Hosted payment page integration guide

Version 2.3 | July 2023

Tyl by Natwest offers you a quick and easy way to add payment capabilities to your website using our hosted payment page.

We accept the customer's card data and handle the redirection to their bank for 3D Secure authentication − so you don't have to. As the page is fully hosted by us, the burden of compliance with the Data Security Standard of the Payment Card Industry (PCI DSS) is significantly reduced.

This guide shows you how to easily integrate your website with our solution.

Hosted payment page integration guide

Version 2.2

Contents

Getting support

If you'd like to know about settings, customisation, performing refunds and viewing transactions, take a look at our Virtual terminal user guide.

For all Virtual Terminal, Admin and Transactions queries please contact our support team. For more information on how to contact the team please visit:

 https://www.tylbynatwest.com/help-and-support

1.1 Integration Support

Our dedicated gateway integration support team can support you and/or you developer with any matters relating to integration, testing, investigation and troubleshooting of test and live gateway issues.

If you require integration support or are experiencing any problems with your integration or our gateway, please contact us on:

 +44 (0) 1268 666 033

 online@support.natwest-tyl.com

If you've read the documentation and can't find the answer to your question, just give us a call and we'll be happy to help.

2. Integration checklist

Here's a list of things to consider as you integrate our Hosted Payment Page either directly on your website or using a shopping cart.

1. We'd recommend first testing your integration within a testing environment. If you do not have a test environment yet please request this by contacting our online support teams.

2. Once you have been provided with your **Test Store ID** and **Test Shared Secret** along with the Test URL https://test.ipg-online.com/connect/gateway/processing
   This is usually sent to you VIA email.

3. Submit a sale using a test card. (See Appendix 7.)

If you are making use of a third-party plugin, then please follow their instructions.

4. Set-up the URLs on your website that you want to redirect the customer to at the end of the payment journey on the hosted payment page.

   You can configure the Success and Failure URL's within the Virtual Terminal portal or within the payment request. If you're using a shopping cart you should follow the instructions to set-up the Success and Failure URLs.

5. Configure your notification URL in the Virtual Terminal portal or provide with the payment request so that your server or other systems can receive the outcome of the payment. If you are using a shopping cart you should follow the instructions set-up the notification URL.

6. Familiarise yourself with performing a 'Return' (Refund) against a successful payment via the Virtual Terminal portal so that you are prepared should you need to refund a transaction in future.

7. Familiarise yourself with the transaction details found in the Virtual Terminal portal so you understand the outcome of 3D Secure and whether liability lies with you or the issuer in the case of fraudulent transactions and help you answer customer queries. You might also want to understand the outcome of the Address Verification Service (AVS) and Security Code checks.

Depending on the nature of your business you may also want to:

8. Submit a Pre-Authorisation transaction and subsequently complete it via the Virtual Terminal portal.

9. Use the Virtual Terminal portal to:

   a. Void an unsettled payment.

   b. Switch on Merchant email receipts, Customer emails receipts or both.

   c. Set Risk Management rules.

   The **Virtual Terminal User Guide** describes these features in more detail, alternatively you can also visit our Virtual Terminal Help & Support page for more information

Once you are happy with your integration testing, you will require your production details (**Live Store ID**, **Live Shared Secret** and **Live URL**) if you having trouble locating these details please reach out to our online support teams.

10. Point your website or shopping cart integration to the **Live URL** with the **live Store ID** and **live Shared Secret** we sent to you by email.

11. Update any test redirect and notification URLs to their live equivalent, either in the Virtual Terminal Portal, in the payment request itself or in your shopping cart set-up.

3. Your integration options

We offer a few ways to integrate payments into your web site, based on your needs.

3.1 Overview

Checkout Option	Description
Combined	• Redirect solution to our customisable Hosted Payments Page where the cardholder can securely submit their payment details and go through 3D-Secure authentication. • The combined option supports E-Wallets such as ApplePay and GooglePay however, these options are not supported when using an iFrame.
Direct Post	• Create your own payment form • You display and host the form in your website/application • Just like the "Combined" option we handle 3D-Secure authentication **Please note**: Whilst the card details are being captured on your form, there is still a redirect to the gateway which occurs however, your cardholder will only see a spinning wheel (spindle) for a brief moment before being returned back your page.

3.2 Combined Checkout Option

This option consolidates the payment process into a single 'combined' hosted payment page which is automatically optimised for different kinds of user devices, e.g. PC, smartphone or tablet.

It also shows your business name at the top and allows you to display a summary of the purchased items to your customer.

The hosted page can be customised with your own logo, colours, and font types so they match the look and feel of your website. Have a look at the **Virtual Terminal User Guide** to find out how to do this.

3.3 Direct post

In the scenarios where you prefer not to use a hosted payment page, you can submit the required customer data directly from your own form, but please be aware that if you store or process sensitive cardholder data within your own application/website, you must ensure that your system components are compliant with the Data Security Standard of the Payment Card Industry (PCI DSS).

You create the payment form and display it within your website or application. When your customer has entered the card details and presses the "continue button", the customer's device sends the payment information directly to the gateway.

If you choose the Direct Post option and create your own forms, there are additional fields that must be included in your transaction request to the gateway, which are listed in Appendix 6 – Using your own form to capture Data

Please note: Whilst the card details are being captured on your form, there is still a redirect to the gateway which occurs however, your cardholder will only see a spinning wheel (spindle) for a brief moment before being returned back your page.

4. Getting started

This section provides a simple example on how to integrate your website using the Combined Checkout Option ("combinedpage").

4.1 Authentication

In order to authenticate with us, you must have the following items:

- **Store Name**
  This is the Store ID given to you by us. (For example: 72123456789)
- **Shared Secret**
  This is the shared secret we have sent to you by email. This is used when constructing the hash value.

4.2 ASP Example

The following ASP example demonstrates a simple page hosting the HTML form with the required parameters necessary to redirect the cardholder to the Hosted payment page.

When the cardholder clicks 'Submit', they're redirected to our secure page to enter the card details. After payment's been completed, the user will be redirected to your receipt page.

The response URL's can be configured statically in the Virtual Terminal or alternatively dynamically within each transaction request:

```
<!-- #include file="ipg-util.asp"-->


 <html>
<head><title>IPG Connect Sample for ASP</title></head>
<body>
  <p><h1>Order Form</h1></p>
  <form method="post" action="https://test.ipg-online.com/connect/gateway/processing">
    <input type="hidden" name="txntype" value="sale"/>
    <input type="hidden" name="timezone" value="Europe/Berlin"/>
    <input type="hidden" name="txndatetime" value="<% getDateTime() %>"/>
    <input type="hidden" name="hash_algorithm" value="HMACSHA256"/>
    <input type="hidden" name="hashExtended" value="<% call createExtendedHash("13.00","978") %>"/>
    <input type="hidden" name="storename" value="10123456789"/>
    <input type="hidden" name="checkoutoption" value="combinedpage"/>
    <input type="hidden" name="responseFailURL" value="https://localhost:8643/webshop/response_failure.jsp"/>
    <input type="hidden" name="responseSuccessURL" value=" https://localhost:8643/webshop/response_success.jsp"/>
    <input type="hidden" name="transactionNotificationURL"value="https://www.example.merchant/webshop/transactionNotification"/>
    <input type="text" name="chargetotal" value="13.00"/>
    <input type="hidden" name="currency" value="978"/>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

The code shown in Appendix 2 represents the included file ipg-util.asp. It includes code for generating the SHA-256 hash required by us. The provision of a hash in the example ensures that you're the only merchant that can perform transactions for this store.

Please note, that the POST URL used is for integration testing only. When you're ready to go live, please use the live URL we sent you or contact us and we'll share it with you.

You should also note that the included file, ipg-util.asp, uses a server-side JavaScript file to build the SHA-256 hash. To prevent fraud, it's recommended that the 'hash' is calculated within your server and that JavaScript isn't used (as we've indicated).

4.3 PHP Example

This PHP example demonstrates a simple page hosting the HTML form with the required parameters necessary to redirect the cardholder to the Hosted payment page.

When the cardholder clicks 'Submit', they're redirected to our secure page to enter their card details. After payment's been completed, the user will be redirected to your receipt page.

The response URL's can be configured statically in the Virtual Terminal or alternatively dynamically within each transaction request:

```
<? include("ipg-util.php"); ?>


<html>
<head><title>IPG Connect Sample for PHP</title></head>
<body>
  <p><h1>Order Form</h1></p>
  <form method="post" action="https://test.ipg-online.com/connect/gateway/processing">
    <input type="hidden" name="txntype" value="sale"/>
    <input type="hidden" name="timezone" value="Europe/Berlin"/>
    <input type="hidden" name="txndatetime" value="<% getDateTime() %>"/>
    <input type="hidden" name="hash_algorithm" value="HMACSHA256"/>
    <input type="hidden" name="hashExtended" value="<% call createExtendedHash("13.00","978") %>"/>
    <input type="hidden" name="storename" value="10123456789"/>
    <input type="hidden" name="checkoutoption" value="combinedpage"/>
    <input type="hidden" name="responseFailURL" value="https://localhost:8643/webshop/response_failure.jsp"/>
    <input type="hidden" name="responseSuccessURL" value="https://localhost:8643/webshop/response_success.jsp"/>
    <input type="hidden" name="transactionNotificationURL" value="https://www.example.merchant/webshop/transactionNotification"/>
    <input type="text" name="chargetotal" value="13.00"/>
    <input type="hidden" name="currency" value="978"/>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

The POST URL used in this example is for integration testing only. When you're ready to go live, please use the live URL we've sent you or contact us and we'll share it with you.

The code shown in Appendix 3 represents the included file ipg-util.php. It includes code for generating the SHA-256 hash required by us. The provision of a hash in the example ensures that you're the only merchant that can perform transactions for this store.

5. Mandatory fields

Depending on the transaction type, the following form fields must be included in the form submitted to us. Please refer to the Appendices for further implementation details.

Field Name	Description, possible values and format
txntype	'sale' or 'preauth'
timezone	Time zone of the transaction in Area/Location format, e.g. Africa/Johannesburg America/New_York America/Sao_Paulo Asia/Calcutta Australia/Sydney Europe/Amsterdam Europe/Berlin Europe/Dublin Europe/London Europe/Rome
txndatetime	YYYY:MM:DD-hh:mm:ss (exact current time of the chosen time zone)
hash_algorithm	This is to indicate the algorithm that you're using for the hash calculation. The possible values are: HMACSHA256 HMACSHA384 HMACSHA512
hashExtended	The extended hash needs to be calculated using all non-empty gateway specified request parameters in ascending order of the parameter names, where the upper-case characters come before the lower case (based on ASCII value) and the shared secret must be used as the secret key for calculating the hash value. When you are using Direct Post, there is also an option where you do not need to know the card details (PAN, CVV and Expiry Date) for the hash calculation. This will be managed with a specific setting performed on your store. Please contact your local support team if you want to enable this feature. An example of how to generate a hash is given in Appendix 1 – How to generate an Extended Hash
responseFailURL*	The URL where you wish to direct customers after a declined or unsuccessful transaction (your Sorry URL) *This parameter is only needed if it has not been setup within Virtual Terminal / Customisation.
responseSuccessURL*	The URL where you wish to direct customers after a successful transaction (your Thank You URL) – *This parameter is only needed if it has not been setup within Virtual Terminal / Customisation.

`storename`	This is the ID of the store – this is provided to you by Tyl.
`chargetotal`	This is the total amount of the transaction using a dot or comma as decimal separator, e. g. `12.34` for an amount of 12 GBP and 34 pence. Group separators like 1,000.01 / 1.000,01, are not allowed.
`currency`	The numeric ISO code for the transaction currency. The ISO code for GBP is `826`.

6. Optional form fields

Field Name	Description, possible values and format
`cardFunction`	This field allows you to indicate the card function in case of combo cards (which provide credit and debit functionality on the same card). It can be set to '`credit`' or '`debit`'. The field can also be used to validate the card type in such a way that transactions where the submitted card function doesn't match the card's capabilities will be declined (e.g. if you submit "`cardFunction=debit`" and the card is a credit card, the transaction will be declined.
`checkoutoption`	This field must be set as: '`combinedpage`' For a payment process where the payment method choice and the typical next step (e.g. entry of card details or selection of bank) in consolidated in a single page.
`comments`	Place any comments about the transaction here. **Max Length**: *1024* **Format**: *Standard ASCII character set allowed*
`customerid`	This field allows you to transmit any value, e. g. your ID for the customer. **Max length**: 32 characters **Format**: *Standard ASCII character set allowed*
`invoicenumber`	This field allows you to transmit any value, e. g. an invoice number or class of goods. **Max length**: 48 characters **Format**: *Standard ASCII character set allowed*

`item1 up to item999`	Line items are regular integration key-value parameters (URL-encoded), where:
	• the name is a combination of the keyword item and a number, where the number indicates the list position, e.g. item1
	• the value is represented by a semicolon-separated list of values, where the position indicates the meaning of the list item property e.g. <1>;<2>;<3>;<4>;<5>;<6>;<7>
	
	The 'item1' to 'item999' parameters allow you to send basket information in the following format:
	
	`id;description;quantity;item_total_price;sub_total;vat_tax;shipping`
	
	'shipping' always has to be set to '0' for a single line item. If you want to include a shipping fee for an order, please use the predefined id `IPG_SHIPPING`.
	
	For other fees you may want to add to the total order, you can use the predefined id `IPG_HANDLING`.
	
	When you want to apply a discount, you should include an item with a negative amount and change the total amount of the order accordingly.
	
	Don't forget to consider the 'quantity' when calculating the values, e.g. subtotal and VAT since they're fixed by items.
	
	Examples:
	
	`A;Product A;1;5;3;2;0`
	`B;Product B;5;10;7;3;0`
	`C;Product C;2;12;10;2;0`
	`D;Product D;1;-1.0;-0.9;-0.1;0`
	`IPG_SHIPPING;Shipping costs;1;6;5;1;0`
	`IPG_HANDLING;Transaction fee;1;6.0;6.0;0;0`

<table>
<tr><td>language</td><td>This parameter can be used to override the default payment page language configured for your website

Additional languages are available, for more information on which values to configure for other languages not otherwise listed below, please reach out to your local support team.

<table><tr><th>Language</th><th>Value</th></tr><tr><td>Chinese (simplified)</td><td>zh_CN</td></tr><tr><td>Chinese (traditional)</td><td>zh_TW</td></tr><tr><td>Czech</td><td>cs_CZ</td></tr><tr><td>Dutch</td><td>nl_NL</td></tr><tr><td>English (USA)</td><td>en_US</td></tr><tr><td>English (UK)</td><td>en_GB</td></tr><tr><td>Finnish</td><td>fi_FI</td></tr><tr><td>French</td><td>fr_FR</td></tr><tr><td>German</td><td>de_DE</td></tr><tr><td>Greek</td><td>el_GR</td></tr><tr><td>Italian</td><td>it_IT</td></tr><tr><td>Polish</td><td>pl_PL</td></tr><tr><td>Portuguese (Brazil)</td><td>pt_BR</td></tr><tr><td>Serbian</td><td>sr_RS</td></tr><tr><td>Slovak</td><td>sk_SK</td></tr><tr><td>Spanish</td><td>es_ES</td></tr></table></td></tr>
<tr><td>merchantTransactionId</td><td>Allows you to assign a unique ID to the transaction.
This ID can be used to reference to this transaction in a PostAuth or Void request (‘referencedMerchantTransactionId’).</td></tr>
<tr><td>oid</td><td>This field allows you to assign a unique ID to your order. If you choose not to assign an order ID, our system will automatically generate one for you.
Max Length: 100 characters
Format: Standard ASCII character set allowed</td></tr>
<tr><td>paymentMethod</td><td>If you let the customer select the payment method (e.g. Mastercard, Visa) on your website prior to reaching the hosted payment page or want to define the payment type yourself, send the parameter ‘paymentMethod’ along with your Sale or PreAuth transaction. If you don’t submit this parameter, we’ll display a drop-down menu for the customer to choose from the payment methods available for your website.

For valid payment method values, please refer to Appendix 4 – Payment Method List</td></tr>
<tr><td>ponumber</td><td>This field allows you to submit a Purchase Order Number:
Max Length: 50 characters.
Format: Standard ASCII character set allowed</td></tr>
<tr><td>refer</td><td>This field describes who referred the customer to your store.</td></tr>
<tr><td>shipping</td><td>This parameter can be used to submit the shipping fee, in the same format as ‘chargetotal’. If you submit ‘shipping’, the parameters ‘subtotal’ and ‘vattax’</td></tr>
</table>

	have to be submitted as well. Note, that the 'chargetotal' must be equal to 'subtotal' plus 'shipping' plus 'vattax'.
vattax	This field allows you to submit an amount for Value Added Tax or other taxes. Please ensure the sub-total amount, plus shipping, plus tax, equals the charge total.

7. Sending billing and shipping addresses

If you're using the hosted payment page to only capture the payment card data, you may also want to send your customer's billing address and shipping address with the payment. The following sections describe the format of the fields you need to send:

7.1 Billing address

The following table outlines the format of these additional fields:

Field Name	Possible Values	Description
bcompany	Alphanumeric characters, spaces, and dashes	Company name
bname	Alphanumeric characters, spaces, and dashes	Recipient's Name
baddr1	Max Length: 96 characters, including spaces	Billing address 1
baddr2	Max Length: 96 characters, including spaces	Billing address 2
bcity	Max Length: 96 characters, including spaces	Billing city
bstate	Max Length: 96 characters, including spaces	Billing State, province or territory
bcountry	2 Letter Country Code	Billing Country
bzip	Max Length: 24 characters, including spaces	Billing Zip or postal code
phone	Max Length: 32 Characters	Customers phone number
fax	Max Length: 32 Characters	Customer's fax number
email	Max Length: 254 Characters	Customer's email address

7.2 Shipping address

The following table outlines the format of the shipping fields:

Field Name	Possible Values	Description
sname	Alphanumeric characters, spaces, and dashes	Ship-to name
saddr1	Max Length: 96 characters, including spaces	Shipping address Line 1
saddr2	Max Length: 96 characters, including spaces	Shipping address Line 2
scity	Max Length: 96 characters, including spaces	Shipping city
sstate	Max Length: 96 characters, including spaces	Shipping State, province or territory
scountry	2 Letter Country Code	Shipping Country
szip	Max Length: 24 characters, including spaces	Billing Zip or postal code

8. Additional custom fields

If you want to gather a bit more customer data to help your business, you can send as many custom fields to us as you want – and we'll return them – along with all other fields, to the response URL.

You can submit up to 10 custom fields to us to appear in the Virtual Terminal's Order Detail View.

Field Name	Description, possible values and format
customParam_key	If you want to use this feature, please send the custom fields in the format customParam_key=value. Max Length 100 characters. Example: <input type="hidden" name="customParam_color" value="green"/>

9. 3D Secure

All electronic payments in scope of SCA will be subjected to 3D secure Authentication for the schemes we offer.

3D Secure helps protect your business in the event a payment is made fraudulently by passing the liability for specific types of chargebacks to the cardholder's issuing bank when the cardholder successfully authenticates with their bank during the payment. The hosted payment page automatically redirects the cardholder to their bank's authentication page so you don't need to build this into your website.

Card transactions with 3D Secure hold in a 'pending' status while cardholders enter their password or activate their card for 3D Secure during checkout. During this time, when the final transaction result isn't yet determined, we set the Approval Code to „?:waiting 3dsecure". If the session expires before the cardholder returns from the 3D Secure dialogue with his bank, you'll see "N:-5103:Cardholder did not return from ACS".

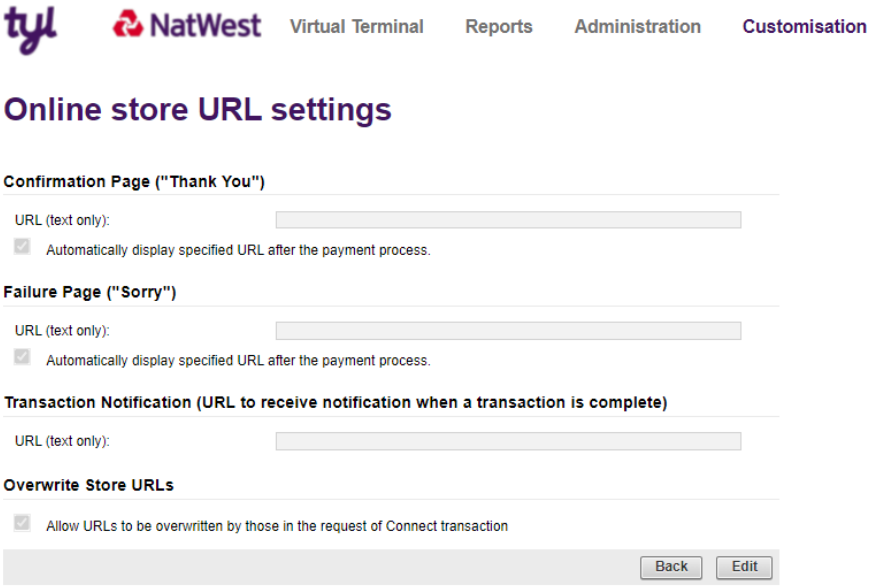
10. Transaction response

10.1 Response to your success/failure URLs

When the transaction's complete, the details will be sent back to the defined 'responseSuccessURL' or 'responseFailURL' as hidden fields. You can define these URLs in your transaction request.

Field Name	Possible Values
approval_code	This is the approval code for the transaction. The first character of this parameter is the most helpful indicator for verification of the transaction. 'Y' indicates that the transaction has been successful 'N' indicates that the transaction hasn't been successful "?" indicates that the transaction has been successfully initialised, but that a final result isn't yet available (as the transaction is now in a 'waiting' status). The transaction status is updated at a later stage.
oid	Order ID
refnumber	Reference number
status	This is the transaction status, e.g. 'APPROVED', 'DECLINED' (by authorisation endpoint or due to fraud prevention settings), 'FAILED' (wrong transaction message content/ parameters, etc.) or WAITING.
txndate_processed	The time the transaction was processed.
ipgTransactionId	The transaction identifier assigned by us.
tdate	Data and timestamp for the transaction.
fail_reason	The reason the transaction failed.
response_hash	Hash-Value to protect the communication (see the note below).
extended_response_hash	Hash-Value to protect the communication, where all response parameters are included in the hash calculation.
processor_response_code	The response code provided by the authorisation system of the payment method. Please note that response codes can be different depending on the payment method and the authorisation system.
fail_rc	The internal processing code for failed transactions.
terminal_id	The terminal ID used for transaction processing.
ccbin	The 6-digit identifier of the card issuing bank.
cccountry	The 3-letter alphanumeric ISO code of the cardholder's country (e.g. USA, DEU, ITA, etc.), filled with 'N/A' if the cardholder's country can't be determined.
ccbrand	The credit or debit card brand: MASTERCARD DINERSCLUB VISA AMEX MAESTRO Filled with 'N/A' for any payment method which isn't a credit or debit card.

Alternatively, you can define 'responseSuccessURL' or 'responseFailURL' in the Customisation tab of our Virtual Terminal, by selecting the "*Define the URLs for the integration with your online store*" option. (screenshot below)



For 3D Secure transactions only:

response_ code_3dsecure	Return code indicating the classification of the transaction: 1 – Successful authentication (VISA ECI 05, Mastercard ECI 02) 2 – Successful authentication without AVV (VISA ECI 05, Mastercard ECI 02) 3 – Authentication failed (Issuer rejected the Authentication) 4 – Authentication attempt (VISA ECI 06, Mastercard ECI 01) 5 – Unable to authenticate / Directory Server not responding (VISA ECI 07) 6 – Unable to authenticate / Access Control Server not responding (VISA ECI 07) 7 – Cardholder not enrolled for 3D Secure (VISA ECI 06) 8 – Invalid 3D Secure values received, most likely by the credit card issuing bank's Access Control Server (ACS) 9 – Challenge requested (Whitelist prompt requested if challenge required) Please see the note about blocking ECI 7 transactions in the 3D Secure section of this document.

Your custom fields and billing/shipping fields will also be sent back to the specified URL.

When you're integrating, it's worth knowing that new response parameters may be added from time to time, relating to product enhancements and new functionality.

10.2 Creating the extended response hash

Step 1: Retrieve all non-empty Gateway specified response parameters and then remove the parameter 'extended_response_hash' from your list, so that it will not get included in the hash calculation. Consider also that shared secret will be used as a key in HMAC to calculate the hash and the hash algorithm must be the same as the one that you have set in the transaction request.

Step 2: Sort the response parameters in ascending order of the parameter names, where the upper-case characters come before the lower case (based on ASCII value). Join the parameters' values to one string with pipe separator (use only parameters' values and not the parameters' names).

Step 3: Pass the created string to the HMAC algorithm while using shared secret ('sharedsecret') as a key for calculating the hash value.

Step 5: Encode the result of HMAC with Base64 to generate the extended response hash.

Only HMAC algorithm (i.e.: HMACSHA256, HMACSHA384 or HMACSHA512) is supported for generating the extended response hash.

10.3 Server-to-Server Notification

In addition to the response you receive, in hidden fields, to your '`responseSuccessURL`' or '`responseFailURL`', we can send server-to-server notifications with the above result parameters to a defined URL. This is really useful for keeping your systems in sync with the status of a transaction. To use this notification method, you can specify a URL in the 'Customisation' section of the Virtual Terminal or submit the URL in the following extra transaction parameter '`transactionNotificationURL`'.

It's worth noting that:

- The Transaction URL is sent as received. Don't add additional escaping (e.g. using %2f for a Slash (/).
- No SSL handshake verification of SSL certificates will take place in this process.
- The Notification URL needs to listen either on port 80 (http) or port 443 (https) – other ports aren't supported.
- The response hash parameter for validation (using the same algorithm that you've set in the transaction request) '`notification_hash`' is calculated as:

```
chargetotal + sharedsecret + currency + txndatetime + storename +
approval_code
```

11. Appendix 1 – How to generate an Extended Hash

Extended Hash example:
If you are using an HTML form to initiate a transaction, your request needs to include a security hash for verification of the message integrity.

The hash (parameter 'hashExtended') needs to be calculated using all non-empty gateway specified request parameters in ascending order of the parameter names, where the shared secret (parameter 'sharedsecret') must be used as the secret key for calculating the hash value. The gateway sorts the request parameters in the "natural order". For strings this means the "Lexicographic Order", thus the upper-case characters come before the lower case (based on ASCII value).

The request parameters that are not specified in our solution can still be submitted in your request to the gateway, but they must be excluded from the hash calculation. They will be ignored during processing and returned in the response.

When you are using Direct Post, there is also an option where you do not need to know the card details (PAN, CVV and Expiry Date) for the hash calculation. This will be managed with a specific setting performed on your store. Please contact your local support team if you want to enable this feature.

Creating the hash with all parameters

Transaction request values used for the hash calculation can be considered as a set of mandatory as well as optional gateway specified request parameters depending on the way you decide to build your request. See an example below:

```
chargetotal= 13.00
checkoutoption = combinedpage
currency= 978
hash_algorithm=HMACSHA256
paymentMethod=M
responseFailURL=https://localhost:8643/webshop/response_failure.jsp
responseSuccessURL=https://localhost:8643/webshop/response_success.jsp
storename=10123456789
timezone= Europe/Berlin
transactionNotificationURL=https://localhost:8643/webshop/transactionNotifica
tion
txndatetime= 2021:09:06-16:43:04
txntype=sale
sharedsecret=sharedsecret
```
(to be used as the secret key for calculating the hash value)

The steps below provide the guidelines on how to calculate a hash, while using the values from our example.

Step 1. Extended hash needs to be calculated using all non-empty gateway specified request parameters in ascending order of the parameter names, where the upper-case characters come before the lower case (based on ASCII value). Join the parameters' values to one string with pipe separator (use only parameters' values and not the parameters' names).

```
stringToExtendedHash =
13.00|combinedpage|978|HMACSHA256|M|https://localhost:8643/webshop/response_f
ailure.jsp|https://localhost:8643/webshop/response_success.jsp|10123456789|Eu
rope/Berlin|https://localhost:8643/webshop/transactionNotification|2021:09:06
-16:43:04|sale
```

Corresponding hash string does not include 'sharedsecret', which has to be used as the secret key for the HMAC instead.

Step 2. Pass the created string to the HMACSHA256 algorithm and using shared secret as a key for calculating the hash value.

```
HmacSHA256(stringToExtendedHash, sharedsecret)
```

Step 3. Encode the result of HMACSHA256 with Base64 and pass it to the gateway as part of your request.

Base64:
EapafBqqOF6N/kch8USkHPGh+fwSko24h6FpQnQHfQ8=

```
<input type="hidden" name="hashExtended" value="
EapafBqqOF6N/kch8USkHPGh+fwSko24h6FpQnQHfQ8="/>
```

Only HMAC algorithm (i.e.: HMACSHA256, HMACSHA384 or HMACSHA512) is supported for generating the extended request hash.

12. Appendix 2 – ipg-util.asp

```
<!-- google
CryptoJS for HMAC -
->

<script LANGUAGE=JScript RUNAT=Server src="script/cryptoJS/crypto-js.min.js"></script>
<script LANGUAGE=JScript RUNAT=Server src="script/cryptoJS/enc-base64.min.js"></script>
<script LANGUAGE=JScript RUNAT=Server>
    var today = new Date();
    var txndatetime = today.formatDate("Y:m:d-H:i:s");

    /*
        Function that calculates the hash of the following parameters as an example:
         - chargetotal
         - checkoutoption
         - currency
         - hash_algorithm
         - paymentMethod
         - responseFailURL
         - responseSuccessURL
         - storename
         - timezone
         - transactionNotificationURL
         - txndatetime
         - txntype
         - and sharedsecret as the secret key for calculating the hash value
     */

    function createExtendedHash(chargetotal, currency) {
        // Please change the storename to your individual Store Name
        var storename = "10123456789";
        // NOTE: Please DO NOT hardcode the secret in that script. For example read it from a database.
        var stringToExtendedHash =
chargetotal|checkoutoption|currency|hash_algorithm|paymentMethod|responseFailURL|responseSuccessURL|storename|timezone|transactio
nNotificationURL|txndatetime|txntype;

        var hashHMACSHA256 = CryptoJS.HmacSHA256(stringToExtendedHash, sharedSecret);
          var extendedhash = CryptoJS.enc.Base64.stringify(hashHMACSHA256);

        Response.Write(extendedhash);
    }

    function getDateTime() {
        Response.Write(txndatetime);
    }
</script>
```

13. Appendix 3 – ipg-util.php

```
<!DOCTYPE HTML>
<html>
<head><title>IPG Connect Sample for PHP</title></head>
<body>
<p><h1>Order Form</h1>

<form method="post" action="https://test.ipg-online.com/connect/gateway/processing">

<fieldset>
        <legend>IPG Connect Request Details</legend>
        <p>
            <label for="storename">Store ID:</label>
            <input type="text" name="storename" value="10123456789" readonly="readonly" />
        </p>
        <p>
            <label for="timezone">Timezone:</label>
            <input type="text" name="timezone" value="Europe/London"  readonly="readonly"/>
        </p>
        <p>
            <label for="chargetotal">Transaction Type:</label>
            <input type="text" name="txntype" value="sale" readonly="readonly" />
        </p>
        <p>
            <label for="chargetotal">Transaction Amount:</label>
            <input type="text" name="chargetotal" value="13.00" readonly="readonly" />
        </p>
        <p>
            <label for="currency">Currency (see ISO4217):</label>
            <input type="text" name="currency" value="978" readonly="readonly" />
        </p>
        <p>
            <label for="txndatetime">Transaction DateTime:</label>
            <input type="text" name="txndatetime" value="<?php echo getDateTime(); ?>"/>
        </p>
        <p>
            <label for="hashExtended">Hash Extended:</label>
               <input type="text" name="hashExtended" value="<?php echo
createExtendedHash('13.00', '978'); ?>"  readonly="readonly" />
        </p>
        <p>
            <label for="hashExtended">Hash Algorithm :</label>
            <input type="text" name="hash_algorithm" value="HMACSHA256" readonly="readonly" />
        </p>
        <p>
            <label for="hashExtended">Checkout option :</label>
            <input type="text" name="checkoutoption" value="combinedpage" readonly="readonly" />
        </p>
        <p>
            <input type="submit" id="submit" value="Submit" />
        </p>
    </fieldset>

</form>

<?php

function getDateTime() {
    return date("Y:m:d-H:i:s");
}

function createExtendedHash($chargetotal, $currency) {
// Please change the store Id to your individual Store ID
// NOTE: Please DO NOT hardcode the secret in that script. For example read it from a database.
$sharedSecret = "sharedsecret";
$separator = "|";
$storeId= "10123456789";
$timezone= "Europe/London";
$txntype= "sale";
$checkoutoption = "combinedpage";

$stringToHash = $chargetotal . $separator . $checkoutoption . $separator . $currency . $separator
. "HMACSHA256" . $separator . $storeId . $separator . $timezone. $separator . date("Y:m:d-H:i:s")
. $separator . $txntype;

$hash = base64_encode(hash_hmac('sha256', $stringToHash, $sharedSecret, true));
return $hash;
}
```

```
?>
</body>
</html>
```

The above is the working PHP example, to run it you can copy the above and paste it on https://www.w3schools.com/php/phptryit.asp?filename=tryphp_function1

14. Appendix 4 – Payment Method List

If you let your customer select the payment method in your website or want to define the payment method yourself, simply submit the parameter 'paymentMethod' in your transaction request. If you don't submit this parameter, we'll display a hosted page to the customer – which lets them choose from the payment methods that are enabled for your store.

Payment Method	**Value**
American Express	`A`
Apple Pay	`applePay`
Discover	`C`
Google Pay	`googlePay`
Maestro	`MA`
Maestro UK	`maestroUK`
Mastercard	`M`
Visa (Credit/Debit/ Electron/Delta)	`V`
Paypal	`paypal`

15. Appendix 5 – Apple Pay and Google Pay

Apple Pay and Google Pay e-wallet payment methods are presented to cardholders by default when using our "Combined" Hosted Payment Page. If your cardholders don't currently see these options when making a payment and would like to enable it, please get in touch with our support teams who can assist you with this.

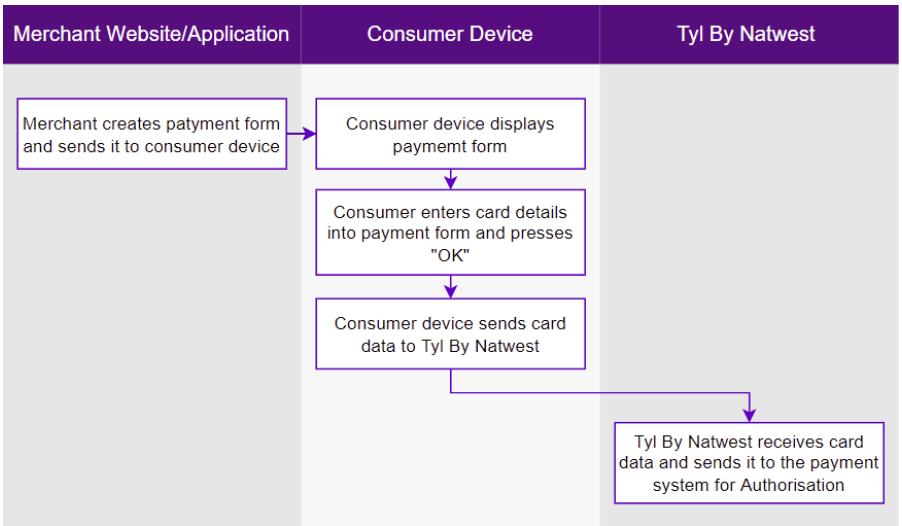
Please Note: Apple Pay and Google Pay transactions are SCA compliant by default.

16. Appendix 6 – Using your own form to capture data

If you decide to create your own forms, i.e.: Direct Post (not to use the ones provided by us), there are additional mandatory fields that you need to include. These fields are listed below.

Using Direct Post allows you to have full control over the look and feel of the form where your customers enter their card details for payment while simultaneously avoiding the need to have sensitive card data within your systems.

It is also important that you check if JavaScript is activated in your customer's browser. If necessary, inform your customer that JavaScript needs to be activated for the payment process.



16.1 Capture payment details

After your customer has decided how to pay, you present a corresponding HTML-page with a form to enter the payment data as well as hidden parameters with additional transaction information. In addition to the mandatory fields found in Mandatory fields section, your form needs to contain the following fields.

For Credit/Debit Card payment requests

Field Name	Description and Possible Values
`cardnumber`	Cardholders card number or PAN Min Length: 12 Max Length: 24
`expmonth`	The expiry month of the card (2 digits) i.e: MM
`expyear`	The expiry year of the card (4 digits) i.e: YYYY

cvm	This is the card verification code also known as CVV and, in most cases found on the backside of the card. (3 or 4 digits)

17. Appendix 7 – Test Cards

Use the test cards to help you set-up your integration in the Test environment.

You can only use test card details in the test environment – they can't be used for testing in the live environment.

You can find a list of test cards in our Help & Support pages:

https://www.tylbynatwest.com/help-and-support/online-payments/setting-up-online-payments