Hosted payment page integration guide

V1.2 | April 2021

NatWest Tyl offers you a quick and easy way to add payment capabilities to your website using a 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 1.2

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.

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.

1. 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. Make sure you can connect to the NatWest Tyl test URL with the Store ID and shared secret that you received by email : https://test.ipg-online.com/connect/gateway/processing. If you are using a shopping cart plugin you can enter these details when you set up the shopping cart.

2. Set up the URL on your website that you want to redirect the customer to when the payment is successful. You can configure this in the Virtual Terminal portal or provide with the payment request if you are integrating directly on your website. If you're using a shopping cart you should follow the set up instructions to set up the success URL.

3. Submit a sale using a test card in Appendix VI.

If you're happy with the results of the above integration you should:

4. Set up the URLS on your website that you want to redirect the customer to when the payment is not successful. You can configure these in the Virtual Terminal portal or provide with the payment request. If you're using a shopping cart you should follow the set up instructions to set up the failure URL.

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 set up instructions set up the notification URL.

6. Familiarise yourself with performing a Refund against a successful payment via the Virtual Terminal portal.

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 and to 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 consider:

8. Submitting an Pre authorisation transaction and subsequently completing it via the Virtual Terminal portal.

9. Using 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.

Once you're happy that the integration is working as you would expect, you should:

10. Point your website or shopping cart integration to the Live URL with the Store ID and 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.

2. Your integration options

There are a few ways to integrate the hosted payment page.

2.1 Checkout option 'classic'

This option splits the payment process into multiple pages where you can easily decide what information you want to collect via the hosted page and what you'd rather collect yourself.

For example, you can let your customers select their preferred payment method from your website and submit that to us or, if you'd prefer not to send the payment method, we'll automatically show a payment method selection page to your customer – where they can choose from all the methods available for your store.

You can let us know what data you want us to capture by choosing one of three options below:

- Payonly: shows a hosted page that collects basic payment data (e.g. the cardholder name, card number, expiry date and security code).
- Payplus: in addition to the above, this collects a full set of billing information on an additional page.
- Fullpay: in addition to the above, this displays a third page that also collects shipping information.

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

2.2 Checkout option 'combinedpage'

The checkout option 'combinedpage' consolidates the payment method choice and the typical next step (e.g. entry of card details) in a single page which is automatically optimised for different kinds of user devices, e.g. PC, smartphone or tablet.

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

- Please note, that this checkout option has some functional limitations in comparison to the 'classic' option and may not work with older browsers.

2.3 Checkout option 'simpleform'

'Simpleform' is a basic option for only capturing sensitive data (for example, the card number and/or security code). It's displayed within an iFrame embedded in your website.

It works with HTML 5 enabled browsers such as Microsoft Internet Explorer 9 onwards, Google Chrome and Mozilla Firefox – if the minimum size of the iFrame is: 900 px (height) and 460 px (width). You can also send a hex colour code to easily align the background colour of the buttons in an iFrame (to look and feel like your website).

When using an iFrame, you need to have a mechanism on your website to receive the response from the iFrame after the processing by us is complete. In order to do so, you need to register a callback method, e.g. receiveMessage(), with a Window EventListener, listening on the Javascript 'message' event. The callback method will have the 'event' object. This object will contain all the information needed to process the response from the iFrame.

Examples of code used to integrate the simplified hosted payment form with the checkout option 'simpleform' can be found in Appendix V.

3. Getting started

This section tells you how to integrate your website using the 'classic' checkout option in 'payonly' mode. Examples are provided using ASP and PHP.

3.1 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: 10123456789

- Shared Secret
  This is the shared secret we sent to you by email.
  This is used when constructing the hash value (see below).

3.2 ASP Example

The following ASP example demonstrates a simple page that'll communicate with us in 'payonly' mode.

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 location of the receipt page can be configured:

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

<html>
    <head><title>Example 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/London"/>
        <input type="hidden" name="txndatetime" value="<% getDateTime() %>"/>
        <input type="hidden" name="hash_algorithm" value="SHA256"/>
        <input type="hidden" name="hash" value="<% call createHash( "13.00","826" ) %>"/>
        <input type="hidden" name="storename" value="10123456789" />
        <input type="hidden" name="mode" value="payonly"/>
        <input type="hidden" name="paymentMethod" value="M"/>
        <input type="text" name="chargetotal" value="13.00" />
        <input type="hidden" name="currency" value="826"/>
        <input type="submit" value="Submit">
    </form>
    </body>
</html>
```

The code shown in Appendix II 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 send in 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).

3.3 PHP Example

This PHP example demonstrates a simple page that'll communicate with us in 'payonly' mode.

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 URL for the receipt page can be configured in the Virtual Terminal portal:

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

<html>
<head><title>Example for PHP</title></head>

    <body>
    <h1>Order Form</h1>

    <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/London"/>
    <input type="hidden" name="txndatetime" value="<?php echo getDateTime() ?>"/>
    <input type="hidden" name="hash_algorithm" value="SHA256"/>

    <input type="hidden" name="hash" value="<?php echo createHash( "13.00","826" ) ?>"/>
    <input type="hidden" name="storename" value="10123456789"/>
    <input type="hidden" name="mode" value="payonly"/>
    <input type="hidden" name="paymentMethod" value="M"/>
    <input type="text" name="chargetotal" value="13.00"/>
    <input type="hidden" name="currency" value="826"/>

    <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 sent you or contact us and we'll share it with you.

The code shown in Appendix III 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 send in transactions for this store.

4. Mandatory fields

Depending on the transaction type, the following form fields must be included in the form submitted to us. Please refer to this Integration Guide's 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 time of the transaction)
hash_algorithm	This is to indicate the algorithm that you use for hash calculation. The possible values are: SHA256 and SHA512.
hash	This is a SHA hash of the following fields: storename + txndatetime + chargetotal + currency + sharedsecret. Note, that it's important to have the hash generated in this exact order. An example of how to generate a SHA-256 hash is given in Appendix I. Either hash SHA256 or hash SHA512 should be used, not both parameters.
storename	This is the ID of the store provided by Tyl.
made	'fullpay', 'payonly' or 'payplus' (the chosen mode for the transaction when using the 'classic' checkout option).
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 like1,000.01 / 1.000,01, aren't allowed.
currency	The numeric ISO code for the transaction currency. The ISO code for GBP is 826.

5. 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 allows you to set the checkout option to: • 'classic' for a payment process that's split into multiple pages, • '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, • 'simpleform' to only capture the sensitive data by using the simplified hosted payment form displayed within an iFrame embedded in the context of your website.
comments	Place any comments about the transaction here.
customerid	This field allows you to transmit any value, e. g. your ID for the customer.
invoicenumber	This field allows you to transmit any value, e. g. an invoice number or class of goods. The maximum length for this parameter is 48 characters.
hashExtended	The extended hash is an optional security feature that allows you to include all parameters of the transaction request. It needs to be calculated using all request parameters in ascending order of the parameter names.

hexColorCode	This is an optional parameter when you use the simplified iFrame integration with the checkout option 'simpleform'. The 'hexColorCode' parameter allows you to easily align the background colour of the buttons in an iFrame to the look and feel of your website. You can send e.g. '#9c22ce' then the colour of buttons will be 'violet' but when you send 'hexColorCode' set with non-existing hex colour code, there won't be any impact.
hostURI	This is a mandatory parameter when you use the simplified iFrame integration with the checkout option 'simpleform'. The 'hostURI' parameter (with an upper case 'I') allows you to send the URI of the page where an iFrame is hosted, in order to dissolve an iFrame after the sensitive data's been submitted, i.e. the response parameters go to the parent window.
item1 up to item 999	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

The following values are currently possible:<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.</td></tr>
<tr><td>mobileMode</td><td>If your customer uses a mobile device to shop on your website, you can submit this parameter with the value ‘true’. This’ll lead your customer to a payment page flow that’s been specifically designed for mobile devices.</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.</td></tr>
</table>

paymentMethod	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 to the customer to choose from the payment methods available for your website. For valid payment method values, please refer to Appendix IV.
Ponumber	This field allows you to submit a Purchase Order Number of up to 50 characters.
Refer	This field describes who referred the customer to your store.
responseFailURL	The URL where you wish to direct customers after a declined or unsuccessful transaction. If provided, the value will override the URL provided under 'Customisation' in the Virtual Terminal portal.
responseSuccessURL	The URL where you wish to direct customers after a successful transaction. If provided, the value will override the URL provided under 'Customisation' in the Virtual Terminal portal.
shipping	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' 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.

6. 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:

6.1 Billing address

The following table outlines the format of these additional fields:

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

6.2 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	Limit of 96 characters, including spaces	Shipping address Line 1
saddr2	Limit of 96 characters, including spaces	Shipping address Line 2
Scity	Limit of 96 characters, including spaces	Shipping city
sstate	Limit of 96 characters, including spaces	State, province or territory
scountry	2 Letter Country Code	Country of shipping address
szip	Limit of 24 characters, including spaces	Zip or postal code

7. 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. If you want to use this feature, please send the custom fields in the format customParam_key=value.

Example:

<input type="hidden" name="customParam_color" value="green"/>

8. 3D Secure

You can authenticate transactions using Verified by Visa and Mastercard SecureCode 3D Secure solutions. 3D Secure helps protect your business in the event a payment is made fraudulently by passing the liability of any chargebacks to the cardholder's 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".

9. Transaction response

9.1 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:

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).
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 VISA AMEX DINERSCLUB JCB CHINA_UNION_PAY CABAL MAESTRO RUPAY BCMC SOROCRED Filled with 'N/A' for any payment method which isn't a credit or debit card.

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 / incorrect password (transaction declined) **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) Please see 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.

The parameter 'response_hash' allows you to recheck if the received transaction response has really been sent by us; this protects you from fraud. The value is created with a SHA Hash, using the following parameter string:

sharedsecret + approval_code + chargetotal + currency + txndatetime + storename

The hash algorithm is the same as the one you've set in the transaction request.

You must implement the response hash validation so – when doing so – it's worth remembering to store the 'txndatetime' you've submitted with the transaction request (in order to validate the response hash). Also, you must always use the https-connection (instead of http) to keep transaction details secure.

9.2 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 below:

chargetotal + sharedsecret + currency + txndatetime + storename + approval_code

APPENDIX I

How to generate a SHA-256 Hash or Extended Hash

Example of Hash

- storename = 98765432101
- txndatetime = 2013:07:16-09:57:08
- chargetotal = 1.00
- currency = 826
- sharedsecret = TopSecret

Step 1: Collect the selected parameters: storename, txndatetime, chargetotal, currency and sharedsecret and join the parameters' values to one string (only use the parameter values, not the parameter names).

987654321012013:07:16-09:57:081.00826TopSecret

Step 2: Convert the created string to its ascii hexadecimal representation.

3938373635343332313031323031333a30373a31362d30393a35373a3038312e303038323654 6f70536563726574

Step 3: Pass the ascii hexadecimal representation of the created string to the SHA-256 algorithm.

SHA256(3938373635343332313031323031333a30373a31362d30393a35373a3038312e3030383 236546f70536563726574)

Step 4: Use the value returned by the SHA-256 algorithm and submit it to us in the below form.

3d7e75aa0b4e0e1d4a7ac87e451e64692cced46f4358ef35a69d96721341243c

<input type="hidden" name="hash" value="3d7e75aa0b4e0e1d4a7ac87e451e64692cced46f4358ef35a69d96721341243c"/>

Example of Extended Hash

- P1 = abc
- P2 = xyz
- P3 = ccc
- sharedsecret = TopSecret
- t1=zzz
- t2=yyy

Step 1: Extended hash needs to be calculated using all request parameters in ascending order of the parameter names, adding sharedsecret at the end. You should join the parameters' values to one string (use only parameter values, not parameter names).

abcxyzccczzzyyyTopSecret

Step 2: Convert the created string to its ascii hexadecimal representation (as shown below).

393837363534333231303132303133 3a30373a31362d30393a35373a3038312e303038323654 6f70536563726574

Step 3: Pass the ascii hexadecimal representation of the created string to the SHA-256 algorithm.

SHA256(393837363534333231303132303133 3a30373a31362d30393a35373a3038312e3030383 236546f70536563726574)

Step 4: Use the value returned by the SHA-256 algorithm and submit it to us in the below form.

3d7e75aa0b4e0e1d4a7ac87e451e64692cced46f4358ef35a69d96721341243c

<input type="hidden" name="hashExtended" value="3d7e75aa0b4e0e1d4a7ac87e451e64692cced46f4358ef35a69d96721341243c"/>

APPENDIX II

ipg-util.asp

```
<!-- sha1.js contains also helper functions (dateFormatter, charToByte, byteToHex, ...) -->
<script LANGUAGE=JScript RUNAT=Server src="sha1.js">
</script>

<!-- google CryptoJS for SHA256 -->
<script LANGUAGE=JScript RUNAT=Server src="sha256.js">
</script>

<script LANGUAGE=JScript RUNAT=Server>
    var today = new Date();
    var formattedDate = today.formatDate("Y:m:d-H:i:s");

    /*
        Function that calculates the hash of the following parameters:
        - Store Id
        - Date/Time(see $dateTime above)
        - chargetotal
        - currency (numeric ISO value)
        - shared secret
    */

    function createHash(chargetotal, currency) {
        // Please change the store Id to your individual Store ID
        var storeId = "10123456789";
        // NOTE: Do not hardcode the shared secret.

        var sharedSecret = "sharedsecret";

        var stringToHash = storeId + formattedDate + chargetotal + currency + sharedSecret;

        var ascii = getHexFromChars(stringToHash);

        var hash = CryptoJS.SHA256(ascii);

        Response.Write(hash);
    }
```

```
function getHexFromChars(value) {
        var char_str = value;
        var hex_str = "";
        var i, n;
        for(i=0; i < char_str.length; i++) {
            n = charToByte(char_str.charAt(i));
            if(n != 0) {
                hex_str += byteToHex(n);
            }
        }
        return hex_str.toLowerCase();
    }

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

APPENDIX III

ipg-util.php

```
<?php
    // Timezeone needs to be set
    date_default_timezone_set('Europe/London');
    $dateTime = date("Y:m:d-H:i:s");

    function getDateTime() {
        global $dateTime;
        return $dateTime;
    }

    /*
        Function that calculates the hash of the following parameters:
        - Store Id
        - Date/Time(see $dateTime above)
        - chargetotal
        - currency (numeric ISO value)
        - shared secret
    */

    function createHash($chargetotal, $currency) {
        // Please change the store Id to your individual Store ID
        $storeId = "10123456789";
        // NOTE: Do not hardcode the shared secret.
        $sharedSecret = "sharedsecret";

        $stringToHash = $storeId . getDateTime() . $chargetotal . $currency . $sharedSecret;

        $ascii = bin2hex($stringToHash);

        return hash("sha256", $ascii);
    }
```

APPENDIX IV

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 and supported for the combination of the consumer's country and the transaction currency.

Payment Method	**Value**
American Express	A
Maestro	MA
Maestro UK	maestroUK
Mastercard	M
Visa (Credit/Debit/ Electron/Delta)	V

APPENDIX V

Code examples for the simplified hosted payment form

You should refer to the following section when you're integrating the simplified hosted payment form with the checkout option 'simpleform'.

When you're building a request with the checkout option 'simpleform', as well as the mandatory fields you need for every payment request, you'll also need to include some specific fields in your transaction request.

The specific fields you'll need to consider are listed below:

Field Name	Description, possible values and format
checkoutoption	Set the value for this parameter to 'simpleform'.
hostURI	Set the value for this parameter to the URI (with an upper case 'I') of the page where an iFrame is hosted. For the cross-domain communication between Tyl and your website, the HTML 5 postmessage API is utilised, therefore, for security reasons, it's mandatory that you send the value for the URI of your website to let us know where sensitive transaction data has to be sent once the iFrame gets dissolved. e.g. hostURI="https://www.mywebsite.com"
hexColorCode	Set the value for this parameter when you want to match the background colour of the buttons in an iFrame to the look and feel of your website. e.g. hexColorCode="#9c22ce"

Here's an example of a form with the minimum number of fields. 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.

```
<form id="checkoutForm" target="myFrame" method="post" action="https://test.ipg-online.com/
connect/gateway/processing">
<input type="hidden" name="checkoutoption" value="simpleform">
<input type="hidden" name="hostURI" value="https://www.mywebsite.com/.../.../">
```

```
<input type="hidden" name="txntype" value="preauth">
<input type="hidden" name="timezone" value="Europe/London"/>
<input type="hidden" name="txndatetime" value="<% getDateTime() %>"/>
<input type="hidden" name="hash_algorithm" value="SHA256"/>
<input type="hidden" name="hash" value="<% call createHash( "13.00","826" ) %>"/>
<input type="hidden" name="storename" value="12123456789" />
<input type="hidden" name="chargetotal" value="13.00" />
<input type="hidden" name="currency" value="826"/>
<input type="submit" value="Submit">
</form>
```

This is a JSP example, showing how the simplified hosted payment form is hosted in an iFrame:

```
<div id="embeddableConnect">
    <table border="0" cellpadding="0" cellspacing="0" width="100%">
        <tbody>
            <tr>
                <iframe name="myFrame" id="myFrame" src="#" width="460px"
                height="900px"
                style="border: none;">
                Your browser does not support inline frames.
                </iframe>
            </tr>
        </tbody>
    </table>
</div>
```

Below is a JavaScript example, showing the submission of the request to us, via the simplified hosted payment form, and the need to add the event listeners:

```
function submitForm() {

    ----
    ----
    var obj1 = document.getElementById('checkoutoption_simpleform');
    if(obj1.checked){
        document.myForm.target = "myFrame";
        obj2.style.visibility = 'visible';
        obj2.style.display = 'block';
        window.addEventListener("message", receiveMessage, false);
        document.myForm.submitFormBtn.disabled = true;
    }
    ----
    ----

    }

function receiveMessage(event){
    var hostName = getHostName();
    if (event.origin != "https://"+getHostName())
        return;
```

```
        forwardForm(event.data);
}

function forwardForm(responseObj) {
        var newForm = document.createElement("form");
        newForm.setAttribute('method',"post");
        newForm.setAttribute('action',responseObj.redirectURL);
        newForm.setAttribute('id',"newForm");
        newForm.setAttribute('name',"newForm");
        document.body.appendChild(newForm);
        var elementArr = responseObj.elementArr;
        for(j=0 ; j<elementArr.length; j++){
                var element = elementArr[j];
                var input = document.createElement("input");
                input.setAttribute("type", "hidden");
                input.setAttribute("name", element.name);
                input.setAttribute("value", element.value);
                document.newForm.appendChild(input);
        }
        document.newForm.submit();
}
```

You'll need to register a method with the listener, e.g. 'receiveMessage'. To learn more about adding event listeners and Javascript MessageEvent, have a look at the below links:

- https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener
- https://developer.mozilla.org/en-US/docs/Web/API/MessageEvent

Once the transaction is complete, we use window.postMessage() to send the sensitive transaction data back to the parent window. Then receiveMessage() is called. This creates a new form with all the sensitive transaction data and submits it to the preconfigured successful/failure URL while dissolving the iFrame.

The window.postMessage() method safely enables cross-origin communication between the parent and the child window objects, i.e. between your website and the child iFrame embedded within it. Generally, scripts on different pages are allowed to access each other if, and only if, the pages they originate from share the same protocol, port number and host – window. postMessage() means this isn't necessary.

APPENDIX VI

Test cards

Use these test cards to help you set up your integration with our Test environment.

1. If you change the password or enter the wrong value for Verified by Visa or SecureCode, your test payment might not turn out as you expected.

2. When you're testing situations where your customer doesn't authenticate successfully to Verified by Visa or SecureCode, you don't need to use the password (below) just use the 'cancel' button in the pop-up window.

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

Visa	
Card number	4035874000424977
Expiry date	Any date in future
Card security code	Any three digit value
Verified by Visa password	Secret!33

Mastercard	
Card number	5426064000424979
Expiry date	Any date in future
Card security code	Any three digit value
SecureCode password	Secret!33