# PHP Sample Payment Page

## To get the sample page working

In the Config file, you need to edit the following variables:

1. **MerchantID** and **Password** – these need to be set to the MerchantID and Password for your gateway account (you would have been created a test gateway account during initial registration) - the details of all your gateway accounts (both test and production) are sent out to you on creation, in a New Gateway Account email.
2. **PaymentProcessorDomain** and **PaymentProcessorPort** – these need to be set to the domain and port for your payment processor entry points

e.g. if your payment processor has entry points of the form https://gwX.paymentprocessor.net:4430/, then

PaymentProcessorDomain = "paymentprocessor.net"

PaymentProcessorPort = "4430"

if your payment processor has entry points of the form https://gwX.thepaymentgateway.net/, then

PaymentProcessorDomain = "thepaymentgateway.net"

PaymentProcessorPort = "443"

1. **SecretKey** – this is a secret key that is known ONLY to your system. It is used to generate the hash keys that "sign" the data that is transmitted between the various pages (e.g. StartHere and PaymentForm pages). The reason that this data is "signed" is because it the mechanism by which it is transmitted is the customer's browser. The use of has keys means that if the customer tampers with this data in any way (e.g. lowering the amount down from £100.00 to £1.00) then this tampering is detectable. The payment form will work with the secret key supplied, but it is STRONGLY recommended that you change this to something else before using the payment form in a production environment.

|  |
| --- |
| **InformationBig.jpg The entry point for the sample page is StartHere.asp** |

## Integrating the payment form into your system

How you use the sample code will depend on your shopping cart package, and what it allows you to do. Generally, there will be 2 types of integration scenarios:

1. You are writing your own bespoke shopping cart package, and have complete freedom to implement the payment page in its entirety (this will be more likely if the shopping cart that you have written doesn't yet have a payment page, or plug into any other payment providers)
2. You are using a pre-written shopping cart, or have a bespoke shopping cart that is already integrated into other payment providers and already has a generalised payment form system that works with all of the other processors

### Scenario 1

Scenario 1 would allow you to use the sample payment form in its entirety. You would only manage the passing of data to the payment form, which would then basically handle the payment and the displaying of the transaction results to the customer. There is an empty "hook" function that is available in the **PaymentFormHelper** class – "**ReportTransactionResults()**", that allows you to run any updates to your system/database based on the results of the transaction that has just been processed. This function is called by the payment form when the transaction status changes, and all you would need to do is put your system-specific code in here to perform whatever post-processing tasks you need to. TIP: We recommend that you minimise the amount of actual lines of code that goes into this function (by using calls to sub functions in \*your\* files) - that means that the actual difference between your modified version of this file and the original is minimised (which will be extremely helpful if you ever need to update your versions of our files with any newer versions - e.g. to implement bugfixes etc).

### Scenario 2

Scenario 2 means that you would not be able just drop the payment form into your system as a complete and functioning form. Even though, this means that most of the sample payment form code will not of (apart from as an example of a fully functioning payment form), there are some parts of it that will be of great interest:

1. PaymentFormHelper file. This file contains the functions that the sample pages use across various files. These functions are paramount to a successful integration. You can use these as a guide when constructing your own integration.
2. The code within the PaymentForm and PreProcessPaymentForm files will demonstrate how and when to use the required functions within the PaymentFormHelper file.

## IMPORTANT THINGS TO NOTE

* When you receive your production gateway account details (**MerchantID** and **Password**), you only need to change these details in your system to switch from your test account to your production account. You won't need to change ANY other details of the transaction messages, or where they get routed to.
* The gateway entry points will be included as part of the "**New Gateway Account**" email that you would have received from our system when this account was created. If you are developer who is NOT the merchant, then this email will have been sent to the merchant – in this case, you could get this information by requesting a copy of the "**New Gateway Account**" email from the merchant directly.
* Our test gateway accounts will ONLY work with the test card details that are available in the Support | Downloads section of the MMS. You will not be able to process any "real" credit cards through your test account, just as you will not be able to process any test credit cards through your production account. This is intended to stop merchants accidently going live with their test account, and sending out real good based on what they think are real transactions (but are actually test ones)

|  |
| --- |
| **InformationBig.jpg VERY IMPORTANT:**  **IF YOU DON'T READ ANYTHING ELSE, THEN READ THIS!** |

* The "**Integration Library**" makes the process of developing custom integrations MUCH easier – the classes contained do 95% of the heavy lifting of the integration. **Using them can MASSIVELY reduce the amount of time it takes to complete (and so reduce the cost of) the integration process**. They are also very mature and stable, and are in use by many of our existing merchants. If you find that you have to contract a developer to write a custom module for you, then we would STRONGLY recommend that you ask them to use the "**Integration Library**" files as a basis for their integration.