Mandatory username and password to authenticate the submitter

A-21.3.4 Status Response HTTP Headers

The batch status response is identical to the submission status response as documented in section A-21.3.2.

A-21.3.5 Submission Example

The following shows an example of a batch submission request:

```
PUT /batch/?validate=0 HTTP/1.1
Authorization: Basic bmljay50dXJuZXI6dGVzdGluZzI=
Host: gateway.example.com
Accept: */*
Content-type: multipart/mixed; charset=UTF-8; boundary=5de63a42507a9
Content-length: 1404

--5de63a42507a9
Content-Id: TX5de63a42507ac
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

merchantID=100001&action=SALE&type=1&currencyCode=826&countryCode=826&amount=680&transactionUnique=5de63a42507ac&orderRef=Test+Transaction&cardNumber=4929+4212+3460+0821&cardExpiryDate=1219&duplicateDelay=0&signature=3cd686fdd40449ef33534baa62732c95fc127ff591fae3b5b611ccb38573ad921d199396e27cffd14faa4f46df8dde310252920fd1b33607b029b9b6ff669e2b

--5de63a42507a9
Content-Id: TX5de63a42af062
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

merchantID=100001&action=SALE&type=1&currencyCode=826&countryCode=826&amount=681&transactionUnique=5de63a42af062&orderRef=Test+Transaction&cardNumber=4929+4212+3460+0821&cardExpiryDate=1219&duplicateDelay=0&signature=55f411d40954be7f7089e84fe489438f09fc1b37c0964e46b0fab8bdcb44e13ed3ea11b9deb9da89a6d7b45133709a126bd3581f6329bf888b83231184597231

--5de63a42507a9
Content-Id: TX5de63a42ca9cd
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

merchantID=100001&action=SALE&type=1&currencyCode=826&countryCode=826&amount=682&transactionUnique=5de63a42ca9cd&orderRef=Test+Transaction&cardNumber=4929+4212+3460+0821&cardExpiryDate=1219&duplicateDelay=0&signature=c296266cb9bc8082957c700da9651d98add176dd8bd62eb3b7098566c7d8e23a3426b776de815e99149c6681978b1addedac762339563732d8a449b6cca3a3c2

--5de63a42507a9--
```

The following shows an example of a batch submission response:

```
HTTP/1.1 201 Created
Date: Tue, 01 Jan 2019 09:30:45 GMT
Server: Apache/2.4.23 (Win64) OpenSSL/1.0.2k-fips PHP/5.4.12
X-Powered-By: PHP/5.4.12
x-p3-token: YTo1OntzOjc6InZlcnNpb24iO3M6ODoiUDNUSy8yLjAiO3M6NzoicHVycG9zZSI7czo0OiJhdXRoIjtzOjc6ImNyZWF0b3IiO3M6NToiQkFUQ0giO3M6NzoiY3JlYXRlZCI7aToxNTc1MzY5Mjg1O3M6NzoiZXhwaXJlcyI7aToxNTc1MzcyODg1O30.czozOiI2MjkiOw.zdfxxXYtC2Wc4yyk-lEos-wZ99pEJtPGYpXR4KCiWW_56nmOysarOaMucrWPIt-NzwFzgq3-7u4Ud6uYkQcWBQ
Location: /batch/2D6D-AC2C-BF55-2A8C
Content-disposition: attachment; filename="batch-2D6D-AC2C-BF55-2A8C"
Content-Length: 1857
Content-Type: multipart/mixed; charset=UTF-8; boundary=5de63a5c1a071

Transaction 'TX5de63a42507ac' - pending - queued for processing
Transaction 'TX5de63a42af062' - pending - queued for processing
Transaction 'TX5de63a42ca9cd' - pending - queued for processing

--5de63a5c1a071
Content-Id: TX5de63a42507ac
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Transaction-ID:
X-Transaction-Response: pending - queued for processing

merchantID=100001&action=SALE&type=1¤cyCode=826&countryCode=826&amount=680&transactionUnique=5de63a42507ac&orderRef=Test+Transaction&cardNumber=492942%2A%2A%2A%2A%2A%2A0821&cardExpiryDate=1219&duplicateDelay=0&signature=0384bbf6ca0fc153e1e27a0cfc51f3b1cd1c2cff7a49aa4e9439bba38262183e9ac7d156f218eba1ef8d04f9e6a7fa6fbc9c2b3ab990c70e06dc7c6923e5b27b

--5de63a5c1a071
Content-Id: TX5de63a42af062
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Transaction-ID:
X-Transaction-Response: pending - queued for processing

merchantID=100001&action=SALE&type=1¤cyCode=826&countryCode=826&amount=681&transactionUnique=5de63a42af062&orderRef=Test+Transaction&cardNumber=492942%2A%2A%2A%2A%2A%2A0821&cardExpiryDate=1219&duplicateDelay=0&signature=1e13e23c2b90a30f4403d604ac20302b5504b886b0b5c9ace0764fc8d966d120f5a1beca975805292780c22953b4e6ca71f67f499804f19d2718518463a598c4

--5de63a5c1a071
Content-Id: TX5de63a42ca9cd
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Transaction-ID:
X-Transaction-Response: pending - queued for processing

merchantID=100001&action=SALE&type=1¤cyCode=826&countryCode=826&amount=682&transactionUnique=5de63a42ca9cd&orderRef=Test+Transaction&cardNumber=492942%2A%2A%2A%2A%2A%2A0821&cardExpiryDate=1219&duplicateDelay=0&signature=c456aa211f8e3e568a40051bfd38406be02566fcd72d3bb1547f4d43e75db1d069eaa4158aa035337cac084633df945a13471db6b1a3fcd6c0749626d9bc0044

--5de63a5c1a071--
```

A-22 Example Integration Code

The follow section provides samples of how to integrate with the Gateway using the PHP scripting language to communicate directly with the API without the use of any our SDKs.

A-22.1 Hosted Integration

A-22.1.1 Sale Transaction

The following example PHP code shows how to send a SALE transaction:

```php
<?PHP

// Signature key entered on MMS. The demo account is fixed to this value,
$key = 'Circle4Take40Idea';

// Gateway URL
$url = 'https://gateway.example.com/hosted/';


if (!isset($_POST['responseCode'])) {
    // Send request to gateway

    // Request
    $req = array(
        'merchantID' => '100001',
        'action' => 'SALE',
        'type' => 1,
        'countryCode' => 826,
        'currencyCode' => 826,
        'amount' => 1001,
        'orderRef' => 'Test purchase',
        'transactionUnique' => uniqid(),
        'redirectURL' => ($_SERVER['HTTPS'] == 'on' ? 'https' : 'http') . '://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'],
    );

    // Create the signature using the function called below.
    $req['signature'] = createSignature($req, $key);

    echo '<form action="' . htmlentities($url) . '" method="post">' . PHP_EOL;

    foreach ($req as $field => $value) {
        echo '    <input type="hidden" name="' . $field . '" value="' . htmlentities($value) . '">' . PHP_EOL;
    }

    echo '    <input type="submit" value="Pay Now">' . PHP_EOL;
    echo '</form>' . PHP_EOL;

// Check the return signature
    if (!$signature || $signature !== createSignature($res, $key)) {
        // You should exit gracefully
        die('Sorry, the signature check failed');
    }

    // Check the response code
    if ($res['responseCode'] === "0") {
        echo "<p>Thank you for your payment.</p>";
    } else {
        echo "<p>Failed to take payment: " . htmlentities($res['responseMessage']) .  "</p>";
    }

}

// Function to create a message signature
```

```php
function createSignature(array $data, $key) {
    // Sort by field name
    ksort($data);

    // Create the URL encoded signature string
    $ret = http_build_query($data, '', '&');

    // Normalise all line endings (CRNL|NLCR|NL|CR) to just NL (%0A)
    $ret = str_replace(array('%0D%0A', '%0A%0D', '%0D'), '%0A', $ret);

    // Hash the signature string and the key together
    return hash('SHA512', $ret . $key);
}

?>
```

A-22.2 Direct Integration

A-22.2.1 Sale Transaction (with 3-D Secure)

The following example PHP code shows how to send a SALE transaction with support for 3-D Secure:

```php
<?PHP

// Signature key entered on MMS. The demo account is fixed to this value,
$key = 'Circle4Take40Idea';

// Gateway URL
$url = 'https://gateway.example.com/direct/';

// Setup PHP session as use it to store data between 3DS steps
if (isset($_GET['sid'])) {
    session_id($_GET['sid']);
}

session_start();

// Compose current page URL (removing any sid and acs parameters)
$pageUrl = ((isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] == 'on') ? 'https://' : 'http://')
    . $_SERVER['SERVER_NAME'] . ($_SERVER['SERVER_PORT'] != '80' ? ':' . $_SERVER['SERVER_PORT'] : '')
    . preg_replace('/(sid=[^&]+&?)|(acs=1&?)/', '', $_SERVER['REQUEST_URI']);

// Add back the correct sid parameter (used as session cookie may not be passed when the page is redirected fr
om an IFRAME)
$pageUrl .= (strpos($pageUrl, '?') === false ? '?' : '&') . 'sid=' . urlencode(session_id());


// If ACS response into the IFRAME then redirect back to parent window
if (!emptyempty($_GET['acs'])) {
    echo silentPost($pageUrl, array('threeDSResponse' => $_POST), '_parent');
    exit();
}

if (!isset($_POST['threeDSResponse'])) {
    // Initial request

    // Gather browser info - can be done at any time prior to the checkout
    if (!isset($_POST['browserInfo'])) {
        echo collectBrowserInfo();
        exit();
    }

    // Direct Request
    $req = array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
        'amount' => 1001,
        'cardNumber' => '4012001037141112',
        'cardExpiryMonth' => 12,
        'cardExpiryYear' => 15,
        'cardCVV' => '083',
        'customerName' => 'Test Customer',
        'customerEmail' => 'test@testcustomer.com',
        'customerAddress' => '16 Test Street',
        'customerPostCode' => 'TE15 5ST',
        'orderRef' => 'Test purchase',

        // The following fields are mandatory for 3DS
```

```
        'remoteAddress' => $_SERVER['REMOTE_ADDR'],
        'threeDSRedirectURL' => $pageUrl . '&acs=1',

        // The following field allows options to be passed for 3DS
        // and the values here are for demonstration purposes only
        'threeDSOptions' => array(
            'paymentAccountAge' => '20190601',
            'paymentAccountAgeIndicator' => '05',
        ),
    );

    // Append the fields contained in browserInfo to the request as some are
    // mandatory for 3DS as detailed in section 5.5.5 of the Integration Guide.
    $req += $_POST['browserInfo'];

} else {
    // 3DS continuation request
    $req = array(
        'threeDSRef' => $_SESSION['threeDSRef'],
        'threeDSResponse' => $_POST['threeDSResponse'],
    );

}

// Create the signature using the function called below.
$req['signature'] = createSignature($req, $key);

// Initiate and set curl options to post to the gateway
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($req));
curl_setopt($ch, CURLOPT_HEADER, false);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

// Send the request and parse the response
if (($res = curl_exec($ch)) === false) {
    // You should exit gracefully
    die('Sorry, the request could not be sent: ' . curl_error($ch));
}

parse_str($res, $res);

// Close the connection to the gateway
curl_close($ch);

// Extract the return signature as this isn't hashed
$signature = null;
if (isset($res['signature'])) {
    $signature = $res['signature'];
    unset($res['signature']);
}

// Check the return signature
if (!$signature || $signature !== createSignature($res, $key)) {
    // You should exit gracefully
    die('Sorry, the signature check failed');
}

// Check the response code
if ((int)$res['responseCode'] === 65802) {
    // Send request to the ACS server displaying response in an IFRAME

    // Render an IFRAME to show the ACS challenge (hidden for fingerprint method)
    $style = (isset($res['threeDSRequest']['threeDSMethodData']) ? 'display: none;' : '');
    echo "<iframe name=\"threeds_acs\" style=\"height:420px; width:420px; {$style}\"></iframe>\n";

```

```
126.     // Silently POST the 3DS request to the ACS in the IFRAME
127.     echo silentPost($res['threeDSURL'], $res['threeDSRequest'], 'threeds_acs');
128.
129.     // Remember the threeDSRef as need it when the ACS responds
130.     $_SESSION['threeDSRef'] = $res['threeDSRef'];
131.
132. } else if ((int)$res['responseCode'] === 0) {
133.     echo "<p>Thank you for your payment.</p>";
134. } else {
135.     echo "<p>Failed to take payment: " . htmlentities($res['responseMessage']) . "</p>";
136. }
137.
138. // Function to create a message signature
139. function createSignature(array $data, $key) {
140.     // Sort by field name
141.     ksort($data);
142.
143.     // Create the URL encoded signature string
144.     $ret = http_build_query($data, '', '&');
145.
146.     // Normalise all line endings (CRNL|NLCR|NL|CR) to just NL (%0A)
147.     $ret = str_replace(array('%0D%0A', '%0A%0D', '%0D'), '%0A', $ret);
148.
149.     // Hash the signature string and the key together
150.     return hash('SHA512', $ret . $key);
151. }
152.
153. // Return HTML to render a hidden form used to collect some browser details
154. function collectBrowserInfo(array $options = null) {
155.
156.     $form_attrs = 'id="collectBrowserInfo" method="post" action="?"';
157.
158.     if (isset($options['formAttrs'])) {
159.         $form_attrs .= $options['formAttrs'];
160.     }
161.
162.     $device_data = array(
163.         'deviceChannel'             => 'browser',
164.         'deviceIdentity'            => (isset($_SERVER['HTTP_USER_AGENT']) ? htmlentities($_SERVER['HTTP_USER_
    AGENT']) : null),
165.         'deviceTimeZone'            => '0',
166.         'deviceCapabilities'        => '',
167.         'deviceScreenResolution'    => '1x1x1',
168.         'deviceAcceptContent'       => (isset($_SERVER['HTTP_ACCEPT']) ? htmlentities($_SERVER['HTTP_ACCEPT'])
     : null),
169.         'deviceAcceptEncoding'      => (isset($_SERVER['HTTP_ACCEPT_ENCODING']) ? htmlentities($_SERVER['HTTP_
    ACCEPT_ENCODING']) : null),
170.         'deviceAcceptLanguage'      => (isset($_SERVER['HTTP_ACCEPT_LANGUAGE']) ? htmlentities($_SERVER['HTTP_
    ACCEPT_LANGUAGE']) : null),
171.         'deviceAcceptCharset'       => (isset($_SERVER['HTTP_ACCEPT_CHARSET']) ? htmlentities($_SERVER['HTTP_A
    CCEPT_CHARSET']) : null),
172.     );
173.
174.     $form_fields = fieldToHtml('browserInfo', $device_data);
175.
176.     if (isset($options['formData'])) {
177.         foreach ((array)$options['formData'] as $name => $value) {
178.             $form_fields .= fieldToHtml($name, $value);
179.         }
180.     }
181.
182.     $ret = <<<EOS
183.         <form {$form_attrs}>
184.             {$form_fields}
185.         </form>
186.         <script>
187.             var screen_width = (window && window.screen ? window.screen.width : '0');
```

```
            var screen_height = (window && window.screen ? window.screen.height : '0');
            var screen_depth = (window && window.screen ? window.screen.colorDepth : '0');
            var identity = (window && window.navigator ? window.navigator.userAgent : '');
            var language = (window && window.navigator ? (window.navigator.language ? window.navigator.languag
    e : window.navigator.browserLanguage) : '');
            var timezone = (new Date()).getTimezoneOffset();
            var java = (window && window.navigator ? navigator.javaEnabled() : false);
            var fields = document.forms.collectBrowserInfo.elements;
            fields['browserInfo[deviceIdentity]'].value = identity;
            fields['browserInfo[deviceTimeZone]'].value = timezone;
            fields['browserInfo[deviceCapabilities]'].value = 'javascript' + (java ? ',java' : '');
            fields['browserInfo[deviceAcceptLanguage]'].value = language;
            fields['browserInfo[deviceScreenResolution]'].value = screen_width + 'x' + screen_height + 'x' + s
    creen_depth;
            window.setTimeout('document.forms.collectBrowserInfo.submit()', 0);
        </script>
EOS;

    return $ret;
}

// Render HTML to silently POST data to URL in target brower window
function silentPost($url = '?', array $post = null, $target = '_self') {

    $url = htmlentities($url);
    $target = htmlentities($target);
    $fields = '';

    if ($post) {
        foreach ($post as $name => $value) {
            $fields .= fieldToHtml($name, $value);
        }
    }

    $ret = "
        <form id=\"silentPost\" action=\"{$url}\" method=\"post\" target=\"{$target}\">
            {$fields}
            <noscript><input type=\"submit\" value=\"Continue\"></noscript>
        </form>
        <script>
            window.setTimeout('document.forms.silentPost.submit()', 0);
        </script>
    ";

    return $ret;
}

// Return a value as hidden HTML FORM fields
function fieldToHtml($name, $value) {
    $ret = '';
    if (is_array($value)) {
        foreach ($value as $n => $v) {
            $ret .= fieldToHtml($name . '[' . $n . ']', $v);
        }
    } else {
        // Convert all applicable characters or none printable characters to HTML entities
        $value = preg_replace_callback('/[\x00-\x1f]/', function($matches) { return '&#' . ord($matches[0]) .
    ';'; }, htmlentities($value, ENT_COMPAT, 'UTF-8', true));
        $ret = "<input type=\"hidden\" name=\"{$name}\" value=\"{$value}\" />\n";
    }

    return $ret;
}


?>
```

A-22.2.2 Sale Transaction (without 3-D Secure)

The following sample PHP code shows how to send a SALE transaction without support for 3-D Secure:

```php
<?PHP

// Signature key entered on MMS. The demo account is fixed to this value,
$key = 'Circle4Take40Idea';

// Gateway URL
$url = 'https://gateway.example.com/direct/';

// Request
$req = array(
    'merchantID' => '100001',
    'action' => 'SALE',
    'type' => 1,
    'countryCode' => 826,
    'currencyCode' => 826,
    'amount' => 1001,
    'cardNumber' => '4012001037141112',
    'cardExpiryMonth' => 12,
    'cardExpiryYear' => 15,
    'cardCVV' => '083',
    'customerName' => 'Test Customer',
    'customerEmail' => 'test@testcustomer.com',
    'customerPhone' => '+44 (0) 123 45 67 890',
    'customerAddress' => '16 Test Street',
    'customerPostCode' => 'TE15 5ST',
    'orderRef' => 'Test purchase',
    'transactionUnique' => uniqid(),
);

// Create the signature using the function called below.
$req['signature'] = createSignature($req, $key);

// Initiate and set curl options to post to the gateway
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($req));
curl_setopt($ch, CURLOPT_HEADER, false);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

// Send the request and parse the response
parse_str(curl_exec($ch), $res);

// Close the connection to the gateway
curl_close($ch);

// Extract the return signature as this isn't hashed
$signature = null;
if (isset($res['signature'])) {
    $signature = $res['signature'];
    unset($res['signature']);
}

// Check the return signature
if (!$signature || $signature !== createSignature($res, $key)) {
    // You should exit gracefully
    die('Sorry, the signature check failed');
}

// Check the response code
if ($res['responseCode'] === "0") {
    echo "<p>Thank you for your payment.</p>";
```

```
} else {
    echo "<p>Failed to take payment: " . htmlentities($res['responseMessage']) .  "</p>";
}

// Function to create a message signature
function createSignature(array $data, $key) {
    // Sort by field name
    ksort($data);

    // Create the URL encoded signature string
    $ret = http_build_query($data, '', '&');

    // Normalise all line endings (CRNL|NLCR|NL|CR) to just NL (%0A)
    $ret = str_replace(array('%0D%0A', '%0A%0D', '%0D'), '%0A', $ret);

    // Hash the signature string and the key together
    return hash('SHA512', $ret . $key);
}

?>
```

A-22.3 Batch Integration

A-22.3.1 Batch Submission

The following example PHP code shows how to send a batch request containing three SALE transactions:

```php
<?PHP

// Signature key entered on MMS. The demo account is fixed to this value,
$key = 'Circle4Take40Idea';

// Gateway URL
$url = 'https://gateway.example.com/batch/';

// Create a unique multipart boundary
$boundary = uniqid();

// Requests
$reqs = array(
    array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
        'amount' => 1001,
        'cardNumber' => '4012001037141112',
        'cardExpiryMonth' => 12,
        'cardExpiryYear' => 15,
        'cardCVV' => '083',
        'customerName' => 'Test Customer',
        'customerEmail' => 'test@testcustomer.com',
        'customerAddress' => '16 Test Street',
        'customerPostCode' => 'TE15 5ST',
        'orderRef' => 'Test purchase',
        'transactionUnique' => uniqid(),
        'threeDSRequired' => 'N',
        'avscv2CheckRequired' => 'N',
    ),
    array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
        'amount' => 2002,
        'cardNumber' => '4012001037141112',
        'cardExpiryMonth' => 12,
        'cardExpiryYear' => 15,
        'cardCVV' => '083',
        'customerName' => 'Test Customer',
        'customerEmail' => 'test@testcustomer.com',
        'customerAddress' => '16 Test Street',
        'customerPostCode' => 'TE15 5ST',
        'orderRef' => 'Test purchase',
        'transactionUnique' => uniqid(),
        'threeDSRequired' => 'N',
        'avscv2CheckRequired' => 'N',
    ),
    array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
```

```
        'amount' => 3003,
        'cardNumber' => '4012001037141112',
        'cardExpiryMonth' => 12,
        'cardExpiryYear' => 15,
        'cardCVV' => '083',
        'customerName' => 'Test Customer',
        'customerEmail' => 'test@testcustomer.com',
        'customerAddress' => '16 Test Street',
        'customerPostCode' => 'TE15 5ST',
        'orderRef' => 'Test purchase',
        'transactionUnique' => uniqid(),
        'threeDSRequired' => 'N',
        'avscv2CheckRequired' => 'N',
    ),
);

// Create the batch parts
$parts = array();
foreach ($reqs as $req) {

    // Create the signature using the function called below.
    $req['signature'] = createSignature($req, $key);

    $parts[] =
        "Content-Id: TX{$req['transactionUnique']}\r\n" .
        "Content-Type: application/x-www-form-urlencoded; charset=\"UTF-8\"\r\n" .
        "\r\n" .
        http_build_query($req);
}

// Join the parts together separated by the boundary string
$post = "\r\n--{$boundary}\r\n" . join("\r\n--{$boundary}\r\n", $parts) . "\r\n--{$boundary}--\r\n";

// Initiate and set curl options to post to the gateway
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
curl_setopt($ch, CURLOPT_HEADER, true);
curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Content-type: multipart/mixed; charset="UTF-8"; boundary=' . $boundary,
    'Content-length: ' . strlen($post),
));

// Send the request
$res = curl_exec($ch);

// Normally would process the response here, but for this example just echo it out
header ('Content-Type: text/plain');
echo $res . PHP_EOL;

// Close the connection to the gateway
curl_close($ch);

// Function to create a message signature
function createSignature(array $data, $key) {
    // Sort by field name
    ksort($data);

    // Create the URL encoded signature string
    $ret = http_build_query($data, '', '&');

    // Normalise all line endings (CRNL|NLCR|NL|CR) to just NL (%0A)
    $ret = str_replace(array('%0D%0A', '%0A%0D', '%0D'), '%0A', $ret);

    // Hash the signature string and the key together
```

```
127.     return hash('SHA512', $ret . $key);
128. }
129.
130. ?>
```

A-23 Example Library Code

The follow section provides samples of how to integrate with the Gateway using our integration libraries as documented in section A-20.1.

A-23.1 Gateway Integration Library

A-23.1.1 Hosted Sale Transaction

The following example PHP code shows how to send a SALE transaction using the Gateway library:

```php
<?PHP
require('gateway.php');

use \P3\SDK\Gateway;

// Signature key entered on MMS. The demo account is fixed to this value,
Gateway::$merchantSecret = 'Circle4Take40Idea';

// Gateway URL
Gateway::$hostedUrl = 'https://gateway.example.com/hosted/';

if (!isset($_POST['responseCode'])) {
    // Send request to gateway
    $req = array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
        'amount' => 1001,
        'orderRef' => 'Test purchase',
        'redirectURL' => ($_SERVER['HTTPS'] == 'on' ? 'https' : 'http') . '://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'],
    );

    try {
        echo Gateway::hostedRequest($req);
    } catch (\Exception $e) {
        // You should exit gracefully
        die('Sorry, the request could not be sent: ' . $e);
    }

} else {
    // Received response from gateway
    try {
        Gateway::verifyResponse($_POST);
    } catch(\Exception $e) {
        // You should exit gracefully
        die('Sorry, the request could not be sent: ' . $e);
    }

    // Check the response code
    if ($_POST['responseCode'] === 0) {
        echo "<p>Thank you for your payment.</p>";
    } else {
        echo "<p>Failed to take payment: " . htmlentities($_POST['responseMessage']) .  "</p>";
    }
}

?>
```

A-23.1.3 Direct Sale Transaction (with 3-D Secure)

The following example PHP code shows how to send a SALE transaction with support for 3-D Secure using the Gateway library:

```php
<?PHP

require('gateway.php');

use \P3\SDK\Gateway;

// Signature key entered on MMS. The demo account is fixed to this value,
Gateway::$merchantSecret = 'Circle4Take40Idea';

// Gateway URL
Gateway::$directUrl = 'https://gateway.example.com/direct/';

// Setup PHP session as use it to store data between 3DS steps
if (isset($_GET['sid'])) {
    session_id($_GET['sid']);
}

session_start();

// Compose current page URL (removing any sid and acs parameters)
$pageUrl = ((isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] == 'on') ? 'https://' : 'http://')
    . $_SERVER['SERVER_NAME'] . ($_SERVER['SERVER_PORT'] != '80' ? ':' . $_SERVER['SERVER_PORT'] : '')
    . preg_replace('/(sid=[^&]+&?)|(acs=1&?)/', '', $_SERVER['REQUEST_URI']);

// Add back the correct sid parameter (used as session cookie may not be passed when the page is redirected fro
m an IFRAME)
$pageUrl .= (strpos($pageUrl, '?') === false ? '?' : '&') . 'sid=' . urlencode(session_id());


// If ACS response into the IFRAME then redirect back to parent window
if (!empty($_GET['acs'])) {
    echo silentPost($pageUrl, array('threeDSResponse' => $_POST), '_parent');
    exit();
}

if (!isset($_POST['threeDSResponse'])) {
    // Initial request

    // Gather browser info - can be done at any time prior to the checkout
    if (!isset($_POST['browserInfo'])) {
        echo Gateway::collectBrowserInfo();
        exit();
    }

    // Direct Request
    $req = array(
        'merchantID' => 100001,
        'action' => 'SALE',
        'type' => 1,
        'currencyCode' => 826,
        'countryCode' => 826,
        'amount' => 1001,
        'cardNumber' => '4012001037141112',
        'cardExpiryMonth' => 12,
        'cardExpiryYear' => 15,
        'cardCVV' => '083',
        'customerName' => 'Test Customer',
        'customerEmail' => 'test@testcustomer.com',
        'customerAddress' => '16 Test Street',
        'customerPostCode' => 'TE15 5ST',
        'orderRef' => 'Test purchase',

```

```
62.
63.             // The following fields are mandatory for 3DS
64.             'remoteAddress' => $_SERVER['REMOTE_ADDR'],
65.             'threeDSRedirectURL' => $pageUrl . '&acs=1',
66.
67.             // The following field allows options to be passed for 3DS
68.             // and the values here are for demonstration purposes only
69.             'threeDSOptions' => array(
70.                 'paymentAccountAge' => '20190601',
71.                 'paymentAccountAgeIndicator' => '05',
72.             ),
73.         );
74.
75.     // Append the fields contained in browserInfo to the request as some are
76.     // mandatory for 3DS as detailed in section 5.5.5 of the Integration Guide.
77.     $req += $_POST['browserInfo'];
78.
79. } else {
80.     // 3DS continuation request
81.     $req = array(
82.         // The following field are only required for tbe benefit of the SDK
83.         'merchantID' => 100001,
84.         'action' => 'SALE',
85.
86.         // The following field must be passed to continue the 3DS request
87.         'threeDSRef' => $_SESSION['threeDSRef'],
88.         'threeDSResponse' => $_POST['threeDSResponse'],
89.     );
90.
91. }
92.
93.
94. try {
95.     $res = Gateway::directRequest($req);
96. } catch (\Exception $e) {
97.     // You should exit gracefully
98.     die('Sorry, the request could not be sent: ' . $e);
99. }
100.
101.// Check the response code
102.if ($res['responseCode'] === Gateway::RC_3DS_AUTHENTICATION_REQUIRED) {
103.    // Send request to the ACS server displaying response in an IFRAME
104.
105.    // Render an IFRAME to show the ACS challenge (hidden for fingerprint method)
106.    $style = (isset($res['threeDSRequest']['threeDSMethodData']) ? 'display: none;' : '');
107.    echo "<iframe name=\"threeds_acs\" style=\"height:420px; width:420px; {$style}\"></iframe>\n";
108.
109.    // Silently POST the 3DS request to the ACS in the IFRAME
110.    echo silentPost($res['threeDSURL'], $res['threeDSRequest'], 'threeds_acs');
111.
112.    // Remember the threeDSRef as need it when the ACS responds
113.    $_SESSION['threeDSRef'] = $res['threeDSRef'];
114.
115.} else if ($res['responseCode'] === Gateway::RC_SUCCESS) {
116.    echo "<p>Thank you for your payment.</p>";
117.} else {
118.    echo "<p>Failed to take payment: " . htmlentities($res['responseMessage']) . "</p>";
119.}
120.
121.
122.
123.// Render HTML to silently POST data to URL in target brower window
124.function silentPost($url = '?', array $post = null, $target = '_self') {
125.
126.    $url = htmlentities($url);
127.    $target = htmlentities($target);
128.    $fields = '';
```

```
129.
130.    if ($post) {
131.        foreach ($post as $name => $value) {
132.            $fields .= Gateway::fieldToHtml($name, $value);
133.        }
134.    }
135.
136.    $ret = "
137.        <form id=\"silentPost\" action=\"{$url}\" method=\"post\" target=\"{$target}\">
138.            {$fields}
139.            <noscript><input type=\"submit\" value=\"Continue\"></noscript
140.        </form>
141.        <script>
142.            window.setTimeout('document.forms.silentPost.submit()', 0);
143.        </script>
144.    ";
145.
146.    return $ret;
147.}
148.
149.  ?>
```

A-23.2 Hosted Payment Page Library

A-23.2.1 Hosted Sale Transaction

The following example code shows how to prepare a payment form to open the Hosted Payment Page in a lightbox style overlay on your website using the Hosted Payment Page library:

```
<html>
  <head>
    <!-- Load the Hosted Payment Page library -->
    <script src="https://gateway.example.com/sdk/web/v1/js/hostedforms.min.js"></script>
  </head>
  <body>
    <!--
      Hosted Payment <form> as created by the Gateway Integration Library hostedRequest() method
      with addition of 'data-hostedform-modal' attribute to signify a modal form is required.
    -->
    <form name="payment-form" method="post" action="https://gateway.example.com/hosted/" data-hostedform-modal>
      <input type="hidden" name="merchantID" value="100001" />
      <input type="hidden" name="action" value="SALE" />
      <input type="hidden" name="type" value="1" />
      <input type="hidden" name="currencyCode" value="826" />
      <input type="hidden" name="countryCode" value="826" />
      <input type="hidden" name="amount" value="1001" />
      <input type="hidden" name="orderRef" value="Test purchase" />
      <input type="hidden" name="redirectURL" value="https://www.merchant.com/payment/" />
      <input type="hidden" name="signature" value="07599ef4cdb2e26cb2bf34a9c65190a7ce82494bc1df144c3bb0d20ee2655d8278dc663b2b0421ef12b8f081e821151bb4c644277c5d65b5523a96539b53b5aa" />
      <input type="submit" value="Pay Now">
    </form>
    <script>
        // Create a new Hosted Form object which will cause the above <form> to load into a modal
        // overlay over this page.
        var form = new window.hostedForms.classes.Form(document.forms[0]);
    </script>
  </body>
</html>
```

A-23.2.2 Hosted Sale Transaction (jQuery)

The following example code shows how to prepare a payment form to open the Hosted Payment Page in a lightbox style overlay on your website using the Hosted Payment Page and jQuery libraries:

```
<html>
  <head>
    <!-- Load the jQuery library -->
    <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>

    <!-- Load the Hosted Payment Page library -->
    <script src="https://gateway.example.com/sdk/web/v1/js/hostedforms.min.js"></script>
  </head>
  <body>
      <!--
        Hosted Payment <form> as created by the Gateway Integration Library hostedRequest() method
        with addition of 'data-hostedforms-modal' attribute to signify a modal form is required.
      -->
      <form name="payment-form" method="post" action="https://gateway.example.com/hosted/" data-hostedforms-modal>
        <input type="hidden" name="merchantID" value="100001" />
        <input type="hidden" name="action" value="SALE" />
        <input type="hidden" name="type" value="1" />
        <input type="hidden" name="currencyCode" value="826" />
        <input type="hidden" name="countryCode" value="826" />
        <input type="hidden" name="amount" value="1001" />
        <input type="hidden" name="orderRef" value="Test purchase" />
        <input type="hidden" name="redirectURL" value="https://www.merchant.com/payment/" />
        <input type="hidden" name="signature" value="07599ef4cdb2e26cb2bf34a9c65190a7ce82494bc1df144c3bb0d20ee2655d8278dc663b2b0421ef12b8f081e821151bb4c644277c5d65b5523a96539b53b5aa" />
        <input type="submit" value="Pay Now">
      </form>
      <script>
          // Create a new Hosted Form object which will cause the above <form> to load into a modal
          // overlay over this page.
          var form = $(document.forms[0]).hostedForm();
      </script>
  </body>
</html>
```

A-23.2.3 Hosted Sale Transaction #2

The following example code shows how to create a payment form to open the Hosted Payment Page in a lightbox style overlay on your website using the Hosted Payment Page library:

```
<html>
  <head>
    <!-- Load the Hosted Payment Page library -->
    <script src="https://gateway.example.com/sdk/web/v1/js/hostedforms.min.js"></script>
  </head>
  <body>
    <!– Pay button placeholder -->
    <div id="paynow"></div>
    <script>
        // Create a new Hosted Form object which will render a payment button which will load
        // the Hosted Payment Pageo load into a modal overlay over this page.

        // The request can be provided from your server.
        var req = {
          merchantID: '100001',
          action: 'SALE',
          type: '1',
          currencyCode: '826',
          countryCode: '826',
          amount: '1001',
          orderRef: 'Test purchase',
          redirectURL: 'https://www.merchant.com/payment/',
          signature: '07599ef4cdb2e26cb2bf34a9c65190a7ce82494bc1df144c3bb0d20ee2655d8278dc663b2b0421ef12b8f081e821151bb4c644277c5d65b5523a96539b53b5aa',
        };

        var data = {
          id: 'my-payment-form',
          url: 'https://gateway.example.com/hosted/modal/',
          modal: true,
          data: req,
          submit: {
            type: 'button',
            label: 'Pay <i>Now</i>'
          }
        };

        var form = new window.hostedForms.classes.Form('paynow', data);
    </script>
  </body>
</html>
```

A-23.2.4 Hosted Sale Transaction #2 (jQuery)

The following example code shows how to create a payment form to open the Hosted Payment Page in a lightbox style overlay on your website using the Hosted Payment Page and jQuery libraries:

```
<html>
  <head>
    <!-- Load the jQuery library -->
    <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>

    <!-- Load the Hosted Payment Page library -->
    <script src="https://gateway.example.com/sdk/web/v1/js/hostedforms.min.js"></script>
  </head>
  <body>
    <!— Pay button placeholder -->
    <div id="paynow"></div>
    <script>
        // Create a new Hosted Form object which will render a payment button which will load
        // the Hosted Payment Pageo load into a modal overlay over this page.

        // The request can be provided from your server.
        var req = {
          merchantID: '100001',
          action: 'SALE',
          type: '1',
          currencyCode: '826',
          countryCode: '826',
          amount: '1001',
          orderRef: 'Test purchase',
          redirectURL: 'https://www.merchant.com/payment/',
          signature: '07599ef4cdb2e26cb2bf34a9c65190a7ce82494bc1df144c3bb0d20ee2655d8278dc663b2b0421ef12b8f081e821151bb4c644277c5d65b5523a96539b53b5aa',
        };

        var data = {
          id: 'my-payment-form',
          url: 'https://gateway.example.com/hosted/modal/',
          modal: true,
          data: req,
          submit: {
            type: 'button',
            label: 'Pay <i>Now</i>'
          }
        };

        var form = $('#paynow').hostedForm(data);
    </script>
  </body>
</html>
```

A-23.3 Hosted Payment Fields Library

The following example code shows how to create and manage Hosted Payment Fields using the Hosted Payment Field library.

The example shows how to style fields using an inline stylesheet and how to listen and react to the field's events.

The example also shows how to set up the payment form both automatically and manually and integrate with the jQuery validator plugin. You should choose the set-up method best suited for your needs and whatever validation plugin or functions you are familiar with.

Note: The example code demonstrates including the static transaction information, such as the ***merchantID*** *and* ***amount****, in hidden form fields and POSTing the form directly to the Gateway's Direct Integration using partial message signing. We would however recommend that you capture just the information you require and then POST this data to your own website where you can use it to build a new fully signed request to send to the Gateway's Direct Integration as a server-to-server request.*

```
<html>
  <head>
    <!-- Load the jQuery library -->
    <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-
CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"></script>

    <!-- Load the jQuery Validator plugin -->
    <script src="https://cdn.jsdelivr.net/npm/jquery-validation@1.19.1/dist/jquery.validate.min.js"></script>

    <!-- Load the Hosted Payment Field library -->
    <script src="https://gateway.example.com/sdk/web/v1/js/hostedfields.min.js"></script>

    <!-- General styles -->
    <style>
      body {
        font-size: 14px;
      }

      .form-group {
        margin: 4px 0 15px 0;
      }

      .form-group LABEL {
        display: inline-block;
        max-width: 100%;
        margin-bottom: 5px;
        font-weight: bold;
      }

      .form-control {
        display: block;
        box-sizing: border-box;
        height: 34px;
        width: 400px;
        padding: 6px 12px;
        font-size: 14px;
        color: #555;
        background-color: #fff;
        background-image: none;
        border: 1px solid #ccc;
        border-radius: 4px;
        -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, .075);
```

```
      box-shadow: inset 0 1px 1px rgba(0, 0, 0, .075);
      -webkit-transition: border-color ease-in-out .15s, -webkit-box-shadow ease-in-out .15s;
      -o-transition: border-color ease-in-out .15s, box-shadow ease-in-out .15s;
      transition: border-color ease-in-out .15s, box-shadow ease-in-out .15s;
    }

    .form-control.hf-focus {
      border-color: #66afe9;
      outline: 0;
      -webkit-box-shadow: inset 0 1px 1px rgba(0,0,0,.075), 0 0 8px rgba(102,175,233,.6);
      box-shadow: inset 0 1px 1px rgba(0,0,0,.075), 0 0 8px rgba(102,175,233,.6);
    }

    .has-error .form-control.hf-focus {
      border-color: #843534;
      -webkit-box-shadow: inset 0 1px 1px rgba(0,0,0,.075), 0 0 6px #ce8483;
      box-shadow: inset 0 1px 1px rgba(0,0,0,.075), 0 0 6px #ce8483;
    }
  </style>

  <!-- Hosted Field internal styles -->
  <style class="hostedfield">
    /* Grey out when disabled */
    .hostedfield:disabled {
      cursor: not-allowed;
      background-color: #eee;
      opacity: 1;
    }

    /* Change border and text to green when valid */
    .form-control:valid,
    .hostedfield:valid {
      border-color: #28a745 !important;
      color: #28a745 !important;
    }

    /* Change border and text to red when invalid */
    .form-control:invalid,
    .hostedfield:invalid {
      border-color: #a94442 !important;
      color: #a94442 !important;
    }

    /* Change text to light grey when readonly */
    .form-control:readonly,
    .hostedfield:readonly {
      color: lightgrey !important;
    }

    /* Emulate webkit auto fill style */
    .form-control.hf-autofill,
    .hostedfield.hf-autofill {
      background-color: rgb(250, 255, 189) !important;
      background-image: none !important;
      color: rgb(0, 0, 0) !important;
    }

    /* Add light blue placeholder */
    .form-control::placeholder,
    .hostedfield::placeholder {
      color: lightblue;
    }

    /* Show hovering over the control */
    .form-control:hover,
    .hostedfield:hover {
      font-style: italic;
```

```
109.      }
110.
111.      /* Style by id (hosted field will have '-hostedfield' appended to the id) */
112.      #form-card-number, #form-card-number-hostedfield {
113.        color: darkcyan;
114.      }
115.
116.    </style>
117.
118.    <!-- Hosted Field card-number internal styles -->
119.    <style class="card-number">
120.
121.      .hostedfield::placeholder {
122.        color: orange;
123.      }
124.
125.    </style>
126.  </head>
127.
128.  <body>
129.    <!-- tokenise payment data and send directly to the Gateway -->
130.    <form id="form" method="POST" novalidate="novalidate" lang="en"
131.      action="https://gateway.example.com/direct/"
132.      data-hostedform-tokenize='{"#form-customer-name": "customerName"}'>
133.      <input type="hidden" name="merchantID" value="100001">
134.      <input type="hidden" name="action" value="SALE">
135.      <input type="hidden" name="type" value="1">
136.      <input type="hidden" name="countryCode" value="826">
137.      <input type="hidden" name="currencyCode" value="826">
138.      <input type="hidden" name="amount" value="1001">
139.      <input type="hidden" name="orderRef" value="Test purchase">
140.      <input type="hidden" name="transactionUnique" value="1234">
141.      <input type="hidden" name="redirectURL" value="https://www.merchant.com/payment/">
142.      <input type="hidden" name="signature" value="5a0dd6fed71ef68bb3f20175b6a04bbd9d1c904d32ae3f160bd3b8f55740
    207e5d1e8de5e7e9960b136407e7454b82e428b8378003aa0146df3efa91a3e61b17|merchantID,action,type,countryCode,currenc
    yCode,amount,orderRef,transactionUnique,redirectURL">
143.      <input type="hidden" name="paymentToken" value="">
144.
145.      <div class="form-group">
146.        <label for="form-customer-name">Name on card:</label>
147.        <input id="form-customer-name" type="text" name="paymentToken[customerName]" autocomplete="cc-
    name" class="form-control form-control-native hostedfield-tokenise" placeholder="Firstname Surname" required>
148.      </div>
149.
150.      <div class="form-group">
151.        <label for="form-card-number">Card Number:</label>
152.        <input id="form-card-number" type="hostedfield:cardNumber" name="card-number" autocomplete="cc-
    number" class="form-control form-control-
    hosted" style="background: #f2f8fb;" placeholder="**** **** **** ****" required>
153.      </div>
154.
155.      <div class="form-group">
156.        <label for="form-card-expiry-date">Card Expiry Date:</label>
157.        <input id="form-card-expiry-date" type="hostedfield:cardExpiryDate" name="card-expiry-
    date" autocomplete="cc-exp" class="form-control form-control-hosted" required>
158.      </div>
159.
160.      <div class="form-group">
161.        <label for="form-card-start-date">Card Issue Date:</label>
162.        <input id="form-card-start-date" type="hostedfield:cardStartDate" name="card-start-
    date" autocomplete="cc-iss" class="form-control form-control-hosted" data-hostedfield='{"dropdown":true}' data-
    hostedfield-format="N - m | y" data-hostedfield-min-date="-40" data-hostedfeld-max-date="0">
163.      </div>
164.
165.      <div class="form-group">
166.        <label for="form-card-cvv">CVV:</label>
```

```
        <input id="form-card-cvv" type="hostedfield:cardCVV" name="card-cvv" autocomplete="cc-csc" class="form-control form-control-hosted" required>
      </div>

      <button id="form-submit" type="submit">Pay <span>▷</span></button>
    </form>

    <script>
      // This example demonstrates both automatic and manual form setup
      var automatic_setup = true;

      $(document).ready(function () {

        var $form = $('#form');

        // Listen for events on the form to see those sent from the Hosted Payment Fields
        // (For demonstration purposes only)
        $form.on(events);

        if (automatic_setup) {
          ////////////////////////////////////////////////////////////////
          // FORM AUTOMATIC SETUP
          ////////////////////////////////////////////////////////////////

          var opts = {
            // Auto setup the form creating all hosted fields (default)
            autoSetup: true,

            // Auto validate, tokenise and submit the form (default)
            autoSubmit: true,

            // Optional field configuration (by type)
            fields: {
              any: {
                nativeEvents: true,
              },
              cardNumber: {
                selector: $('#form-card-number'),
                style: 'text-decoration: green wavy underline;',
                stylesheet: $('style.hostedfields, style.card-number')
              }
            }
          };

          try {
            // Create form, automatically creating all child Hosted Payment Fields
            $form.hostedForm(opts);
          } catch(e) {
            showError('Failed to create hosted form: ' + e);
            throw e; // Can't continue with this script
          }

          // Listen for some events from the form thrown by the auto methods
          $form.on({
            // Let jQuery Validator check the form on submission
            'hostedform:presubmit': function (event) {
              console.log('Form submitting');
              return $form.valid();
            },

            // Show form is valid
            'hostedform:valid': function (event) {
              console.log('Form valid');
              return true;
            },

            // Show any validation errors
```

```
            'hostedform:invalid': function (event, details) {
              console.log('Form invalid');
              showFieldErrors(details.invalid);
              return true;
            },

            // Show general error
            'hostedform:error': function (event, details) {
              showError(details.message);
              return true;
            }
          });

          // Use jQuery validator to validate the form
          $form.validate();

          // End of form automatic setup

        } else {
          ////////////////////////////////////////////////////////////////
          // FORM MANUAL SETUP
          ////////////////////////////////////////////////////////////////

          try {
            // Create the card number field with custom options
            $('#form-card-number').hostedField({
              nativeEvents: true,
              style: 'text-decoration: green wavy underline;',
              stylesheet: $('style.hostedfields, style.card-number')
            });

            // Create the remaining hosted fields
            $('.form-control-hosted:input', $form).hostedField({nativeEvents: true});

          } catch (e) {
            showError('Failed to create hosted fields: ' + e);
            throw e; // Can't continue with this script
          }

          $form.validate({
            // Get the hosted form widget for the submitted form (Form1 only)
            submitHandler: function () {
              try {
                console.log('getPaymentToken');

                // Check we have some enabled fields to submit
                if ($($form[0].elements).filter(':enabled:not([type="hidden"])').length === 0) {
                  showError('You must enable some fields');
                  return false;
                }

                var hostedform = $form.hostedForm('instance');

                var also = {
                  customerName: $('#form-customer-name').val()
                };

                hostedform.getPaymentDetails(also, true).then(

                  // Success validating the form and requesting a payment token
                  function (details) {
                    if (details.success) {
                      $form[0].elements['paymentToken'].value = details.paymentToken;
                      $form[0].submit();
                    } else if (details.invalid) {
                      $form.valid();
                      showFieldErrors(details.invalid);
```

```
                  } else {
                    showError('There was a problem fetching the payment token. Please seek assistance.');
                  }
                },

                // Failure either validating the form or requesting the payment details
                function (e) {
                  showError('There was a problem fetching the payment token. Please seek assistance.');
                }
              );

            } catch (e) {
              showError('There was a problem fetching the payment token. Please seek assistance.');
            }
          }
        });

        // End of form manual setup

      }

      // Hide errors once all fields are valid
      $('#form :input').on('valid', function () {
        if ($(this.form).find(':invalid').length === 0) {
          hideError($(this.form));
        }
      })

      // Listen for some events on the none Hosted Fields
      $('.form-control-native').on('invalid', bsMarkInvalid);
      $('.form-control-native').on('valid', bsMarkValid);

      // Check we can see the Hosted Fields via their new class
      // (For demonstration purposes only)
      console.log($('.form-control-hosted.hostedfield-element'));

      // Check we can see the Hosted Fields via the psuedo element
      // (For demonstration purposes only)
      console.log($('.form-control:hostedfield'));

    });

    ////////////////////////////////////////////////////////////////////
    // Supporting functions
    ////////////////////////////////////////////////////////////////////

    // Display events that are passed from hosted field
    var events = {
      'hostedfield:create.example'                          : showEvent,
      'hostedfield:destroy.example'                         : showEvent,
      'hostedfield:ready.example'                           : showEvent,
      'hostedfield:style.example'                           : showEvent,
      'hostedfield:placeholder.example'                     : showEvent,
      'hostedfield:invalid.example invalid.example'         : showEvent,
      'hostedfield:userinvalid.example userinvalid.example' : showEvent,
      'hostedfield:valid.example valid.example'             : showEvent,
      'hostedfield:uservalid.example uservalid.example'     : showEvent,
      'hostedfield:disabled.example disabled.example'       : showEvent,
      'hostedfield:enabled.example enabled.example'         : showEvent,
      'hostedfield:required.example required.example'       : showEvent,
      'hostedfield:optional.example optional.example'       : showEvent,
      'hostedfield:readonly.example readonly.example'       : showEvent,
      'hostedfield:readwrite.example readwrite.example'     : showEvent,
      'hostedfield:focus.example focus.example'             : showEvent,
      'hostedfield:blur.example blur.example'               : showEvent,
      'hostedfield:mouseenter.example mouseenter.example'   : showEvent,
      'hostedfield:mouseleave.example mouseleave.example'   : showEvent,
```

```
      'hostedfield:mouseover.example mouseover.example'      : showEvent,
      'hostedfield:mouseout.example mouseout.example'        : showEvent,
      'hostedfield:mousemove.example mousemove.example'      : showEvent,
      'hostedfield:keydown.example keydown.example'          : showEvent,
      'hostedfield:keypress.example keypress.example'        : showEvent,
      'hostedfield:keyup.example keyup.example'              : showEvent,
      'hostedfield:change.example change.example'            : showEvent,
      'hostedfield:input.example input.example'              : showEvent,

      'hostedfield:invalid.example invalid.example'          : bsMarkInvalid,
      'hostedfield:valid.example valid.example'              : bsMarkValid,
      'hostedfield:valid.example valid.example'              : hideError,
    };

    function isInvalid(element) {
      return !element[0].checkValidity();
    }

    function showError(msg) {
      $('#error-info').html(msg).show();
    }

    function hideError($form, msg) {
      $('#error-info', $form).hide();
    }

    function showFieldErrors(errors) {
      var msg = '<h5>Error</h5><p>The following fields are invalid:</p><ul>';
      for (var p in errors) {
        msg += '<li><b>' + p + ':</b> ' + errors[p] + '</li>';
      }
      msg += '</ul>'
      showError(msg);
    }

    function bsMarkInvalid(e) {
      var element = (e instanceof $.Event ? this : e);
      $(element).closest('.form-group').addClass('has-error');
    }

    function bsMarkValid(e) {
      var element = (e instanceof $.Event ? this : e);
      $(element).closest('.form-group').removeClass('has-error');
    }

    function showEvent(event) {
      console.log(event);
      console.log('Field ' + event.type + ' event: ', this, arguments);
    }

    jQuery.validator.setDefaults({
      ignore: [],
      rules: {
        'customer-name': {
          checkValidity: true,
          required: false
        },
        'card-details': {
          checkValidity: true,
          required: false
        },
        'card-number': {
          checkValidity: true,
          required: false
        },
        'card-expiry-date': {
          checkValidity: true,
```

```
            required: false
          },
          'card-start-date': {
            checkValidity: true,
            required: false
          },
          'card-issue-number': {
            checkValidity: true,
            required: false
          },
          'card-cvv': {
            checkValidity: true,
            required: false
          }
        },
        keyup: null, // Don\'t validate on keyup
        showErrors: function (errorMap, errorList) {
          if (errorList && errorList.length) {
            var errors = {};
            for (var i = 0, max_i = errorList.length; i < max_i; i++ ) {
              var label = $('label[for="' + errorList[i].element.id + '"]:not(".error")').text();
              errors[label] = errorList[i].message;
            }
            showFieldErrors(errors);
          }
          this.defaultShowErrors(errorMap, errorList);
        },
        highlight: bsMarkInvalid,
        unhighlight: bsMarkValid,
        errorPlacement: function (error, element) {
          $(element).closest('.form-control:not(".hostedfield-element")').after(error);
        }
      });

      $.validator.addMethod('checkValidity',
        function (value, element, params, message) {
          element.checkValidity();
          var valid = (element.validationMessage === '');
          $(element).attr('aria-invalid', !valid);
          return valid;
        },
        function (params, element) {
          return element.validationMessage;
        }
      );

    </script>

  </body>
</html>
```

A-24 Frequently Asked Questions

1. I'm getting Invalid Credentials. What do I do?

- Check your Merchant Account ID in your integration is correct. Our Gateway Merchant Account IDs typically begin with 1 and are currently 6 digits long, eg 100001.

2. I'm getting an invalid signature error message. How do I fix it?

- Check that you are using the correct method for calculating the signature and the correct secret signature key for the Merchant Account used.

- Make sure that you are not using an image form submit button because that will add fields to the post that cannot be removed and will render the signature useless.

Refer to appendix A-12 for a step-by-step guide to creating a signature. If you use the same values as in the example, you can check if your signature generation routine produces the same results.

This test step by step generator is available from the Gateway at the following address[1];

https://gateway.example.com/devtools/sigtest.php

3. I have more than one Merchant Account - how do I use more than one?

- You have a couple of options here. You can set up separate integrations for each Merchant Account, which can be a bit inconvenient. Your other option is to request they are connected together. Please contact our support team to get your Merchant Accounts connected and you will then only need to use one.

4. I receive a 'Bad Testcard Usage' error message. Why?

- If you receive this error message, you are using test cards on a live Merchant Account. Please only use live cards on live Merchant Accounts. Our test cards will only work on the test Merchant Account provided when you sign up with us.

[1] Please use the correct hostname as explained in section 1.6.

INDEX