Secure Hosting and Payments Technical Integration Guide

Version	13.8.20
Released	July 2018
Description	Integrating your website or payment system into the Secure Hosting and Payment ecommerce gateway platform

Document Index

Welcome to the Secure Hosting and Payments (SHP) Integration Guide

The Secure Hosting and Payments (SHP) ecommerce gateway platform provides a secure, simple method of processing credit, debit or payment card transactions as well as PayPal e-wallet transactions from your website or payment system.

The SHP system provides a simple, straightforward payment interface which allows you to process real-time transactions with your merchant acquiring bank. The SHP system supports merchant sites by simplifying a whole range of services, in particular the system encrypts and safely stored the card details to ensure safety and security and reduce the impact of regulations on the services you offer.

This document explains how your website or payment system can integrate to our SHP platform, along with its related features and start processing payments in real-time.

Your SHP Username and Password

The username which you use to log into your SHP account is also your Secure Hosting account number (i.e. SH2XXXXX). This will never change and cannot be changed either by yourself or upon request to us.

Once registered with Secure Hosting & Payments you will be provided with a temporary password for your account. You must change this password as soon as possible by following the "Forgotten Password" link on the login screen. In order to remain secure, your password must be a minimum of 7 characters in length and be alphanumeric (contain both letters and numbers).

Your password will expire every 60 days. During the last several days of each 60 day period you will be reminded upon login that your password is due to expire. You must then follow the "Forgotten Password" link again to reset your password, making it different to a previously used password.

Basic Requirements

What you will need to complete the integration process and to start processing payments:

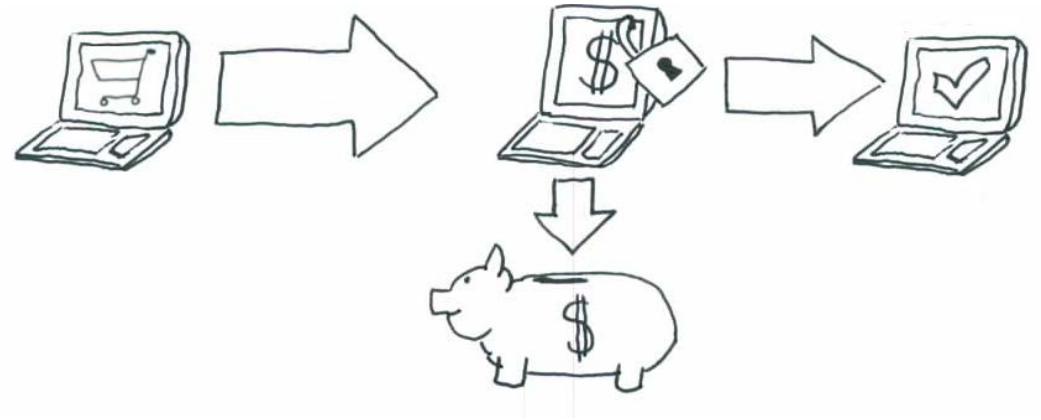
- A merchant account with a supported bank that is configured for internet channel. For a list of our supported banks, please see Appendix A or click here.
- A PayPal account is optional but required to support the specific e-wallet partner.
- A secure hosting account.
- A website /webshop to initiate the payment process, and knowledge of how it is built/operates.
- Skills in HTML mark up.
- Skills in CSS styling and JavaScript are optional but recommended.
- A HTML editor.
- Access to the internet.
- Access to our test cards (details in Appendix C).
- The estimated time to integrate is between 1 hour and 1 day depending on the developers' skill and experience.

Overview of how the integration works

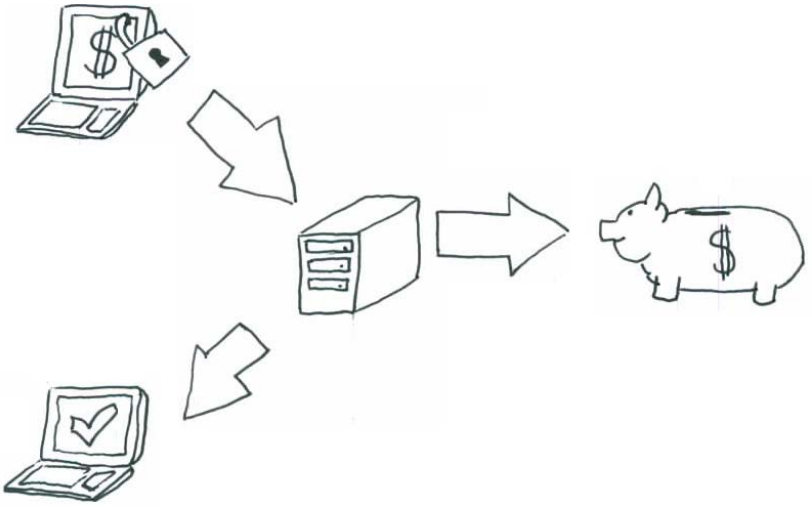
The end goal is for the cardholder to get to a “Pay Now” button and be able to choose their payment type and pay quickly and easily without abandoning the cart, ensuring a secure and fast transaction.

There are two different ways of taking payments:

1. The Hosted Form method where the customer is redirected from the final checkout page on your website, to a fully customised and styled payment page hosted on our PCI:DSS compliant ecommerce environment.



2. The API method allows the customer to experience a smooth, uninterrupted checkout on your website. Your website will capture the payment details and submit them directly to the SHP system for processing.



Recommendations

If you wish to provide a checkout flow, for your customer, that keeps them on your website, or if you have a service providing system which requires payments and you don’t mind taking on PCI:DSS compliance than we recommend the API integration.

If you want a simple, quick and easy set up which has the minimum amount of requirements for PCI:DSS compliance then we recommend our Hosted Form integration, the most commonly used integration into SHP.

The Hosted Form Integration Process in Detail

The Hosted Form integration will work as the final page of your website's checkout process; this means the customer only has to be redirected away from your website for as little time as possible. After the customer has completed their basket and entered all the required information into your website, e.g. name and address, the customer is then redirected to the payment page located on the SHP website.

Step 1: Redirecting the Customer

The first step of integrating into the SHP system is to redirect the customer to the payment page hosted on the SHP website. The redirect is performed using a HTML form POST to the generic payment page parser:

```
<form action="https://www.secure-server-hosting.com/secutran/secuitems.php" method="post">
```

With this form, the one required field you must include tells our system which payment page to use; as the payment page parser is generic it will not automatically know to grab your payment page. The one required field is called "filename", the value of this field must include your account's SH reference value and the file name of your payment page in the below format. The input field itself is usually a hidden field as it has no relevant value to the customer:

```
<input type="hidden" name="filename" value="SH200000/payment.html" />
```

Along with the one required field, you will also want your website to post a series of additional fields from your shopping cart as they allow you to dynamically configure the checkout page:

- shreference – Your account login reference
- checkcode – A second unique identifier for the account
- transactionamount – The total amount to be processed against the card
- transactioncurrency – The ISO 4217 alpha-3 currency code the transaction is to be processed in, please note, SHP will not perform any currency conversions.
- shippingcharge –The amount of shipping the customer has been charged, this will not be added to the transaction amount.
- transactiontax – The amount of tax the customer has been charged, please note, SHP does not perform any tax calculations for you.

If your website's checkout process captures the customer's information before redirecting the customer, the customer's details can be sent in the below fields so they can be captured and stored with the transaction:

- cardholdersname
- cardholdersemail
- cardholderaddr1
- cardholderaddr2
- cardholdercity
- cardholderstate
- cardholderpostcode
- cardholdercountry
- cardholdertelephonenumber

If you want to send any additional fields to the SHP system with your transaction, you can add them to the form. The field name you use will be the field name our system captures and stores the data in. While you can send as many fields as you require, within reason, the limit on the size of the field value is 255 characters, anything longer will be truncated. You should only send fields to Secure Hosting that are required for your checkout process. All information supplied to Secure Hosting as part of a checkout will be inspected by our Web Application Firewall in order ensure you and your customers are kept as safe as possible while using our service.

Test 1: Customer Redirect

In order to test your redirect form, you must first build your payment page template, detailed within the next section of this document. The reason you must complete the next step before testing is because the redirect is to the payment page, if no payment page has been built and uploaded the SHP account, the parser will return an error.

Once you have built up your redirect form and uploaded your payment page template you can run the first integration test!

To test the redirect, simply submit the redirect form from your website, if your payment page appears with all the transaction and customer data in the page then the test has been successful and you are 50% of the way to completing a basic integration.

Troubleshooting

If your payment page does not appear as expected, then something is wrong.

Error	Cause	Fix
You must pass a filename to the script.	Your redirect form does not contain a “filename” field.	Add a field to your redirect form called “filename” with the relevant value.
The file template.html does not exist.	The value of the “filename” is incorrectly formatted.	The format of the “filename” field is “SH200000/template.html”
The file SH200000/template.html does not exist.	Your payment page template has not been uploaded to your account.	Upload your payment page template via the File Manager interface within the client control panel.
None of my customer’s details appear in the payment page	The redirect form is not submitting the customer’s data with the correct field names.	Correct the field names within the redirect form.
	The dynamic fields have not been written into the payment page template correctly.	Correct the payment page template to use the syntax “$fieldname” directly in the raw HTML, no server side tags.

Step 2: Building a Payment Page

The second step when integrating your website into the SHP platform is to build a payment page. The payment pages within the SHP platform work on a HTML template basis, this allows the payment pages to be designed, built and styled using well established, standardised formats (HTML, CSS, JavaScript, etc). You upload the HTML templates into your SHP account via the File Manager interface, the template is then used by the generic payment page parser to output a payment page to the customer's browser. For more information about the File Manager interface, please see the File Manager section of the SHP Users Guide.

The filename of your payment page template is not set by the SHP system; you can define your own filename(s) allowing you to optionally have multiple payment pages which allows you to have multiple payment pages for other purposes (files must have either a .html or .htm extension. This is case insensitive). The way our payment page parser knows which payment page to use when you redirect your customer is via the "filename" field in the redirect form, this allows your website to choose which payment page it wishes to use for the customer.

Along with the "filename" field, you can post as many additional fields to your payment page as required to assist with configuring your checkout, or simply pass data to be captured with the transaction. In order to use those fields within the payment page, within the HTML template you would use the "$fieldname" syntax directly in the raw HTML, no service side tags or coding. If a field with the relevant field name has been posted from the redirect form to the payment page, the "$fieldname" text will be replaced with the value of the field supplied. If not field has been supplied then the parser will blank out the "$fieldname" text. As an extension of this, if you want to forward the value of a form field from your redirect form to the transaction, you can use the below syntax to forward individual field values:

```
<input name="fieldname" value="$fieldname" />
```

To forward the value behind without it being seen, you would make this a hidden input, to allow the customer to modify the field you can make it a text input. To repopulate drop down or multiple select boxes, we would advise using JavaScript within the head of the page, code similar the below example:

```
onload = function(){
    var preSelect = new Array( 'field1' => '$field1', 'field2' => '$field2' );
    for( var fieldName in preSelect ){
        var selectField = document.getElementsByName( fieldName )[0];
        for( var i = 0; i < selectField.options.length; i++ ){
            if( selectField.options[i].value == preSelect[fieldName] ){
                selectField.selectedIndex = i;
            }
        }
    }
};
```

We would advise having knowledge of JavaScript or other Object Orientated languages before using this. For further information regarding JavaScript and how to use it, please have a look at W3 Schools tutorial: http://www.w3schools.com/js/.

Within your payment page, in order for it to perform it's required function of capturing the sensitive payment card details and submitting them, you will need to build a HTML form within the template which will submit the transaction information to one of our transaction scripts:

```
<form action="https://www.secure-server-hosting.com/secutran/transactionsi1.php"
method="post">
```

Please see Appendix B for a list of all transaction scripts.

It is essential that the POST method is used on this form as the SHP transaction scripts will not accept transaction details via any other method.

The minimum, required fields that this form must post are:

- shreference
- checkcode
- transactionamount
- transactioncurrency
- cardnumber
- cardexpiremonth (2 digit month)
- cardexpireyear (2 digit year)
- cv2 (security code on reverse of the card)

The below fields are not strictly required, however we strongly advise always submitting them:

- transactiontax
- shippingcharge
- cardstartmonth (2 digit month)
- cardstartyear (2 digit year)
- cardholdersname
- cardholdersemail (required for confirmation emails if you are using them)
- cardholderaddr1 *
- cardholderaddr2
- cardholdercity
- cardholderstate
- cardholderpostcode *
- cardholdercountry
- cardholdertelephonenumber

* Required for the card issuing bank's Address Verification System.

Any additional fields you submit will be captured and stored using the field name they were submitted with.

As the payment pages work from a HTML template which is built by you, you've got the opportunity to design and style the page how you like; you can include your own CSS files, images and even external JavaScript files. The way to do this, in order to prevent client browser security warnings, you can upload the images, CSS and JavaScript file into your SHP account through the File Manager. You can then address the files using the below path and replacing the account SH reference and file name:

```
https://www.secure-server-hosting.com/secutran/secureforms/sh200000/image.gif
```

Test 2: Submit a Payment

Once you are successfully redirecting the customer to the payment page, the next step is to submit a transaction form the payment page. To perform this test, once you've been redirected to the payment page you can complete the payment form with one of the test card numbers found within Appendix C.

After submitting the transaction, have a look at the new transaction within the client control panel. If all the transaction fields are displayed correctly within the control panel then you have successfully completed a basic integration into the SHP system! Congratulations!

After you have completed the basic integration, there are further features of the platform which you can optionally integrate into.

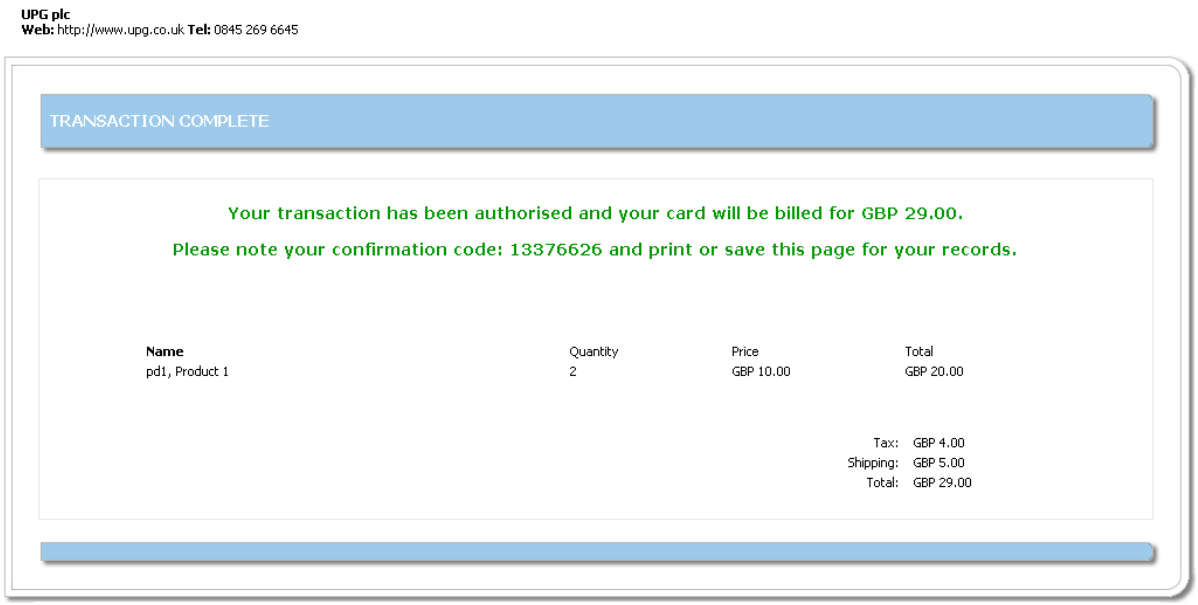
Troubleshooting

If the payment details do not appear in the client control panel or you receive the following error messages, something is wrong:

Error	Cause	Fix
SH reference not supplied. SH reference field empty.	The payment form is not submitting the account SH reference.	Make sure the payment form contains an input field called “shreference” which contains the account SH reference.
Checkcode not supplied. Checkcode field empty.	The payment form is not submitting the account check code.	Make sure the payment form contains an input field called “checkcode” which contains the account check code.
Merchant Implementation Error - Incorrect client SH reference and checkcode combination	The SH reference and checkcode submitted with the transaction are not correct for the account.	Make sure the SH reference and check code values are correct.
Merchant Implementation Error - Invalid Transaction Amount.	The transaction amount is not of a valid format.	Ensure the transaction amount value submitted is formatted as a currency amount with no symbol, for example 10.00.

Step 3: Building a Confirmation Page

After a transaction has been processed by the SHP system, it does not automatically redirect customers back to your website, it outputs a confirmation page. The default confirmation page outputted by the SHP system looks like this:



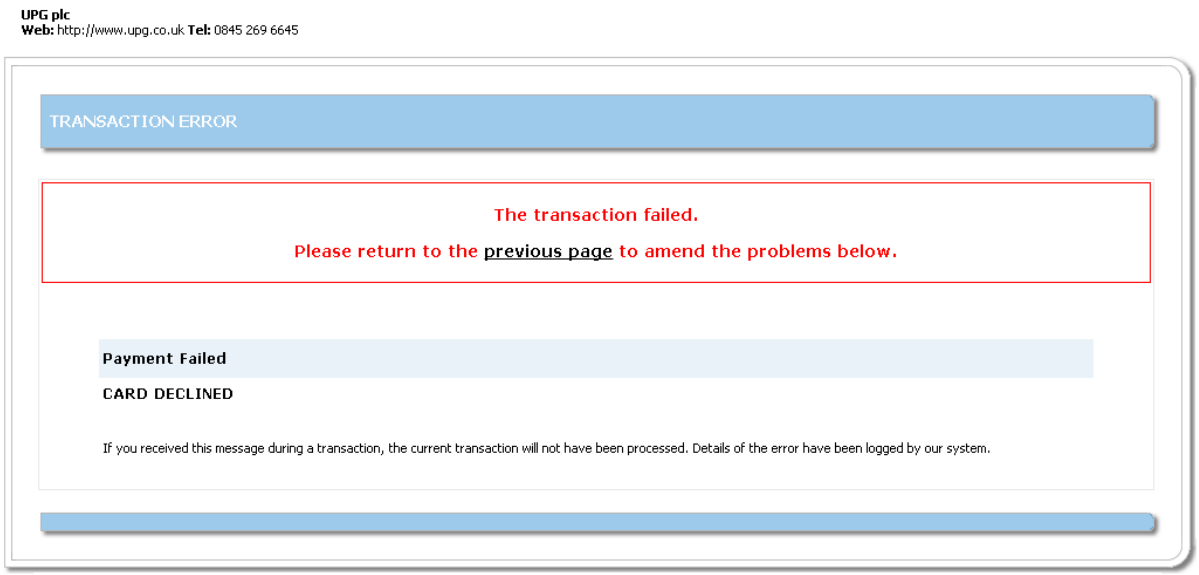
Using the Confirmation/Error Page Settings within the SHP client login system, you are able to insert your own company logo along with your own text at either the top or bottom of this page.

Alternatively you can build and design your own confirmation page using the same HTML template style as the payment page, this allows you to style the confirmation page to match the design style of your website or perform required actions like redirecting the customer back to your site. Unlike the payment pages, the file name of the confirmation page is set by the SHP system; the file name must be “htmlgood.html”. This ensures the SHP system can adequately identify a customised confirmation page within your account.

All the fields submitted from your payment form will be available within the confirmation page using the “$fieldname” syntax allowing you to display the customer’s details on the confirmation page. In addition to the field submitted by your payment form, the below fields are generated by our system:

- $authcode – Auth code returned by the bank
- $cv2avsresult – Result of the CV2 and AVS checks
- $trancdtp – Card type identified from the card number
- $trannum – The transaction ID of the customer’s transaction
- $transactiondate – The date of the transaction formatted as yyyy-mm-dd
- $transactiontime – The time of the transaction formatted as hh:mm:ss

In addition to building a confirmation page, there is also an error page within the platform which can be designed and styled in the same fashion as the confirmation page. The default transaction error page looks like this:



If you wish to upload your own transaction error page, the file name for the HTML template is "htmlbad.html". The system generated fields within the error page are:

- $tranerrdesc – A description of the type of error (e.g. Payment Failed)
- $tranerrdetail – Details of the error where applicable (e.g. Card Declined)
- $posted_fields – A collection of hidden HTML field fields, a hidden field for each form field posted from the payment form.

When the customer has a transaction error, after displaying the relevant error message to the customer, you may wish to redirect them back to their payment page in order to retry their transaction. This can be achieved by building a HTML form within the error page that will rebuild the checkout page for them. The form in the error page will post the customer back to the payment page parser, you can use the "$posted_fields" variable to build up the form fields and add in your own submit button:

```
<form action="https://www.seure-server-hosting.com/secutran/secuitems.php" method="post">
    $posted_fields
    <input type="submit" value="Go Back" />
</form>
```

In order to ensure the correct file name field value is populated into the return form, you need to forward the "filename" field value through the payment form otherwise the return form will not be able to specify the payment page template for the payment page parser. To forward the "filename" value in the payment form, you must use the below hidden input field:

```
<input type="hidden" name="filename" value="$backfile" />
```

Step 4: Building Product Data

With your customer's transaction, you will most likely need the SHP system to store what the transaction was for so you can review your transactions. To do this, the SHP system has a product string format which allows you to build up a list of products to send with the transaction. The name of the product string is the "secutiems" product string and it's built up using the below format:

```
[pd1|sku1|Product 1: size medium|10.00|2|20.00][pd2|sku2|Product 2: BOGOF|15.00|2|15.00]
```

Each product within the string is encased in square brackets, "[ ]". Within each product there are 6 fields separated by a single pipe symbol "|". The different fields that must be present within the product and the order in which they are assembled is:

1. Product code – String, optional, leave blank if not required.
2. SKU Code – String, optional, leave blank is not required.
3. Product description – String, required.
4. Unit price – Currency, required, format 0.00.
5. Quantity – Integer, required.
6. Line total – Currency, required, format 0.00.

Each product must have 6 fields, even if the first two fields are blank, if there are not 6 fields, the product data will be rejected by the SHP system. To supply multiple product simple append a new product, inside its own set of square brackets on the end of the existing products to build 1 long string which must be submitted in a field called "secuitems".

To display the products on your payment, confirmation or transaction error pages, you use some non-standard HTML tags unique to the SHP platform. You have a loop start tag and a loop end tag:

```
<loopstart:shoplst>
<!—Some HTML -->
<loopend>
```

For each product in the product string, our system will repeat the code inside the loop for each product allowing you to visually display each product the customer has purchased. Between the loopstart and loopend tags, there are also 6 additional fields which can be included into the HTML code, each of which will be replaced with relevant value from that product:

- $itemcode – Product code
- $itemskew – SKU code
- $itemdesc – Product description
- $itempric – Unit price
- $itemquan – Quantity
- $itemtota – Line total

Step 5: Securing Your Checkout

In order to secure your checkout from being modified, the SHP system has a feature called Advanced Secuitems Security.

What is the Advanced Secuitems Security System?

The Advanced Secuitems Security System allows merchants to secure their checkout against on-the-fly, inline modifications while the customer has been redirected to our site.

How Does It Work?

The way the security system works is be generating a unique hashed value using a combination of the fields you wish to secure and a secure pass phrase. This unique hash is then added to the redirect and payment forms which process the payment. If the customer attempts to modify one of the now protected fields, our system will identify this and prevent the customer from submitting their payment.

Requirements

As the integration for this system requires development to your shopping cart or other type of payment system, you will need to be a capable developer of the language your site/system is written in (e.g. Php, ASP.NET).

Advanced Secuitems Setup

Before you can start integrating the Advanced Secuitems Security System into your site, you first need to configure and activate it from within the client control Panel. **Please note:** do not activate the security system on a live site, to do so will prevent all customers from making a payment until your integration has been successfully completed. We advise performing this integration either before going live or on a test account.

The configuration settings for the Advanced Secuitems Security System are located within the Advanced Settings page of the client control panel. Below is a description of each of the setting:-

Activate advanced secuitems security

This tick box setting turns the security system on or off, check the box to turn it on.

List of fields to be encrypted

Our system cannot assume the fields it's being asked to protect, here you must define the list of fields by entering them in the form of a comma separated list (no spaces). You do not need to include the "secuitems" product string as this is included by default.

A phrase to be used to further encrypt your data sent through secuitems

In addition to the fields being protected, we also use a pass phrase, this is defined here. The phrase must be between 6 and 9 characters long.

The full URL referrer of your shopping cart

When generating the unique hash, your system makes a call to our system; this field is used by our system to ensure the call to our system comes from your system. You must define the referrer URL for the call from your system. This must include the protocol as part of the URL ("http://").

Generating the Unique Hash

Within the final stage(s) of your shopping cart/payment system you will need to make a call to the SHP platform in order to generate the unique hash to be added to the checkout. The call will be made to the script:

https://www.secure-server-hosting.com/secutran/create_secustring.php

The fields that need to be posted are:

- shreference – The account SH reference number
- secuitems – The SHP secuitems product string
- secuphrase – The additional phrase used generate the unique hash
- Any additional fields are added to the posted fields using the same field name as they would be submitted under.

The response body of the call to our system with either contain one of the error messages listed below or it will be formatted like this:

```
<input type="hidden" id="secuString" name="secuString"
value="9c84f49209fe9cdcb3efbac2dd2c23c8" />
```

This hidden form field is then to be added to your redirect form when redirecting the customer to your payment page on our server.

For some example code to call the SHP system to generate a unique hash, please see Appendix H.

Payment Page

The template driven payment page will need to have the following hidden form field element adding to its form. Please note, the below field names are case sensitive:

```
<input type="hidden" id="secuString" name="secuString" value="$secuString" />
```

Error Responses

Here is a table of the possible error responses that can be received when generating the unique hash and what they mean:

Error Message	What it means
SH Reference value not passed	The shreference has not been supplied
Secuitems value not passed	The secuitems product string has not been supplied
Advanced Secuitems not activated in Merchant Administration Area.	The Advanced Secuitems Security System has not been activated within the client control panel
Advanced Secuitems Secure Phrase does not match that set in Merchant Administration Area.	The additional phrase supplied with the request does not match the phrase configured within the client control panel
Referral Check Failed	The referrer of the request does not match the URL configured within the client control panel
Value for {field} set in account area but not posted.	A one of the fields configured within the client control panel is missing from the request
An error has occurred. Please contact Support	An error in the process of generating the unique hash has occurred

Step 6: Requesting Callbacks

What are callbacks?

Callbacks from the SHP platform are requests from our servers to your server(s) to notify your system that we've processed a transaction. These requests are independent of the customer's client browser session and do not involve a redirect. Typical uses of the callbacks would be to update an order stored within your shopping cart/payment system to mark it as complete and to potentially update your stock management.

Requirements

As the integration for this system requires development to your shopping cart or other type of payment system, you will need to be a capable developer of the language your site/system is written in (e.g. Php, ASP.NET).

Requesting a Callback

To request a callback from the SHP system, you only need to supply 2 additional fields with the transaction, "callbackurl" and "callbackdata". With these two fields, our system will be able to build up a query string of data and send the data to your payment system via a HTTP GET request.

The hidden field "callbackurl" that needs to be added to your payment form should contain the path to the script within your system where you want us to send the data. Please note: this path should contain the path to the script only, no query strings.

The hidden field "callbackdata" contains a name value pair string of data, each element of the string should be separated by a pipe symbol "|":

```
field1|value1|field2|value2|field3|value3
```

If you want to return data entered by the customer from within the payment form, you can use the "#fieldname" syntax:

```
cardholdersname|#cardholdersname|cardholdersemail|#cardholdersemail
```

If you are generating the value for the callback fields from your website and sending them to SHP when redirecting the customer, you will need to add the below hidden form fields to your payment form:

```
<input type="hidden" name="callbackurl" id="callbackurl" value="$callbackurl" />
<input type="hidden" name="callbackdata" id="callbackdata" value="$callbackdata" />
```

Receiving a Callback

When our system makes the callback to your system, it will typically happen around 2 minutes after the transaction has been processed. The time delay happens because callbacks are processed by an automated task within SHP rather than processing the callback as part of the transaction process. The reason for not processing the callback as part of the transaction process is to prevent the customer from waiting longer than necessary to see a confirmation page.

When the callback request calls your script, it will pass all the fields you've requested inside the callback data field along with a series of additional fields about the transaction. If the transaction is successful, you will receive the below fields:

- transactionnumber – Integer value, 8 digits long.
- transactiontime – DateTime format (yyyy-mm-dd hh:mm:ss)
- cv2avsresult – String
- upgcardtype – String
- upgauthcode – Alphanumeric

If the transaction has failed, you will expect to receive these fields:

- transactionnumber – Value of "-1" for a failed transaction
- transactiontime – DateTime format (yyyy-mm-dd hh:mm:ss)
- failurereason – String

The field "transactionnumber" is present in both successful and failed transactions, however you can use this field as an indication whether the transaction was successful or not. If the transaction has failed, the value will always be "-1", if the transaction is successful, the transaction number will be an 8 digit integer number.

When your script has completed all required actions, we advise outputting the work "success" or something similar as this response is captured by our servers and becomes visible from with the client control panel inside the transaction. This method of capturing the response from your script also allows you to output error messages should something go wrong in your script.

When your script has completed, you need to ensure your web server outputs the standard HTTP 200 response code as our system uses the response code to identify the a callback has been successful or not. If we receive a HTTP 200 response, we will consider the callback successful. If we do not receive a HTTP 200 response we will consider the callback as failed and will keep trying the callback repeatedly for up to 24 hours after the transaction. If we have not received a successful HTTP 200 response after 24 hours, we will stop attempting the callback all together. If we receive a HTTP 301 or 302 response (redirect), our system will not follow the redirect, we will also not retry the callback.

Additional Fields

There are a certain number of additional fields that can be requested with the callback. The following fields are requested by adding the extra fields to the callback data field:

- IP Country – "IPCountry|#IPCountry"
- Bin Country – "BinCounry|#BinCountry"
- Bin Bank – "BinBank|#BinBank"

The below fields can be requested by activating the relevant configuration settings within the Advanced Settings page of the client control panel:

- BIN number (first 6 digits of the card number) – The BIN number can be returned in the callback by specifying a field name within the client control panel. If a field name has not been specified we will not return the BIN number.
- Encrypted PAN – A one way encrypted version of the card number can be returned in the callback by entering a field name within the client control panel. If a field name has not been specified, we will not return the encrypted card number. The encrypted card number can also be optionally salted by entering a salt value.
- 3-D Secure Response – If you're account is using 3-D Secure, you may wish to know the 3-D Secure response returned by the card scheme and card issuing bank. By ticking the box in the client control panel, our system will add the enrolment value returned by the card scheme to the callback inside the field "enrolmentresponse". If the enrolment response was a "Y", you will also receive an additional field called "authenticationresponse" with the authentication response returned by the card issuing bank.

Verifying the Callback

If you wish to add additional security to ensure the callback being received is from the SHP platform, there are two checks that can be implemented. The first check is that the referrer of the call is https://www.secure-server-hosting.com/secutran/ProcessCallbacks.php.

A more advanced method of verifying the callback is to use the shared secret system. The shared secret system works by adding a hashed value to the callback which can only be created by knowing how the field was created along with having the required values stored at your end. To activate the shared secret validation, you just enter a shared secret value into the Advanced Settings page of the client control panel. Our system will use the value provided to salt the start and end of the transaction reference number, we will then SHA-1 encrypt the resulting string and add the encrypted value to the callback inside a field called "verify".

Step 7: Continuous Authority

Continuous Authority (CA), also referred to as recurring transactions, allows the merchant to automatically bill customers' cards on a regular basis for a fixed amount and for a predefined period. Typical applications would be for magazine subscriptions, web hosting payments or membership websites.

What are the limitations?

- You will need to set up a special continuous authority merchant account with your bank in order to be able to take recurring payments. We can assist with this if you have any problems.
- It is not possible to use Maestro cards for recurring payments. Check with your bank which cards are compatible with continuous authority.
- The value of the recurring payment is normally a fixed amount.
- The maximum period between two payments cannot be longer than 12 months.
- You can only process a single subscription at a time.
- Your SecureHosting account must be configured for online card processing using our Monek payment gateway.
- Please contact us to enable the continuous authority feature for your SecureHosting account.

Overview of Continuous Authority

With the SecureHosting CA feature you can offer subscribers a trial period, special introductory rates, and a regular standard rate. Subscribers are billed automatically according to the terms you specify.

Adding a new subscriber

There are four methods of adding a new subscriber:-

1. By posting a transaction from your website, referring to a predefined subscription option.
2. By posting a transaction from your website, creating an “On-the-fly” subscription option, created at the point of a transaction.
3. Convert an existing customer into a subscriber.
4. Manually enter a new subscriber.

Details of fields/form actions required when posting a new subscription sign-up

The following fields are used whether the subscriber is to be set up using a predefined subscription option or “on-the-fly”.

Form action:

```
<form action="https://www.secure-server-hosting.com/secutran/transactionsb1.php"
method="POST">
```

Required Fields:

- shreference
- checkcode
- transactionamount
- transactioncurrency
- cardnumber
- cardexpirymonth
- cardexpiryyear

Optional Fields:

- transactiontax
- shipingcharge
- cardstartmonth
- cardstartyear
- cardholdersname
- cardholdersemail
- cardholderaddr1
- cardholderaddr2
- cardholdercity
- cardholderstate
- cardholderpostcode
- cardholdercountry
- cardholdertelephonenumber

Testing phase

When placing test transactions, use the form action below instead:-

```
<form action="https://test.secure-server-hosting.com/secutran/transactionsb1.php"
method="POST">
```

Enter a new subscriber using one of the methods below.

In order to test a full subscription right through to its natural expiry, click the “Schedule Now” button when viewing the subscriber payment detail in your account. This option will only appear if you have entered subscribers via the test URL.

Method 1: Adding a subscriber using a predefined subscription

A subscription template will fist need to be configured in your account before you can set up subscribers using predefined subscriptions.

You will see a menu option in your account named “Manage predefined subscriptions” within the “Settings” tab. Click on this to view a list of any existing subscription templates.

Setting	Description	Example Value
Subscription reference	This is the unique reference for the subscription template	monthlysub
Subscription name	This is the name of the subscription which will be displayed in the customer and merchant emails, transaction confirmation page etc.	Monthly Subscription
Subscription description	The description to be displayed in the customer and merchant emails, transaction confirmation page etc.	12 months' access to our exclusive journal.
Subscription period	The time period between recurring payments. Must be consistent within a single subscription package, ie. It is not possible to mix months with days in one subscription option.	Year, months, weeks, days.
Introductory rate 1	Initial trial period cost. Leave blank if not required, or enter “0” for free trial.	
Introductory rate 2	As above, but allows a secondary trial rate to follow on from the first. If used, introductory rate 1 must also contain values. Leave all fields blank if not required.	

Setting	Description	Example Value
Standard rate	Enter the amount for this subscription at full, normal rate. In the “To be billed every” box enter the period length, i.e. The period between recurring transactions e.g. 1 month. In the “for: payments” box enter the subscription cycle value e.g. 12 (the subscription is for 12 months)	20.00
Number of cycles at standard rate	The life of the subscription. This subscription will automatically end after 12 months. To make a subscription run indefinitely enter “0” in this field or leave blank.	1
Should failed payments be retried?	Option to define whether payments are retried following failure.	
Contact Note	The merchant's contact details and address etc. which will be displayed in the customer login area where they can view and manage their subscription.	Monek Limited, City House, Davidson Road, Lichfield, WS14 9DZ. Tel: 0345 269 6645
Subscription Currency	The currency for all payments for this subscription option. Please note that you must have multi-currency merchant numbers in order to use currencies other than GBP.	GBP
Should customers be able to cancel their own subscription?	Allows the customer control to cancel their subscription prior to normal expiry.	
What is the minimum subscription period	This is only relevant if above option is enabled. It provides an override for the above option, in that you can set a minimum period (in days) during which cancellation by the customer is not possible.	180

Setting	Description	Example Value
From address	The email address that the confirmation email is to appear to be from.	sales@mycompany.com
Reply-to address	The email address to which emails will be sent if the customer replies.	support@mycompany.com
Customer emails	A series of email templates covering various scenarios. If not required leave email content blank.	
Customer sign-up	Email sent to the customer following sign-up.	
Customer details updated	Email sent if customer changes their credit card details.	
Customer scheduled payment notification	Email to warn the customer of an impending payment.	
Customer successful payment notification	Email sent following successful payment	
Customer payment failed notification	Email sent if payment fails.	
Customer payment cancelled notification	Confirmation of the customer's cancellation.	
Merchant emails	Corresponding emails sent to you following each scenario.	

Placing a transaction with a predefined subscription option

When placing the transaction, you will need to add a field to the form called "subscription", the value of which needs to be the value of the unique subscription reference configured against the subscription template. Please note, you cannot set up a subscription using the "default" subscription template as the subscription template settings are reserved for "on-the-fly" subscriptions.

Method 2: Adding a subscriber using "on-the-fly" subscriptions

To post an on-the-fly subscription sign-up there are several additional fields for which values will need to be presented with the transaction:-

Field Name	Description	Example Data
subscription	This is the subscription code and description that will be displayed to the customer for the subscription.	monthlysub|Monthly Magazine Subscription
shrecur_trial1	This would create a 3 month free trial for the subscription, occurring only once.	3|M|1|0 (period length|period type|period cycle|period price)
shrecur_trial2	This would allow the fourth month of the subscription to be given at a reduced rate of £10.	1|M|1|10.00
shrecur_rate	This is the full subscription after the trial period. In this case a monthly subscription recurring for the remainder of the 12 months subscription period. This subscription will terminate after 12 months.	1|M|12|20.00
sh_recur	Life of subscription. This subscription will automatically end after 12 months. To make a subscription run indefinitely enter "0" in this field.	1
sh_retry	This is the length of time to wait before retrying a failed subscription payment. *(In number of days).* If no retry is required, enter "0" in this field.	5

Notes:

1. Available period types are year (Y), month (M), week (W), day (D)
2. All above fields are compulsory except for "trial1" and "trial2".

Method 3: Converting an existing customer into a subscriber

Within your account, go to view an existing customer's payment details. Click the “Create subscription from this transaction” button. A page will be displayed showing any predefined subscription options, or you can create a new subscription option for this subscriber.

Method 4: Manually enter a new subscriber

Within your account, go to “Place Manual Transaction” within the “Transactions” menu option. Select a predefined subscription and then a date for the first payment to be taken.

Callbacks

Callbacks are supported through the recurring payments system when subscribers are set up from a posted transaction. For information on requesting callbacks, please see Step 6: Requesting Callbacks.

When a callback is triggered from our system, you will receive all the standard callback fields, you will also receive 2 additional fields:

- subscription – The subscription reference
- sh_reason – The reason for the callback

The different possible reasons for subscription callbacks are:

- success
- payment_failed

Step 8: PayPal Support

Setting Up PayPal

We have prepared detailed instructions to help you integrate your PayPal Merchant Account with your SHP eCom package.

These instructions are clear and straightforward, however, there are a number of important steps which you must follow so that the service runs smoothly so please take care and ensure you follow all steps, as outlined:

1. Create a PayPal account
2. Confirm your email address with PayPal
3. Verify your PayPal account
4. Request API credentials and create an API signature
5. Request PayPal functionality is turned on in your account
6. Logon to http://www.securehosting.com/
7. In the user log in area enter your client login and password
8. From the home page select "Online Processing > PayPal Settings"
9. Fill in the details supplied to you by PayPal:
    a. PayPal Username
    b. PayPal Password
    c. PayPal API Key
10. Make PayPal Active on your account by selecting "Yes" (you can use this option in the future should you wish to disable the service at any time).
11. Press "Submit above details"

Configure Your Website

To configure your website to be able to process PayPal transactions, having completed steps 1 and 2 of the Hosted Form integration, all you need to do is change the redirect form action from:

```
<form action="https://www.secure-server-hosting.com/secutran/secuitems.php" method="post">
```

To:

```
<form action="https://www.secure-server-hosting.com/secutran/paypal.php" method="post">
```

You will also need to ensure your redirect form contains the following fields on to of any other fields you wish to send:

- shreference
- checkcode
- transactionamount
- transactioncurrency

The value of the transaction currency must be the ISO 4217 Alpha-3 currency code for the desired currency. Please see Appendix L for a list of currencies supported by PayPal.

Additional considerations

Some additional consideration must be made when integrating PayPal, if you wish to submit product data to PayPal, you must ensure that product totals calculated from the product string, and the shipping charge (if supplied) add up to the transaction total. If these values do not add up, PayPal will not allow you to create your checkout. If you do not send product data with your order, the order will still be processed.

For more information on the product string, please see Step 4: Building Product Data.

Customising the New PayPal Templates

With the PayPal supported Hosted Form solution, when the customer checks out through PayPal the customer journey is a little different. After being redirected from your website, the customer is presented with the option of paying with PayPal or a normal card transaction, this option page is the first customisable page on the SHP website. The customer selects PayPal and goes to the PayPal website, when they come back to the SHP website and are then asked to click a button to confirm the payment. The confirm page is the second customisable page on the SHP website.

The two pages can be customised by uploading correctly named HTML templates to your SHP account via the File Manager interface.

The two HTML template file names are:

1. paypal_option.html
2. paypal_confirm.html

You can download copies of our default templates by clicking on the file names above.

In each of the two templates, you must include the text “$paypalbutton” in order for the SHP System to insert a form which will handle the process.

In the PayPal option page, you can use the below code to build a form to give the customer the option to choose your normal checkout method:

```
<form action="https://www.secure-server-hosting.com/secutran/secuitems.php" method="post">
    $postedfields
    <input type="submit" value="Credit Card Payment" />
</form>
```

Alternatively you can actually build your payment form straight into the PayPal option page.

Once the PayPal transaction has been completed by the customer, the SHP system will either use the same confirmation page as your normal card transaction method, or if you have uploaded the following PayPal specific templates, the system will use these templates instead:

1. paypal_good.html
2. paypal_bad.html

The functionality of these templates is the same as the normal confirmation and error pages, the existence of these templates simply allows you have PayPal specific confirmation or error templates.

Additional Data

In order to supply additional checkout information with a PayPal transaction, you must include the additional fields in the redirect form. You can supply the customer name and address details inside the various cardholder fields associated with a card transaction, the SHP system will capture the address supplied by your site as the customer's address but it will also capture all information, including the customer information supplied by PayPal. You will be able to see all information about the transaction within the client control panel.

You can also use callbacks with PayPal for more information please see Step 6.

The API integration process in Detail

The API integration works by allowing you to keep the customer on your system through the checkout process while processing the transactions with SHP in the background. This allows you to provide a smoother, more complete checkout process to the customer. In addition to transaction processing, you can also perform other actions with the API like refunds which can provide a more advanced integration with SHP.

Requirements

In order to build your API integration you will need knowledge in the structure and language that your website or payment system is written in. You and your website/payment will need access to the internet in order to submit the transactions. You will also need to consider the Payment Card Industry Data Security Standard (PCI:DSS) when your system captures card details. For more information, please see https://www.pcisecuritystandards.org/.

Step 1: Build your Request XML

The first thing you will need to do when building your API integration is to build the request XML string. In order to allow future flexibility of your code, we advise you code the building of the request XML string in its own function/method. Below is an example request XML string for a transaction:

```
<?xml version="1.0"?>
<request>
    <type>transaction</type>
    <authtype>authorise</authtype>
    <authentication>
        <shreference>SH200000</shreference>
        <checkcode>000000</checkcode>
    </authentication>
    <transaction>
        <cardnumber>4242424242424242</cardnumber>
        <cardstartmonth>01</cardstartmonth>
        <cardstartyear>10</cardstartyear>
        <cardexpiremonth>01</cardexpiremonth>
        <cardexpireyear>12</cardexpireyear>
        <cv2>424</cv2>
        <cardholdersname>Joe Bloggs</cardholdersname>
        <cardholdersemail>joe@bloggs.com</cardholdersemail>
        <cardholderaddr1>Address 1</cardholderaddr1>
        <cardholdercity>City</cardholdercity>
        <cardholderstate>State</cardholderstate>
        <cardholderpostcode>Postcode</cardholderpostcode>
        <transactioncurrency>GBP</transactioncurrency>
        <transactionamount>29.00</transactionamount>
        <transactiontax>4.50</transactiontax>
        <shippingcharge>5.00</shippingcharge>
        <secuitems><![CDATA[
                    [pd1|sku1|Product 1|10.00|2|20.00]
                ]]></secuitems>
    </transaction>
</request>
```

The “type” node identifies the type of request we are performing, in this instance we are doing a transaction. The “authtype” tells our system whether to perform a fully authorised (“authorise”) or pre-authorised (“pre_auth”) transaction. The “authtype” node is optional, if it’s not supplied, the SHP system will use the default account setting. For more information on Pre-Authorisation, please see the Processing A Transaction section of the SHP Users Guide.

The “authentication” node holds the required data to identify the account to process the request with. All transaction fields are placed inside a “transaction” node. Similar to the hosted form integration, the below nodes are required to perform a transaction:

- transactionamount
- transactioncurrency
- cardnumber
- cardexpiremonth (2 digit month)
- cardexpireyear (2 digit year)
- cv2 (security code on reverse of the card)

The below nodes are optional but we recommend supplying them with every transaction:

- transactiontax
- shippingcharge
- cardholdersname
- cardholdersemail
- cardholderaddr1
- cardholderaddr2
- cardholdercity
- cardholderstate
- cardholderpostcode
- cardholdercountry
- cardholdertelephonenumber

If you wish to submit any additional data with a transaction, you just create additional child nodes to the transaction node. The node name you give the field is the field name our system will use to store the data.

Step 2: Submitting the Request XML

Having build the request XML string, you now need to submit it to the SHP API interface for processing. The string is submitted to the API interface as the value of a POST header field called “xmldoc”. This POST field is then posted to our API interface via an inline HTTP call like cURL or ActiveXObject("MSXML2.ServerXMLHTTP.6.0 "). The response body of the call will be the response XML. For example code, please see Appendix I.

Step 3: Parsing the Response

Once you’ve submitted your request XML to the SHP API interface, the response XML is picked up from the response body. Below is an example of the response XML for a successful transaction:

```
<?xml version="1.0"?>
<result>
    <status>OK</status>
    <reference>1234567</reference>
    <transactiontime>23:59:59</transactiontime>
    <transactiondate>2005-06-10</transactiondate>
    <cv2avsresult>ALL MATCH</cv2avsresult>
    <cardtype>Visa Debit</cardtype>
</result>
```

The “status” node will have the value of either “OK” or “ERROR”, it the status is OK you will received additional node with transaction information. If the status is error there will be an additional node with an error type value called “statustext”. The different possible error types are:

- DATABASE_ERROR – There has been a problem talking to our database. If you see this issue, please contact our support department.
- INVALID_LOGIN– The authentication credentials supplied are invalid.
- NO_XML_PASSED – No XML as been supplied inside the post header field “xmldoc”.
- BAD_XML_PASSED – The XML supplied is improperly formatted and cannot be parsed.
- DATA_ERROR – There is a problem with the request data supplied.
- CREDIT_ERROR – The account does not have enough transaction credits to perform the request.
- UPG_ERROR – There’s been an error talking to our gateway system.
- TRANSACTION_ERROR – There is a transaction problem, an additional node called “reason” will contain a textual description of the error.

Here is an example of a failed transaction response:

```
<?xml version="1.0"?>
<result>
    <status>ERROR</status>
    <statustext>TRANSACTION_ERROR</statustext>
    <reason>Card Declined</reason>
</result>
```

Below is a PHP code example to parse the response XML:

```
$response = simplexml_load_string( $xmlResponse );
if( (string)$response->status == 'OK' ){
    $transactionId = (int)$response->reference;
} else {
    if( (string)$response->statustext == 'TRANSACTION_ERROR' ){
        $errorMessage = (string)$response->reason;
    } else {
        $errorMessage = (string)$response->statustext;
    }
}
```

Step 4: Refunds

Along with processing transactions, you can also perform refunds against transaction via the API interface. Below is an example of a refund request XML string:

```
<?xml version="1.0"?>
<request>
    <type>refund</type>
    <authentication>
        <shreference>SH200000</shreference>
        <password>password</password>
    </authentication>
    <transaction>
        <reference>1374617</reference>
        <amount>34.99</amount>
    </transaction>
</request>
```

The value of the “type” node is “refund”, within the authentication node, we still have the account SH reference value, however instead of the checkcode, we require the account password inside a node called “password”. Inside the transaction node, we only need to reference of the transaction you are refunding along with the amount you wish to refund. As refunds are only processed against transactions, if any card details are supplied, they will be ignored. As the refund request XML does not contain any sensitive card details, you can actually implement refunds through the API interface without bringing your website or payment system into scope of PCI:DSS.

The response XML string, if successful will only contain a status node with a value of “OK”:

```
<?xml version="1.0"?>
<result>
    <status>OK</status>
</result>
```

The response on error is the same as the standard transaction error response.

Step 5: Authorisations

This step is only relevant to merchant who pre-authorise their transactions, for more information on pre-authorisation, please see the Processing Payments section of the SHP User Guide.

When a transaction has been pre-authorised, it will not be settled until the SHP system received a request to settle the transaction, the API authorisation call is that request through the API. In order to authorise a pre-authorised transaction, you will need to send the API the below XML string:

```
<?xml version="1.0"?>
<request>
    <type>authorisation</type>
    <authentication>
        <shreference>SH200000</shreference>
        <password>password</password>
    </authentication>
    <transaction>
        <reference>12345678</reference>
        <amount>10.00</amount>
    </transaction>
</request>
```

The value of the “type” node is “authorisation”, within the authentication node, we still have the account SH reference value, however instead of the checkcode, we require the account password inside a node called “password”. Inside the transaction node, we only need to reference of the transaction you are authorising along with the amount you wish to settle. As authorisations are only processed against transactions, if any card details are supplied, they will be ignored. As the authorisation request XML does not contain any sensitive card details, you can actually implement authorisations through the API interface without bringing your website or payment system into scope of PCI:DSS.

The response XML string, if successful will only contain a status node with a value of “OK”:

```
<?xml version="1.0"?>
<result>
    <status>OK</status>
</result>
```

The response on error is the same as the standard transaction error response.

Step 6: Additional Transactions

An “Additional Transaction” is a new transaction processed with the card details from a previous transaction. This type of transaction allows returning customers to use the same card they used on their last transaction. The way this works is by supplying a reference to the last transaction instead of the card details. If you do not supply values for the card details in the various nodes, we will use the details from the previous transaction. If you do supply a value, we will use the supplied value rather than the value from the last transaction.

The only exception to this is the security code (CVV or CV2) on the reverse of the card. Due to PCI:DSS it cannot be stored by any party but you can supply it in the request XML string, if you are able to obtain it from the cardholder anew. It is not mandatory to supply the CV2 number with an additional transaction but recommended to achieve higher authorisation rates and benefit from lower bank fees. Below is an example of an additional transaction request XML string:

```
<?xml version="1.0"?>
<request>
    <type>additional</type>
    <authentication>
        <shreference>SH200000</shreference>
        <checkcode>000000</checkcode>
    </authentication>
    <transaction>
        <reference>12345678</reference>
        <currency>GBP</currency>
        <transactionamount>29.00</transactionamount>
        <transactiontax>4.50</transactiontax>
        <shippingcharge>5.00</shippingcharge>
        <secuitems><![CDATA[
            [pd1|sku1|Product 1|10.00|2|20.00]
        ]]></secuitems>
        <cv2>123</cv2>
    </transaction>
</request>
```

Responses are the same as normal transactions.

Step 7: Mail Order Telephone Order Transactions

If you wish to build your own virtual terminal system, you will need to ensure when processing your transactions, that your bank understands the transaction is a MOTO (Mail Order Telephone Order) transaction. The reason for this is because your bank will have different requirements in order to classify a transaction as secure, e.g. 3-D Secure is often required for internet transactions, but impossible for MOTO transactions. Also, if your account has been configured to process your MOTO transactions to a different MID from your eCommerce transactions, we will need to know it's a MOTO transaction.

To identify a transaction as MOTO, you send us a normal transaction XML string; however, the value of the "type" node will be "moto":

```
<?xml version="1.0"?>
<request>
    <type>moto</type>
    <authtype>authorise</authtype>
    <authentication>
        <shreference>SH200000</shreference>
        <checkcode>000000</checkcode>
    </authentication>
    <transaction>
        <cardnumber>4242424242424242</cardnumber>
        <cardstartmonth>01</cardstartmonth>
        <cardstartyear>10</cardstartyear>
        <cardexpiremonth>01</cardexpiremonth>
        <cardexpireyear>12</cardexpireyear>
        <cv2>424</cv2>
        <cardholdersname>Joe Bloggs</cardholdersname>
        <cardholdersemail>joe@bloggs.com</cardholdersemail>
        <cardholderaddr1>Address 1</cardholderaddr1>
        <cardholdercity>City</cardholdercity>
        <cardholderstate>State</cardholderstate>
        <cardholderpostcode>Postcode</cardholderpostcode>
        <transactioncurrency>GBP</transactioncurrency>
        <transactionamount>29.00</transactionamount>
        <transactiontax>4.50</transactiontax>
        <shippingcharge>5.00</shippingcharge>
        <secuitems><![CDATA[
                    [pd1|sku1|Product 1|10.00|2|20.00]
                ]]></secuitems>
    </transaction>
</request>
```

Step 8: Continuous Authority

Continuous authority is supported through the API; however, it does not use the advanced functionality of the automated recurring payments feature. The continuous authority API is for users who wish to take full control for their subscription transactions, the CA API system offers you the ability to process CA payments however it is the responsibility of the merchant to regulate the transaction values and frequencies. Please be aware as a rule of thumb the banks expect CA payment to be a predictable transaction amount on a regular or predictable frequency, any deviation from this can be viewed as an abuse of the merchant's CA acquiring account. You must also only ever process a CA transaction on a card provided you have obtained full authorisation and authentication against that card via your normal merchant account, the CA API system covered this by only allowing you to process a manual CA transaction when we have records of a successful authorised transaction to take the card details from.

Below is an example of a request XML string:

```
<?xml version="1.0"?>
    <request>
        <type>manual_ca</type>
        <authentication>
            <shreference>SH200002</shreference>
            <checkcode>607666</checkcode>
        </authentication>
        <transaction>
            <ca_trannum>10000003</ca_trannum>
            <last_trannum>12345678</last_trannum>
            <amount>10.00</amount>
        </transaction>
    </request>
```

The required nodes are the "shreference" and "checkcode" to identify the account and the "amount" to know how much to process. Either the "ca_trannum" or the "last_trannum" must be supplied but they do not both need to be supplied together. The "ca_trannum" is for an existing CA reference if one exists, if a CA reference does not yet exist and this is the first CA payment to the card then send us the transaction reference of the previous transaction in the "last_trannum" node. We will then create a CA reference to be stored and used going forward.

The below XML is returned on success:

```
<?xml version="1.0"?>
    <result>
        <status>OK</status>
        <statustext>Reference: 12345678</statustext>
        <reference>12345678</reference>
        <ca_trannum>10000003</ca_trannum>
        <transactiontime>23:59:59</transactiontime>
        <transactiondate>2005-06-10</transactiondate>
    </result>
```

The result on failure is the same as a normal transaction apart from the below additional errors:

- MANUAL_CA_DISALLOWED – The manual CA system must be turned on for individual accounts, please contact support.
- NO_AMOUNT – No amount has been passed in the XML
- INVALID_REFERENCE – No valid transaction can be obtained from the “ca_trannum” or the “last_trannum”.
- SUBSCRIPTION_ERROR – There has been an error processing the subscription, please contact support.

Step 9: Transaction Downloads

This feature is provided as an additional reporting tool to advanced integrations into SHP. The feature provides you with the ability to download your transactions in XML format allowing you to parse the data and import it into any custom platform within your control. We advise the following requirements for this feature:

- eCom Standard or Premium account
- One-off set up fee of £50 (ex VAT)
- Good understanding of one or more programming language
- An understanding of object orientated programming

How does it work?

The transaction data is capture from SHP by calling a URL within your application and reading the response body. Within the URL you call you must pass a series of fields using query string format, the response body will be XML string for your application to parse.

Base URL

The base URL (before the query string) is:

https://www.secure-server-hosting.com/secutran/transaction_xml.php.

Required Fields

Field Name	Description	Example
shref	SHP account number	SH212345
checkcode	Second level security check code	123456
actpasswd	SHP account password	Password01
transcsv	CSV file setting	1, 2 or 3

Transaction range fields

A transaction range must be supplied in order to return your transaction; you can specify the range either by date or by transaction ID

Field Name	Description	Example
transfrom	Transaction ID range, lower value	12345678
transto	Transaction ID range, upper value	87654321
datefrom	Transaction date range, lower value	2011-01-01
dateto	Transaction date range, upper value	2011-01-31

Optional Fields

In addition to the require fields you can also send in some additional fields to vary the transactions being returned.

Field Name	Description	Example
page	Page listing	1
tranresult	Successful or failed transactions	0 (failed) or 1 (successful)

The response

The response you will receive back will always be in XML format and it will always be the within response body. If the request is successful then the XML root node will be called “transactions”, however if there has been an error then the root node will be “error”.

Response on success

Below is an example response for a successful request:

```
<?xml version="1.0"?>
    <transactions>
        <fieldstofind>
            <field name="transactiontime"/>
            <field name="transactiondate"/>
            <field name="transactionamount"/>
            <field name="transactioncurrency"/>
            <field name="cardholdersname"/>
            <field name="cardholdersaddr1"/>
            <field name="cardholderpostcode"/>
        </fieldstofind>
        <data>
            <transaction>
                <fields>
                    <transactionid>13374878</transactionid>
                    <transactiondate>2011-05-12</transactiondate>
                    <transactiontime>16:07:10</transactiontime>
                    <transactionamount>5.00</transactionamount>
                    <transactioncurrency>GBP</transactioncurrency>
                    <cardholdersname>Test Tran</cardholdersname>
                    <cardholderpostcode>B77 5DQ</cardholderpostcode>
                </fields>
                <orderitems>
                    <item>
                        <description>87654321</description>
                        <price>5.00</price>
                        <qty>1</qty>
                        <totalprice>5.00</totalprice>
                    </item>
                </orderitems>
            </transaction>
        </data>
    </transactions>
```

The path" \transactions\fieldstofind" will contain a child node called "field" for every field being requested, the name of the field will be found in the attribute "name". Each transaction being returned will be under the path "\transactions\data\transaction", with a "transaction" node for each individual transaction.

The response will only ever contain 50 transactions, if there are more than 50 within the transaction range then the download feature will use the page listing feature to split the response into pages of 50, you use the field "page" in the request to return different pages.

If the current page you are requesting is not the last page in the range, an additional node called "nexttransaction" will be available as a child of "transactions". If the current page is the last in the range, the "nexttransaction" node will not be returned.

Response on error

Below is an example of an error response:

```
<?xml version="1.0"?>
    <error>missing or invalid sh reference</error>
```

The different error messages that can be returned are:

- missing or invalid sh reference
- missing or invalid check code
- no account password
- invalid sh reference/password combination
- Csv Download Not Authorised
- no csv file type supplied
- invalid transaction range
- no transaction range

Step 10: 3-D Secure API

What is the 3-D Secure Transaction API

The 3-D Secure API is a method of processing 3-D Secure transactions with SHP while capturing the card details within your website.

How Does It Work?

The way the 3-D Secure API works is your payment system will capture all of the customer's card details in your website then send the details directly from your system to our API interface. The API will then return a HTML page for your system to output to the client browser. This HTML page will redirect the customer to their card issuing bank's page via our platform. When the customer is redirected back to our system, we will perform the transaction with the merchant acquiring bank before instantly redirecting them back to your website.

The reason we have to perform a redirect, unlike the standard transaction API, the nature of 3-D Secure requires the customer to be redirected to their card issuing bank's page for authentication. In order for the SHP system to support this redirect and the transaction, the customer must be redirected to the SHP system between going to and from the card issuing bank's page. The SHP system will attempt to make these redirects appear instant in order to maintain the perspective that the customer's transaction is being processed by your site.

Requirements

As your website will be capturing and handling card details, it will be in scope for PCI:DSS. For information on your website's requirements for PCI:DSS we advise you review the PCI Security Standards website (https://www.pcisecuritystandards.org/) or contact your Qualified Security Assessor.

Submitting a Transaction

To submit a transaction, you will need to build up the transaction request data, this data is an XML formatted string which contains all the transaction data. The XML string will be used for the value of the POST element "xmldoc".

The XML doc field along with an additional field called "script_location" are then posted to the API interface https://www.secure-server-hosting.com/secutran/3dsecure_api.php. The value of the field "script_location" will be the URL of where we will need to redirect the customer back to after the transaction is complete.

Request Data

Below is an example of an XML request for a standard eCommerce transaction:

```
<?xml version="1.0"?>
<request>
        <type>transaction</type>
        <authtype>authorise</authtype>
        <authentication>
                <shreference>SH2XXXXX</shreference>
                <checkcode>XXXXXX</checkcode>
        </authentication>
        <transaction>
                <cardnumber>4242424242424242</cardnumber>
                <cardstartmonth>01</cardstartmonth>
                <cardstartyear>04</cardstartyear>
                <cardexpiremonth>01</cardexpiremonth>
                <cardexpireyear>06</cardexpireyear>
                <cv2>424</cv2>
                <cardholdersname>SecureHosting</cardholdersname>
                <cardholdersemail>blackhole@upg.co.uk</cardholdersemail>
                <cardholderaddr1>Address 1</cardholderaddr1>
                <cardholdercity>City</cardholdercity>
                <cardholderstate>State</cardholderstate>
                <cardholderpostcode>Postcode</cardholderpostcode>
                <currency>GBP</currency>
                <transactionamount>34.99</transactionamount>
                <transactiontax>3.50</transactiontax>
                <shippingcharge>1.00</shippingcharge>
                <secuitems><![CDATA[[1129||Laptop|350.00|1|350.00]]]></secuitems>
        </transaction>
</request>
```

When the customer is returned to your site following a successful transaction, the below data is returned inside the POST header as form fields:

- status – “OK”
- reference – 8 digit Integer
- transactiontime – {00:00:00}
- transactiondate – {0000-00-00}
- cardtype – String, varying length
- cv2avsresult – String, varying length
- threedsresult – String, varying length
- threedsenrolment – Sting, 1 character
- threedsautheitcation – String, 1 character
- xmlresult – XML string, similar formatted string as the response for the standard API

The below data is returned upon a failed transaction:

- status – String, varying length
- statustext – String, varying length
- threedsresult – String, varying length
- threedsenrolment – String, 1 character
- threedsautheitcation – String, 1 character

- xmlresult – XML string, similar formatted string as the response for the standard API

The 3-D Secure API also supports the "additional" transaction type. Please see the standard transaction API Step 6 of the API Integration Process for more details.

Enrolment Check

As the 3-D Secure API involved redirecting the customer, you can perform a call to the Secure Hosting platform to check the enrolment status of a customer's card before performing the transaction.

This check provides you with the option to check the enrolment status of a card before performing a 3-D Secure transaction, if you decide you do not wish to perform a 3-D Secure transaction you can fall back to the standard API.

This call is made in a similar fashion to a 3D secure API transaction. An XML string is posted to the 3-D Secure API for the enrolment check, the response is XML.

Below is an example of an enrolment check XML string:

```
<?xml version="1.0"?>
<request>
        <type>3dscheckenrolled</type>
        <authentication>
                <shreference>SH2XXXXX</shreference>
                <checkcode>XXXXXX</checkcode>
        </authentication>
        <transaction>
                <cardnumber>4242424242424242</cardnumber>
                <cardstartmonth>01</cardstartmonth>
                <cardstartyear>04</cardstartyear>
                <cardexpiremonth>01</cardexpiremonth>
                <cardexpireyear>06</cardexpireyear>
                <currency>GBP</currency>
                <transactionamount>34.99</transactionamount>
        </transaction>
</request>
```

The response XML is formatted as below:

```
<?xml version="1.0"?>
<result>
        <status>OK</status>
        <enrolled>Y</enrolled>
</result>
```

The different possible values for the enrolled status are:

- Y – The card is enrolled into the scheme
- N – The card is not yet enrolled by the card issuing bank into the scheme
- U – We have been unable to determine the enrolment status of the card

The below XML will be returned if there is an error performing the enrolment check:

```
<?xml version="1.0"?>
<result>
        <status>ERROR</status>
        <statustext>ERROR_DESCRIPTION</statustext>
</result>
```

The different possible error descriptions that can be returned from the enrolment call are:

- 3DSECURE_CALL_FAILED – Problem calling the 3-D Secure service
- 3DSECURE_ERROR – An error was returned from the 3-D Secure check, a further node called reason will contain more details
- 3DS_NON_MAESTRO – This is an error response which will only be sent for a non-Maestro card when 3-D Secure is set to Maestro only within your account

For a code example for processing a transaction with the 3-D Secure API, please see Appendix J.

Appendices

Appendix A: Support Acquirers

The SHP platform can currently process payments through the below acquirers:

Acquirer	Type
Barclays Merchant Services	Acquiring Bank
Lloyds TSB Cardnet	Acquiring Bank
First Data Merchant Solutions	Acquiring Bank
HSBC	Acquiring Bank
Elavon	Acquiring Bank
Streamline (RBS WorldPay)	Acquiring Bank
American Express	Acquiring Bank
Diners Club	Acquiring Bank
Voice Commerce/CashFlows	Acquiring Bank
Valitor	Acquiring Bank
PayPal	E-Wallet

Appendix B: Transaction Scripts

Below is a list of different scripts within the SHP platform which support card transactions, details of what they are for and which features they support. All scripts are located within the directory https://www.secure-server-hosting.com/secutran/:

Script Name	Purpose	3-D Secure (VbV/SecureCode)	Product Data
transactionac1.php	Actinic eCom	Yes	No
transactionad1.php	Additional eCom	Yes	Yes
transactionjs1.php	eCommerce	Yes	No
transactionmo1.php	MOTO	No	Yes
transactionpp1.php	PayPal	No	Yes
transactionsb1.php	Recurring (CA)	Yes	Yes
transactionsi1.php	eCommerce	Yes	Yes

Appendix C: Test Card Numbers

Your test account will automatically send transactions to our test server. If you wish to test using a live account, you may do so by re-directing the transaction call to 'test.secure-server-hosting.com'.

Below is a list of test card numbers to be used against our testing domain. Our testing platform is strictly for testing purposes. You must not send live card details or live tokens to our test service; using a live token within the test environment will result in a successful transaction/refund against the card.

Visa Credit

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK1	4015 5011 5000 0216	12/yy*	276	Flat 2 40 Lavender Road NN4 5YG
CK2	4715 3206 2900 0001	12/yy*	865	27 Roseby Avenue M57X 6TH
CK3	4929 4212 3460 0821	12/yy*	356	63 Rogerham Mansions 47 Ermine Street WD17 8TH

Visa Debit

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK4	4539 7910 0173 0106	12/yy*	289	Unit 5 120 Uxbridge Road HA6 7HJ
CK5	4909 6300 0000 0008	12/yy*	891	Mews 8 Denmark 79

Electron

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK6	4917 4800 0000 0008	12/yy*	009	5-6 Ross Avenue B76 8UJ

MasterCard Credit

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK7	5457 3599 8877 6115	12/yy*	550	147 The Larches Narborough LE10 9RT

CK8	5457 3599 8877 6222	12/yy*	547	Pear Tree Cottage 68 The Green MK8 2UY
CK9	5457 3599 8877 6339	12/yy*	209	6-8 Rubbery Close Cloisters Run CV19 9JT
CK10	5457 3599 8877 6446	12/yy*	365	The Hay Market NG99 1HG

MasterCard Debit

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK11	5573 4712 3456 7898	12/yy*	159	Merevale Avenue Leicester LE10 2BU

UK Issued Maestro

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK12	6759 9500 0000 0162	12/yy*	255	12 Merevale Avenue Leicester LE10 2BU

JCB

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK13	3540 5999 9999 1047	12/yy*	209	2 Middle Wallop Merideth-in-the-Wolds LN2 8HG

American Express

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK14	3742 510276 30000	12/yy*	7054	159 Hunts Way SO18 1GW
CK15	3745 810632 70000	12/yy*	3179	70 Peelers Way B4 3AP

Diners Club

Test Number	Card Number	Expiry Date (mm/yy)	Security Code (CV2)	Address
CK16	3607 050000 1012	12/yy*	N/A	N/A
CK17	3625 960000 1002	12/yy*	N/A	N/A

yy - For expiry date always use December of the current year*

In addition to the card details, the transaction amount will also impact on the transaction result:

Amount Range From	Amount Range To	Expected Response
0.01 (e.g. £0.01)	49.99 (e.g. $49.99)	Transaction confirmed
50.00 (e.g. $50.00)	99.99 (e.g. €99.99)	Card Referred
100.00 (e.g. £100.00)	No limit	Card Declined

Appendix D: Example Redirect Form

Bellow is an example HTML form which will redirect the customer to a payment page:

```
<form action="https://www.secure-server-hosting.com/secutran/secuitems.php" method="post">
    <input type="hidden" name="filename" value="SH200000/payment.html" />
    <input type="hidden" name="transactionamount" value="10.00" />
    <input type="hidden" name="transactioncurrency" value="GBP" />
    <input type="hidden" name="shreference" value="SH200000" />
    <input type="hidden" name="checkcode" value="123456" />
    <input type="hidden" name="secuitems" value="[pd1|sku1|Product 1 (size:
large)|50.00|2|10.00]" />
    <input type="submit" value="Proceed to Payment" />
</form>
```

Appendix E: Example Payment Form

Bellow is an example HTML form to submit the card details from your payment form:

```
<form action="https://www.secure-server-hosting.com/secutran/transactionsi1.php" method="post"
name="basketform" id="basketform" onsubmit="return validateForm(this)">
    <input type="hidden" name='shreference' value='$shreference' />
    <input type="hidden" name='checkcode' value='$checkcode' />
    <input type="hidden" name='secuitems' value='$secuitems' />
    <input type="hidden" name='transactioncurrency' value='$transactioncurrency' />
    <input type="hidden" name='shippingcharge' value='$shippingcharge' />
    <input type="hidden" name='transactiontax' value='$transactiontax' />
    <input type="hidden" name='transactionamount' value='$transactionamount' />
    <input type="hidden" name="filename" value="$backfile" />
    Name: <input type="text" name="cardholdersname" /><br />
    Email address: <input type="text" name="cardholdersemail" /><br />
    Phone number: <input type="text" name="cardholdertelephonenumber" /><br />
    Street: <input type="text" name="cardholderaddr1" /><br />
    City/Town: <input type="text" name="cardholdercity" /><br />
    County: <input type="text" name="cardholderstate" /><br />
    Postcode: <input type="text" name="cardholderpostcode" /><br />
    Card number: <input type="text" name="cardnumber" /><br />
    Expiry date: <select name="cardexpiremonth">
        <option value="01">01</option>
        <option value="02">02</option>
        <option value="03">03</option>
        <option value="04">04</option>
        <option value="05">05</option>
        <option value="06">06</option>
        <option value="07">07</option>
        <option value="08">08</option>
        <option value="09">09</option>
        <option value="10">10</option>
        <option value="11">11</option>
        <option value="12">12</option>
    </select> / <select name="cardexpireyear">
        <option value="12">12</option>
        <option value="13">13</option>
        <option value="14">14</option>
        <option value="15">15</option>
        <option value="16">16</option>
        <option value="17">17</option>
        <option value="18">18</option>
        <option value="19">19</option>
        <option value="20">20</option>
    </select><br />
    CV2 number: <input type="text" name="cv2" />
</form>
```

Appendix F: Example Confirmation Page

Below is an example confirmation page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//Dtd HTML 4.01 Transitional//EN"
"http://www.w3.org/tr/html4/loose.dtd">
<html>
 <head>
  <title>Card Payment</title>
 </head>
 <body>
  <p>Your purchases:</p>
  <table width="100%">
   <tr>
    <td colspan="5"><p>Your transaction has been authorised and your card will be billed for
$transactionamount.</p><p> Please note your transaction reference number: $trannum and print
or save this page for your records.</p></td>
   </tr>
   <tr>
    <td>Code</td>
    <td>Name</td>
    <td>Quantity</td>
    <td>Price</td>
    <td>Total</td>
   </tr>
   <loopstart:shoplst:>
   <tr>
    <td>$itemcode</td>
    <td>$itemdesc</td>
    <td>$itemquan</td>
    <td>$transactioncurrency$itempric</td>
    <td>$transactioncurrency$itemtota</td>
   </tr>
   <loopend>
   <tr>
    <td colspan="3">Total:</td>
    <td colspan="2>$transactioncurrency $transactionamount</td>
   </tr>
  </table>
 </body>
</html>
```

Appendix G: Example Transaction Error Page

Below is an example transaction error page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//Dtd HTML 4.01 Transitional//EN"
"http://www.w3.org/tr/html4/loose.dtd">
<html>
 <head>
  <title>Card Payment</title>
 </head>
 <body>
  <p><strong>Your purchases :</strong></p>
  <table>
   <tr>
    <td colspan="5">TRANSACTION ERROR</td>
   <tr>
    <td>Code</td>
    <td>Name</td>
    <td>Quantity</td>
    <td>Price</td>
    <td>Total</td>
   </tr>
   <loopstart:shoplst:>
   <tr>
    <td>$itemcode</td>
    <td>$itemdesc</td>
    <td>$itemquan</td>
    <td>$transactioncurrency$itempric</td>
    <td>$transactioncurrency$itemtota</td>
   </tr>
   <loopend>
   <tr>
    <td colspan="3">Total:</td>
    <td colspan="2"><strong>$transactioncurrency$transactionamount</strong></td>
   </tr>
   <tr>
    <td colspan="5" class="message_fail" style="text-align: center; font-size: 14px; font-
weight: bold; letter-spacing: 1px;"><p>The transaction failed.</p>
     <form action="https://www.secure-server-hosting.com/secutran/secuitems.php"
method="POST">$posted_fields
      <p> Please return to the <input type="submit" name="submit" value="previous page"> to
amend the problems below.</p>
     </form>
     </td>
   </tr>
   <tr>
    <td> </td>
    <td colspan="3">$tranerrdesc</td>
    <td> </td>
   </tr>
   <tr>
    <td> </td>
    <td coldpan="3">$tranerrdetail</td>
    <td> </td>
   </tr>
  </table>
 </body>
</html>
```

Appendix H: Generating Advanced Secuitems Unique Hash

Below is a PHP example to generate the unique hash:

```
$post_data = "shreference=".$shreference;
$post_data .= "&secuitems=".$secuitems;
$post_data .= "&secuphrase=".$secuphrase;
$post_data .= "&transactionamount=".$transactionamount;
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, "https://www.secure-server-
hosting.com/secutran/create_secustring.php");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post_data);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_REFERER, "{The referrer value you set in SH}");
curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_TIMEOUT, 60);
$secuString = trim(curl_exec($ch));
curl_close($ch);
```

Appendix I: API Example

Below is a PHP example to submit your request XML to the SHP API:

```
$postField = "xmldoc=" . urlencode( requestXML() );
$ch = curl_init();
    curl_setopt ($ch, CURLOPT_URL, "https://www.secure-server-hosting.com/secutran/api.php");
    curl_setopt ($ch, CURLOPT_POST, 1);
    curl_setopt ($ch, CURLOPT_POSTFIELDS, $postField);
    curl_setopt ($ch, CURLOPT_HEADER, 0);
    curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt ($ch, CURLOPT_TIMEOUT, 60);
    $xmlResponse = trim(curl_exec ($ch));
    if ($xmlResponse == "") print (date("Y-m-d H:i:s")." XML Doc Empty");
    if (curl_error($ch)) print (date("Y-m-d H:i:s")." cURL Call Failed - ".curl_error($ch));
curl_close ($ch);
print $xmlResponse;
```

Appendix J: 3-D Secure API

Below is a PHP example to process a transaction with the 3-D Secure API:

```
if(!isset($_POST['status'])) {

    $post_vars = "xmldoc=".urlencode($xmldoc);
    $post_vars .= "&script_location=".urlencode('{Redirect URL}');

    $ch = curl_init();
    curl_setopt ($ch, CURLOPT_URL, 'https://www.secure-server-
hosting.com/secutran/3dsecure_api.php');
    curl_setopt ($ch, CURLOPT_POST, 1);
    curl_setopt ($ch, CURLOPT_POSTFIELDS, $post_vars);
    curl_setopt ($ch, CURLOPT_HEADER, 0);
    curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt ($ch, CURLOPT_TIMEOUT, 60);
    $xml_text = trim(curl_exec ($ch));

    if ($xml_text == "") print (date("Y-m-d H:i:s").' XML Doc Empty');
    if (curl_error($ch)) print (date("Y-m-d H:i:s").' cURL Call Failed - '.curl_error($ch));
        curl_close ($ch);
    }
else {
    if($_POST['status'] != 'OK') {
        echo 'Your transaction failed.';
        echo $_POST['statustext'];
    }
    else {
        echo 'Your transaction succeeded.';
        echo 'Transaction Number: '.$_POST['reference'];
    }
}
```

Appendix K: Additional Check Results

CV2 Check is the card verification performed by the banks on the 3 (Visa/MasterCard) or 4-digit (Amex) code at the back of the card. Your account with us is set to require the CV2 number to be submitted for all new transactions but you may opt out of this requirement, if you are unable to obtain the CV2 number e.g. if you process mail order transactions.

AVS Check is the verification banks perform on the address. Not all banks are able to perform the check and not all addresses can be reliably verified and therefore the result of the AVS check should be accepted on advisory basis with further emphasis given to the 3-D Secure verification (available for ecommerce transactions) and the CV2 check.

AVS CV2 Check Results are provided as advisory only. Transactions are not failed based on the result unless specifically requested.

Response	Description
ALL_MATCH	All data is correct
SECURITY_CODE_MATCH_ONLY	The CV2 check was successful, the address check was not
ADDRESS_MATCH_ONLY	The address check was successful, the CV2 check was not
NO_DATA_MATCHES	Neither the CV2 or address checks were successful
DATA_NOT_CHECKES	The card issuing bank was unable to perform the check

3-D Secure

The 3-D Secure is a 2-step check performed by the Card Schemes (Visa/MasterCard) and then the Card Issuing bank.

The Card Scheme will verify whether the customer's card has been enrolled by the Issuing Bank into the 3-D Secure scheme. If the card has been enrolled, we will be able redirect the customer to their Card Issuing bank for authentication. The Card Issuing bank will then send us a response confirming the result of the authentication check.

Below is a table of possible different 3-D Secure responses:

Enrol	Auth	Description
Y	Y	Fully authenticated
Y	N	Card enrolled into the scheme, customer failed to authenticate
Y	A	The card is enrolled; the card issuing attempted to authenticate the customer but were unable to verify the status
Y	U	The card is enrolled; the card issuing bank were unable to attempt the authentication
N		The card is not enrolled into the scheme, authentication is not possible
U		We were unable to verify the enrolment status of the card

Appendix L: PayPal Supported Currencies

Alpha-3 Code	Currency Name	Country of use
AUD	Australian Dollar	Australia
CAD	Canadian Dollar	Canada
CHF	Swiss Franc	Switzerland
CZK	Czech Koruna	Czech Republic
DKK	Danish Krone	Denmark, Faroe Islands, Greenland
EUR	Euro	Europe (Eurozone)
GBP	Pound Sterling	United Kingdom
HKD	Hong Kong Dollar	Hong Kong, Macao
HUF	Hungarian Forint	Hungary
ILS	Israeli New Shekel	Israel, Palestinian territories
JPY	Japanese Yen	Japan
MXN	Mexican Peso	Mexico
NOK	Norwegian Krone	Norway
NZD	New Zealand Dollar	New Zealand, Cook Islands
PHP	Philippine Peso	Philippines
PLN	Polish Zloty	Poland
SGD	Singapore Dollar	Singapore, Brunei
SEK	Swedish Krona/Kronor	Sweden
THB	Thai Baht	Thailand
TWD	New Taiwan Dollar	Taiwan
USD	United States Dollar	United States of America

ISO 4217 data from http://en.wikipedia.org/wiki/ISO_4217