Sage Pay Server
Integration and Protocol Guidelines 3.00

Published: 27/08/2015

Table of Contents

Document Details

Version History

Date	Change	Page
19/07/2013	Document published.	---
	Added Expiry Date as a returned field.	67
	Basket XML includes Discounts.	56
	Allowed characters in BankAuthCode now Alphanumeric.	67
27/02/2014	New screenshots.	---
	References to Sage Pay website updated.	---
	European Payment Information updated.	---
	Removed reference to Laser Cards.	---
	Surcharge XML clearer.	38
	Added StoreToken field.	49
01/08/2014	Rebranded.	---
	Included additional fields for Financial Institutions (MCC 6012).	50
	Information on pre-authorisations.	34
	Sage Software.	39
	3D-Secure simulation.	27
	XML snippets moved to sagepay.com	---
	Updated Test Cards.	27
	Added OK REPEATED response.	09
	Added PPro / PayPal indicators.	---
	Basket XML Amendments.	53
	Maestro LUHN exception.	11
13/02/2015	Updated payment pages to responsive designs	27
27/08/2015	Remove totals validation from BasketXML	55

Legal Notice

This Protocol and Integration Guidelines document ("Manual") has been prepared to assist you with integrating your own (or your client's) service with Sage Pay's payment gateway. You are not permitted to use this Manual for any other purpose.

Whilst we have taken care in the preparation of this Manual, we make no representation or warranty (express or implied) and (to the fullest extent permitted by law) we accept no responsibility or liability as to the accuracy or completeness of the information contained within this Manual. Accordingly, we provide this Manual "as is" and so your use of the Manual is at your own risk.

In the unlikely event that you identify any errors, omissions or other inaccuracies within this Manual we would really appreciate it if you could please send details to us using the contact details on our website at www.sagepay.com.

We may update this Manual at any time without notice to you. Please ensure that you always use the latest version of the Manual, which we publish on our website at www.sagepay.com, when integrating with our payment gateway.

1.0 Introduction

This guide contains all essential information for the user to implement Sage Pay using Server integration.

Sage Pay's Server integration provides a secure, simple means of authorising credit and debit card transactions from your website. In addition, you can accept payments via PayPal and local European Payment methods.

Sage Pay's Server integration provides a straightforward hosted payment interface for the customer and takes complete responsibility for the online transaction, including the collection and encrypted storage of payment data, eliminating the security implications of storing such sensitive information on your own servers.

Server integration talks directly to your web server over a direct, encrypted channel, exchanging digitally signed messages to register the transaction and notify you directly of the authorisation results. No sensitive information is sent via the customer's browser, and because the customer is redirected to Sage Pay, no card details need to be taken or stored on your site (removing the need for you to maintain highly secure encrypted databases or undergo extensive auditing against the PCI-DSS security standard).

This document explains how your web servers should communicate with Sage Pay, how to integrate with our test and live environments, and contains the complete Server protocol in the Appendix.

Indicates additional information specific to European Payment method transactions.

Indicates additional information specific to PayPal transactions.

2.0 Overview of Server Integration

The final 'Pay Now' button on your website is your link to the Sage Pay gateway. Once the customer has selected their purchases, entered billing and delivery details on your site, you present them with a 'Pay Now' button which triggers a secure web post from your servers to Sage Pay, registering the transaction. In response we return a registration status, further transaction identifiers for you to store, and a URL to which your site should redirect the customer.

The redirected customer arrives on the Sage Pay hosted payment page where they enter their payment details. The Sage Pay payment pages will present the customer with available payment methods and allow them to enter their payment details. The hosted payment pages can carry your logo and a `Description` of the goods that the customer is paying for, so they can remain confident they are buying from you. You can even customise the payment pages to carry the look and feel of you site at no additional cost. You can download our payment page templates from sagepay.com. Please note; the most recent responsive designs are not yet customisable but you can continue to customise and use our older design.

Once the customer has selected their payment method and supplied their details, they are shown a full summary or their order, including the basket contents (provided this was included in your post) and asked to confirm that they wish to process. If applicable, Sage Pay will request authentication from the 3D directory (Verified by Visa, MasterCard SecureCode and Amex SafeKey), provided the result passes the rules you have set in MySagePay, then we request authorisation from your acquiring bank. Once the bank has authorised the payment (and assuming the address and card security code results pass the rules you have set), we send a notification POST directly to your web servers, informing you of the outcome. Anti-tampering mechanisms are attached to the POST, so that you can confirm the server messages have not been modified in transit.

Having received this POST your site confirms the transaction status against your own records and replies with a final redirection URL. Your customer is redirected back to your website for confirmation of their order and any other completion pages you wish to display.

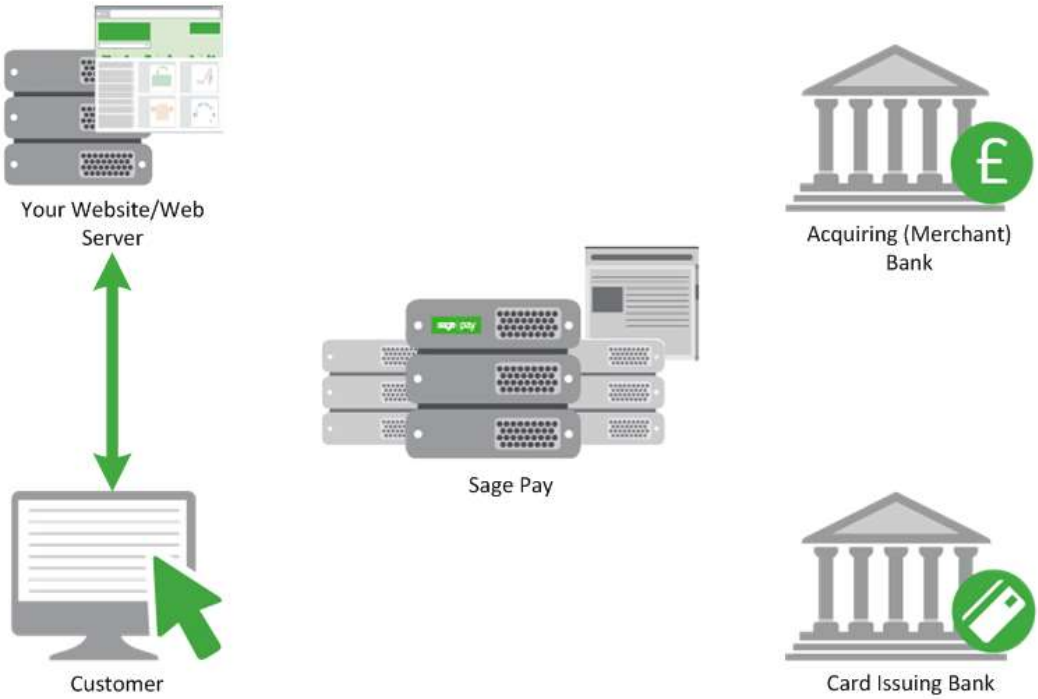
Sage Pay provides Integration Kits, which are simple worked examples in various different scripting languages that perform all the tasks described above. You simply customise these to work with your particular environment. These can be downloaded from sagepay.com.

The following sections explain the integration process in more detail. The protocol is attached in the Appendix providing a detailed breakdown of the contents of HTTPS messages sent between you servers and ours during a transaction.

A companion document, ‘Server and Direct Shared Protocols’, gives details of how to perform other transaction related POSTs, such as Refunds, Repeats, additional Authorisations and the Release/Abort mechanisms for Deferred transactions.

3.0 Server Integration in Detail

Step 1: The customer orders from your site



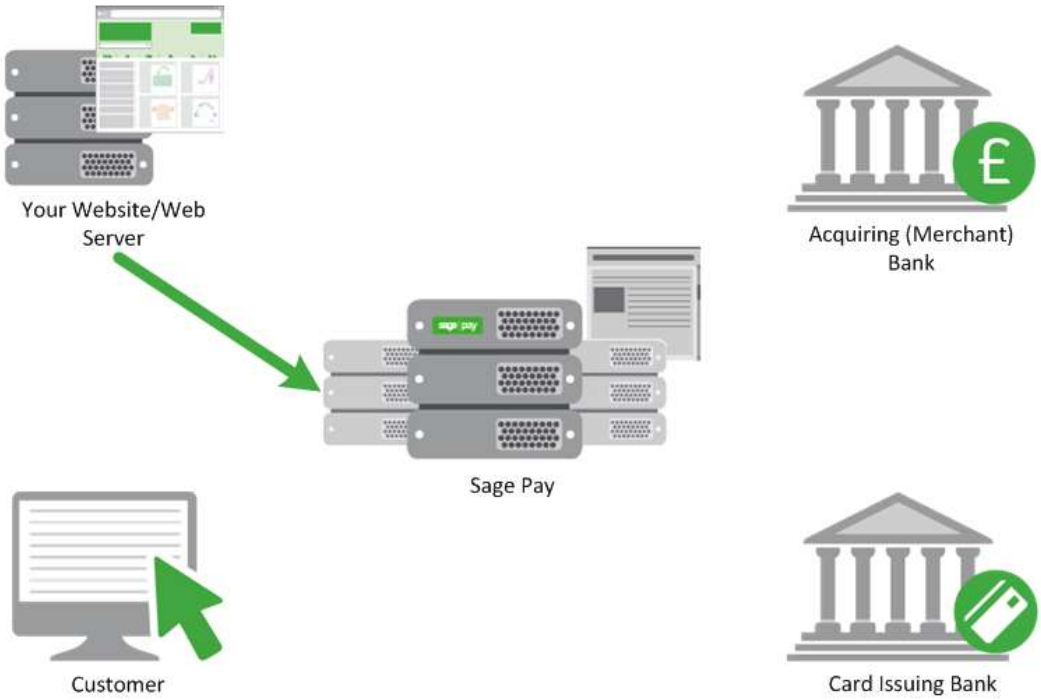
A payment begins with the customer ordering goods or services from your website. This process can be as simple as selecting an item from a drop down list, or can involve a large shopping basket containing multiple items with discounts and delivery charges. Your interaction with your customer is entirely up to you and Server integration requires you to collect only a few compulsory pieces of information, which are detailed in the latter part of this guide.

It is generally a good idea to identify the customer by name, email address, delivery and billing address and telephone number. It is also helpful to have your server record the IP Address from which the user is accessing your system. You should store these details in your session alongside details of the customer's basket contents or other ordered goods.

You do not need to collect payment data, all your site needs to do is calculate the total cost of the order in whatever currency your site operates and present the customer with a confirmation page, summarising their order. On this page there will be a 'Pay Now' button to initiate the payment process outlined in the following sections.

If you wish to apply a surcharge to a particular payment method/currency then this will be applied and shown on the subsequent payment pages.

Step 2: Your server registers the payment with Sage Pay



Once the customer has clicked 'Pay Now' on your site, a script on your web server will construct a payment registration message (see Appendix A1) and POST it via HTTPS to the Sage Pay Server transaction registration service.

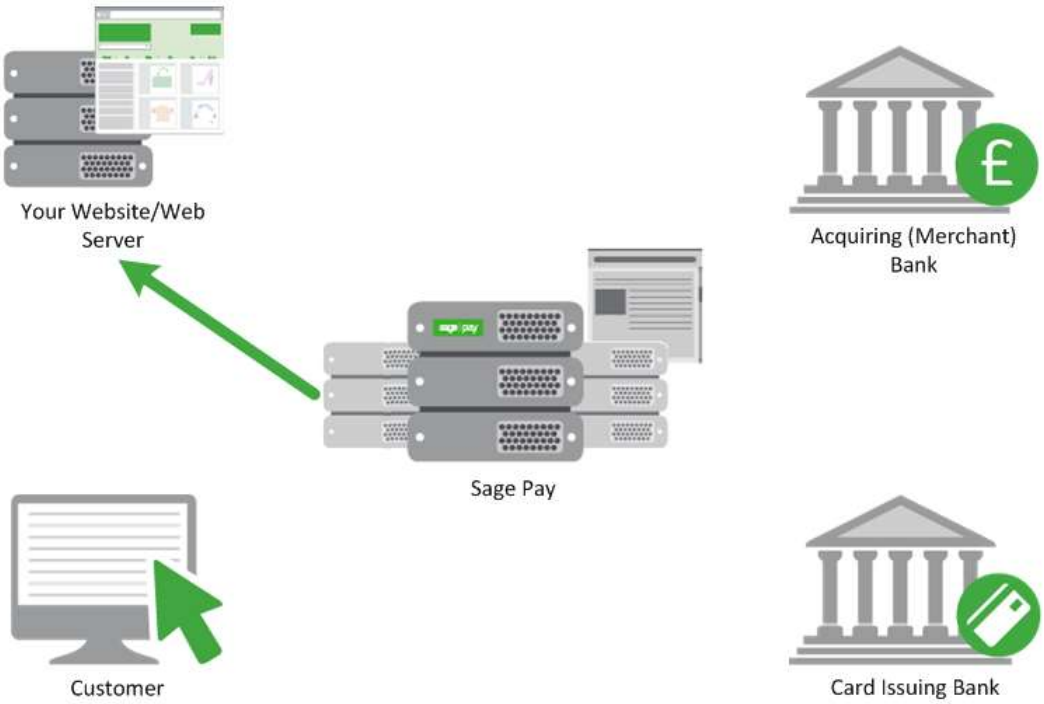
This POST contains your `VendorName` (assigned to you by Sage Pay during sign up) and your own unique reference to this payment (in a field called `VendorTxCode`, which you must ensure is a completely unique value for each transaction).

The message also contains the total value before any surcharges are applied, currency of the payment, billing and delivery address details for the customer. You must specify a brief description of the goods or services purchased, to appear on the payment pages, and provide a URL for our servers to call back to once the payment process is complete (this is called the `NotificationURL`).

Because this message is POSTed directly from your servers to ours across a 128-bit encrypted session, no sensitive information is passed via the customer's browser, and anyone who attempted to intercept the message would not be able to read it. Using the Server integration method, you can be assured that the information you send to us cannot be tampered with, or understood by anyone other than us.

Sage Pay responds to your transaction registration POST synchronously in the response object of the same POST.

Step 3: Sage Pay replies to your payment registration POST



On receipt of your POST our systems start by validating its contents.

Our server first checks to ensure all the required fields are present and that their format is correct. If any are not present a reply with a `Status` of **MALFORMED** is generated, with the `StatusDetail` field containing a human readable error message stating which field is missing. This will assist you during development stage whilst you are refining your integration.

When all fields are present the information in those fields is then validated. The `Vendor` field is checked against a pre-registered set of IP addresses, so that Sage Pay can ensure the POST came from a recognised source. The `Currency` of the transaction is validated against those accepted by your merchant accounts. The `VendorTxCode` is checked to ensure it has not been used before and the `Amount` field is validated. Flag fields are checked and the remaining fields are checked to ensure you have passed valid data. If any of the information is incorrect a reply with a `Status` of **INVALID** is returned, again with a human readable error message in `StatusDetail` explaining what was invalid.

If you receive either a **MALFORMED** or **INVALID** message you should use the detailed response in the `StatusDetail` error message to help debug your scripts. If you receive these messages on your live environment, you should inform your customer that there has been a problem registering their transaction. We recommend flagging an error in your back-office systems to help you debug.

If everything in the original POST checks out, the transaction is registered on the Sage Pay system and a new transaction code is generated that is unique across ALL vendors using our payment gateway, not just unique to you. This code, the `VPSTxId`, is our unique reference to the transaction and is sent back to you in the reply along with a `Status` of **OK** and a blank `StatusDetail` field.

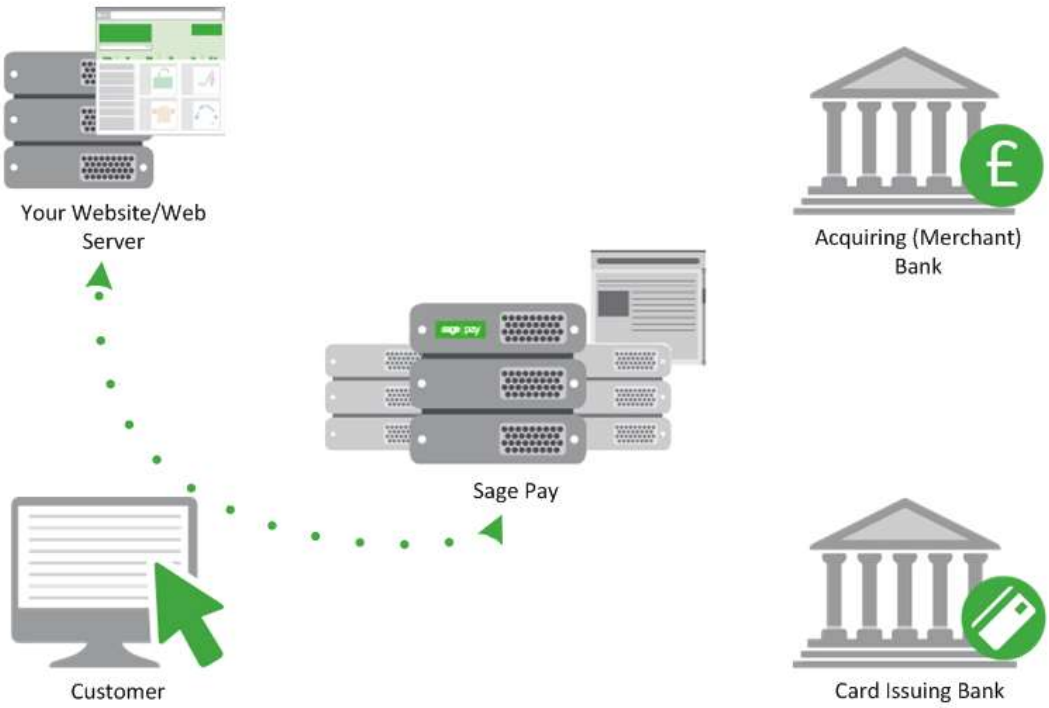
If a transaction is registered using a `VendorTxCode` already used, but that transaction has yet to complete and is still active, provided the `Amount` and `Currency` are the same you will receive an **OK REPEATED** `Status` and the same `VPSTxId`, `SecurityKey` and `NextURL` of the first request.

An **OK** or **OK REPEATED** message also contains a `SecurityKey` field. This is a ten character long, one use, alphanumeric string used as a key for confirming the MD5 hash signature in the notification POST (see Step 9).

You should store the `VPSTxId` and `SecurityKey`, along with your own `VendorTxCode`, in your database alongside the customer and order details for this transaction.

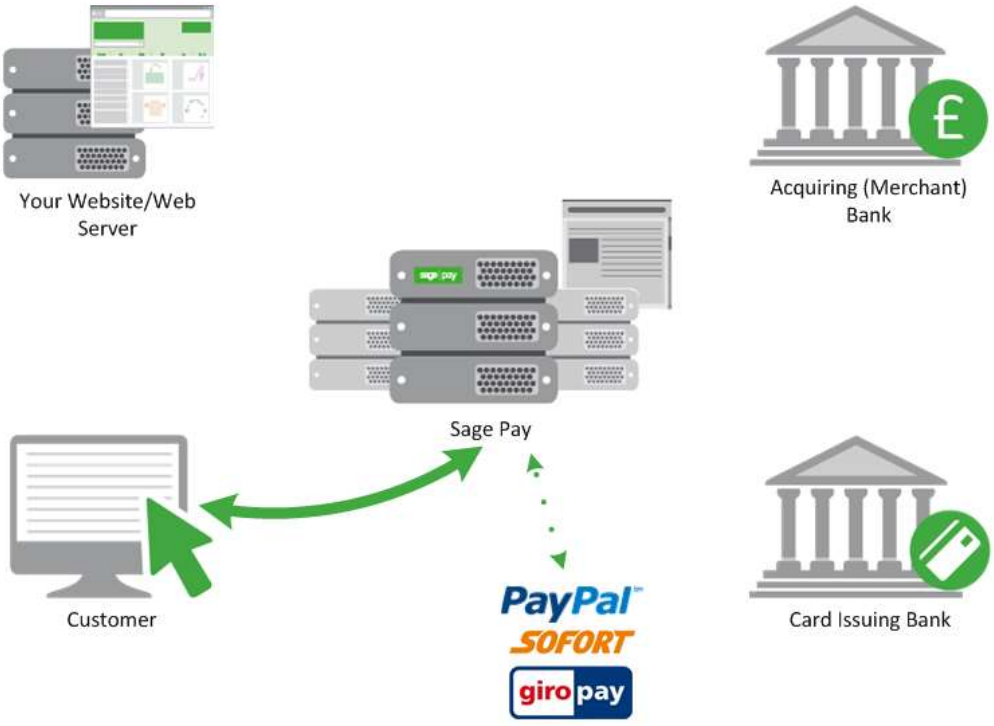
The final component of the reply is a field called `NextURL`, which is the page to which you should redirect the customer to allow them to continue with their purchase.

If the `Status` is **OK** or **OK REPEATED**, your script should send a redirect request containing this URL to your customer's browser.



This is the first stage at which anything noticeable has happened on the customers screen. The HTTPS POST and response described above are completely invisible to the customer. As far as the customer is concerned they clicked the "Pay Now" button and now find themselves on Sage Pay's payment pages.

Step 4: Customer enters payment details



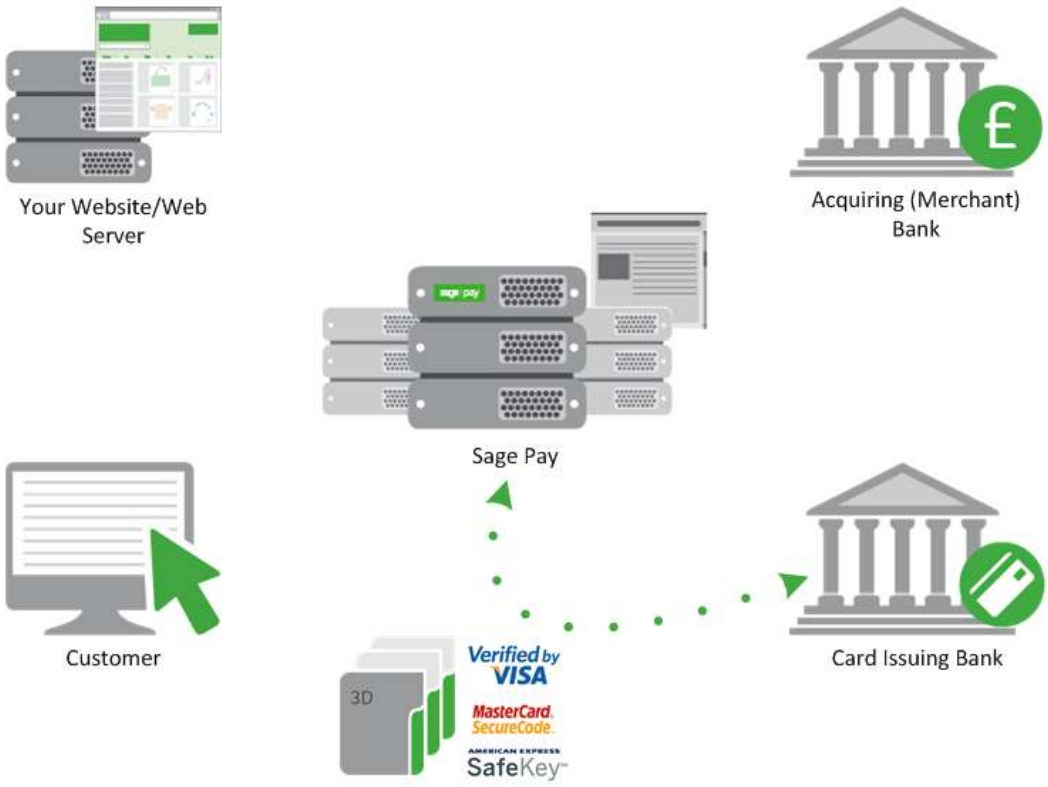
The customer is presented with a selection page where they can select a payment type. If the customer selects a card then their card details are requested. If you are a certified PayPal Business account holder and you have activated PayPal on your Sage Pay account, the PayPal option will also be displayed to your shoppers on this page. Click here to view instructions on setting up PayPal. Similarly, if you have signed up to accept European Payments, these will also appear on this page.

The payment type selection page will contain your company logo and the `Description` passed in Step 2. You can elect to customise these pages further by producing your own custom templates; these can be downloaded from sagepay.com. Please note; the most recent responsive designs are not yet customisable but you can continue to customise and use our older design.

Once the customer has entered their details, Sage Pay verifies that information prior to communicating with the bank. We ensure the card number is valid by performing a Luhn check (except for Maestro) and verification against our Issuer Identification Number database. We also check that the card type selected matches the card number and the expiry date is not in the past. If valid card details have been entered the customer is presented with an order confirmation screen where they have one last chance to change their mind and cancel the transaction.

If the customer selects PayPal on the payment type selection page the customer is redirected to PayPal to select their payment method and enter the required payment details, before being returned to the Sage Pay order confirmation screen.
If the customer selects a European Payment, then they are presented with an order confirmation screen where they have one last chance to change their mind and cancel the transaction before being redirected to their bank login page to enter their login details.
If the customer decides to cancel, you will be sent a cancellation message at the notification stage (jump to Step 9) and no details are sent to your acquiring bank.
If your Sage Pay account is not set up with 3D-Secure or this is not active for this transaction the next step is for the system to obtain authorisation (jump to Step 8).

Step 5: Sage Pay checks for 3D-Secure enrolment



The Sage Pay servers send the card details provided by your customer to the Sage Pay 3D-Secure Merchant Plug-In (MPI). This formats a verification request called a VEReq, which is sent to the 3D-Secure directory servers to query whether you, the merchant, and the card issuer are enrolled in the 3D-Secure scheme.

The 3D-Secure directory servers send a verification response called a VERes back to our MPI where it is decoded and the Sage Pay system is informed of the inclusion or exclusion of the card.

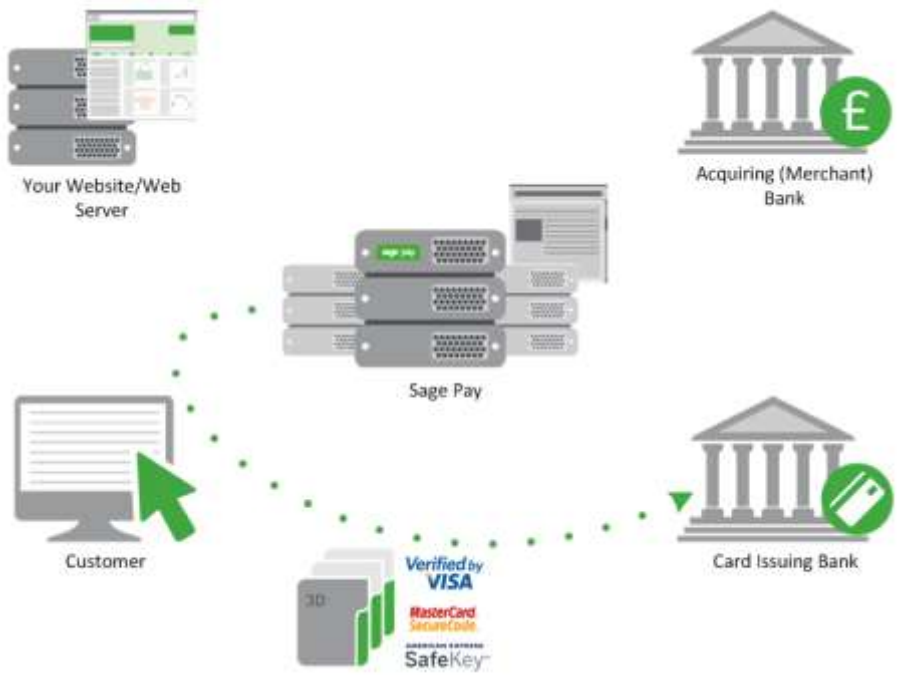
If the issuer is not part of the scheme, or if an MPI error occurs, our server will check your 3D-Secure rulebase to determine if authorisation should occur. By default, transactions that cannot be authenticated will be forwarded to your acquiring back for authorisation.

If you do have a rulebase set up, our system check the rules you have in place to determine whether you wish the customer to proceed with authorisation, or you require them to select a different payment method. In such circumstances the shopper will be returned to the card selection page for another attempt. After the 3rd unsuccessful attempt, Sage Pay will contact your `NotificationURL` with a `Status` of **REJECTED** and a `StatusDetail` indicating the reason for the failure. The `3DSecureStatus` field will contain the results of the authentication. **REJECTED** transactions will never be sent for settlement and the customer never charged, you should reply to the notification POST with a `RedirectURL` which sends your customer to an order failure page which explains why the transaction was cancelled.

If your rulebase does allow authorisation to occur for cards not enrolled, the next step is for the system to obtain this from your acquirer (see Step 8).

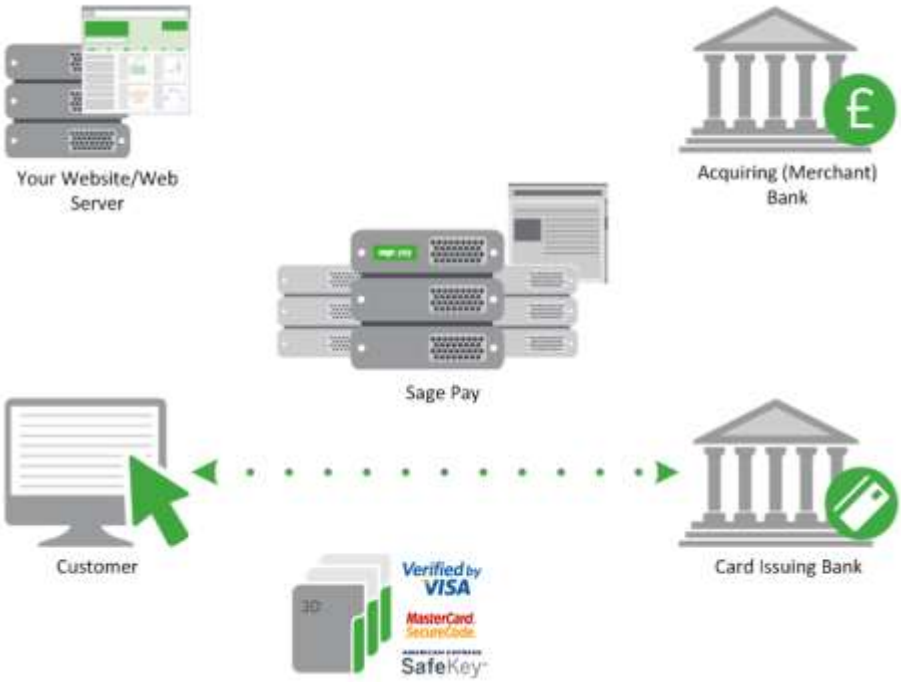
In most cases 3D-Secure verification will be possible and the process continues in the next step.

Step 6: Sage Pay redirects your customer to their Issuer



The customer's browser is redirect to their Card Issuing Bank's 3D-Secure authentication pages. These vary from bank to bank, but their purpose is to require the customer to authenticate themselves as the valid cardholder.

3D-Secure is much like an online version of Chip and Pin. The customer may be asked to answer questions at their card issuer's site (these might be a simple password, characters from a password, or numbers generated via card devices, depending on the level of security employed by the bank) and in so doing, the bank is validating the customer's right to use the card for the transaction on your site. However, some issuers will automatically authenticate what they consider low risk transactions and not ask any questions.

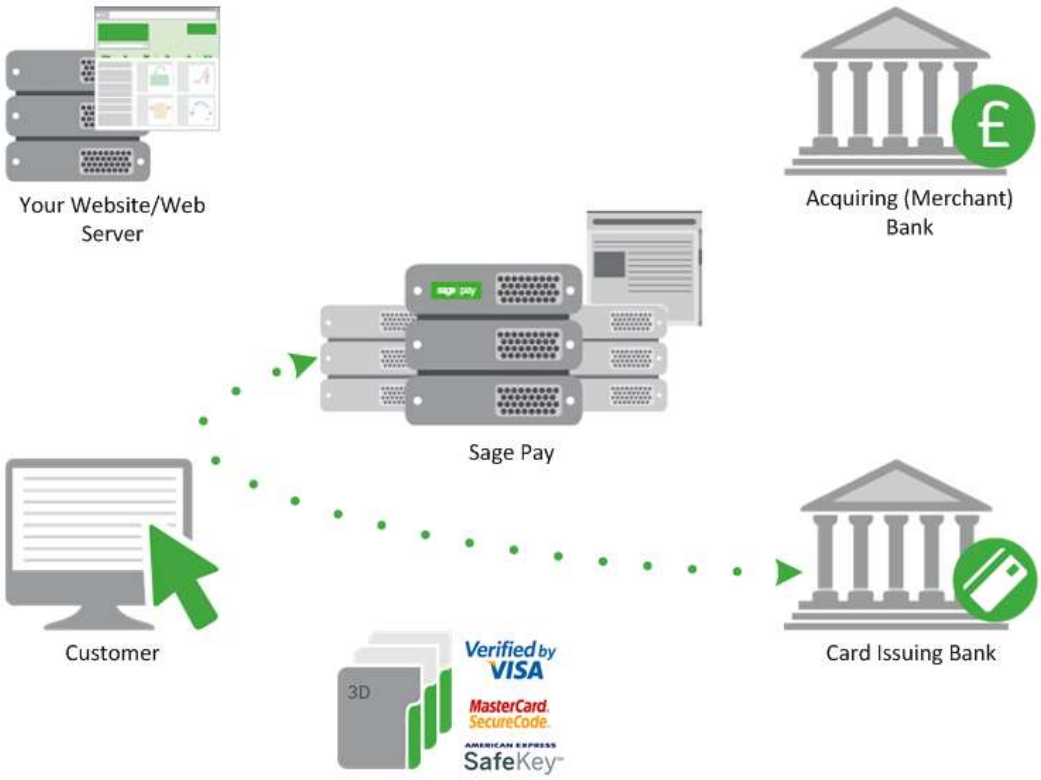


If they determine that the person attempting the transaction is the actual cardholder, they assume liability for the fraudulent use of that card during this transaction and you are protected from what are known as 'Chargebacks'.

Chargebacks occur when the cardholder subsequently challenges an authorisation with their issuing bank on the premise that it was obtain fraudulently. For more information on chargebacks and the rules around liability shift, please contact your acquiring bank.

This level of protection for you is only afforded by 3D-Secure, which is why we recommend you enable this on your Sage Pay account. You can enable and specify rules for 3D-Secure in MySagePay.

Step 7: Issuing bank returns the customer to Sage Pay

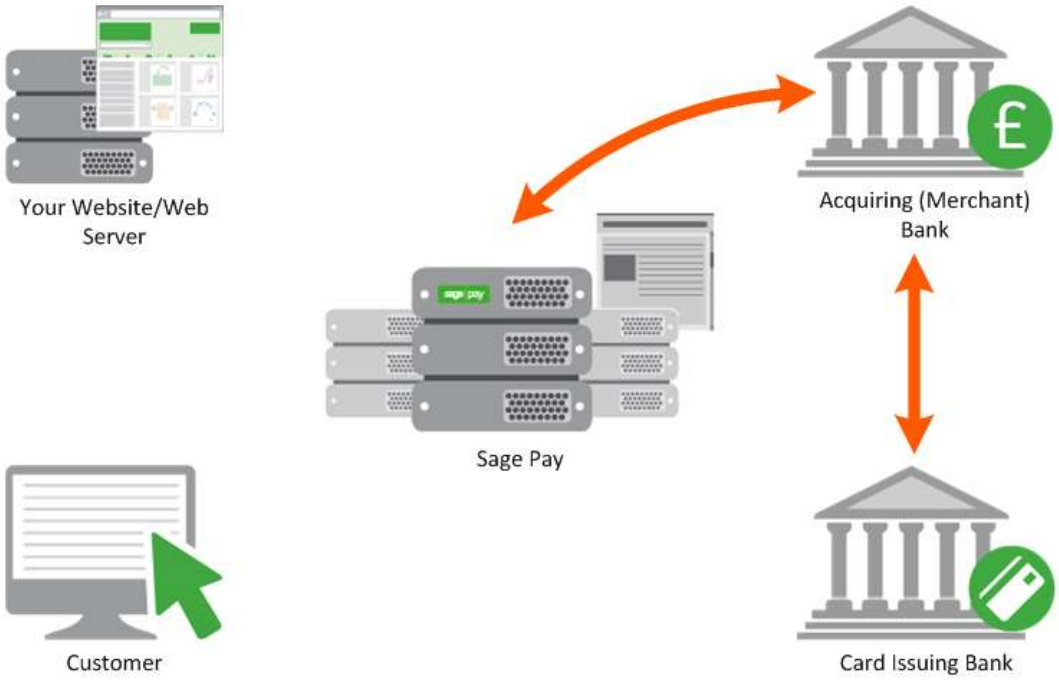


If the customer successfully completes authentication with their issuers, they are redirected to Sage Pay along with a unique authentication value (called CAVV for Visa, and UCAF for MasterCard). This is passed to your acquiring back during authorisation to secure the liability shift for the transaction.

If the customer does not successfully authenticate with their issuing bank, they are passed back to the Sage Pay server without the CAVV/UCAF value. At this stage we consult your 3D-Secure rulebase to see if authorisation should be attempted. By default 3D-Authentication failures are not sent for authorisation. For more information on 3D-Secure and rulebases, please refer to our Fraud Prevention Guide available on sagepay.com.

If authorisation is not possible, your customer is returned to the card selection screen to choose an alternative payment method. After three failed attempts our servers will POST a `Status` of **REJECTED** to your `NotificationURL`, otherwise an authorisation will be gained from your acquiring bank.

Step 8: Sage Pay servers request card authorisation



The Sage Pay servers format a bank specific authorisation message (including any 3D-Secure authentication values where appropriate) and pass it to your merchant acquirer over the private banking network.

The request is normally answered within a second or so with either an authorisation code, or a declined message. This is obtained directly from the issuing bank by the acquiring bank in real time.

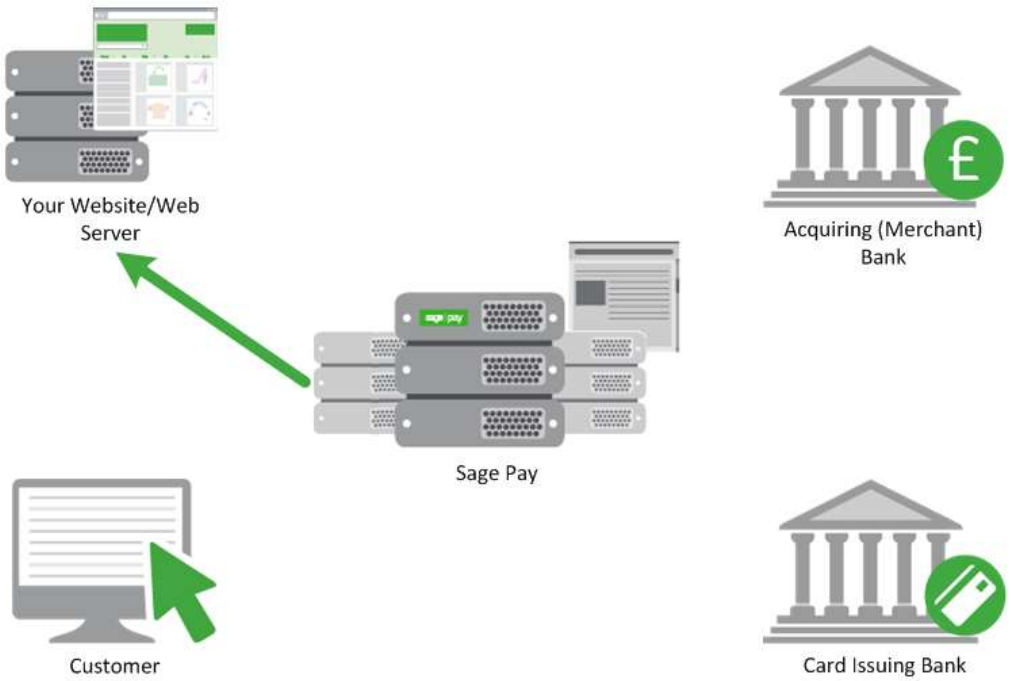
Whilst this communication is on-going, the customer is shown a page containing the text, “Please wait while your transaction is authorised with the bank”.

Sage Pay handles all authorisation failures by replying to your site with a **NOTAUTHED** message and a blank authorisation code after three failed attempts (the first two failures return the customer to the card selection screen to try another card).
If the acquirer does return an authorisation code, Sage Pay prepares an **OK** response to send back to you in Step 9.

If AVS/CV2 fraud checks are being performed, the results are compared to any rulebases you have set up (refer to our Fraud Prevention Guide available on sagepay.com). If the bank has authorised the transaction but the card has failed the fraud screening rules you have set, Sage Pay will immediately reverses the authorisation with the bank, requesting the shadow on the card for this transaction to be cleared, and prepares a **REJECTED** response.

Some card issuing banks may decline the reversal which can leave an authorisation shadow on the card for up to 10 working days. The transaction will never be settled by Sage Pay and will appear as a failed transaction in MySagePay, however it may appear to the customer that the funds have been taken until their bank clears the shadow automatically after a period of time dictated by them.

Step 9: Sage Pay contacts your NotificationURL



The Sage Pay servers send an HTTP or HTTPS POST to the NotificationURL script on your server to indicate the outcome of the transaction using ports 80 and 443. Please ensure you use these ports only as hard coding any other ports will generate errors.

This POST contains a `Status` field that holds either:

- **OK**, if the transaction was authorised.
- **PENDING** (for European Payment Types only), if the transaction has yet to be accepted or rejected
- **NOTAUTHED**, if the authorisation was failed by the bank.
- **ABORT**, if the user decided to cancel the transaction whilst on our payment pages.
- **REJECTED**, if your fraud screening rules were not met.
- **ERROR**, if an error has occurred at Sage Pay. These are very infrequent, but your site should handle them anyway. They normally indicate a problem with bank connectivity.

The `StatusDetail` field of the POST contains further human readable details about the `Status` field, explaining why a certain status was returned.

The URL to which the completion message is POSTed is the `NotificationURL` sent in the original transaction registration (in Step 2).

The transaction authorisation results are always POSTed to your `NotificationURL`, so whether the `Status` is **OK**, **PENDING**, **NOTAUTHED**, **REJECTED**, **ABORT** or **ERROR**, your notification script must decide how to process each message type and redirect the user accordingly. The integration kits have example pages that show how to process the notification POST.

The notification POST can be over HTTPS if you have an SSL certificate securing your website. If you do not then the POST will just be HTTP, which means it will be plain text and not encrypted. The problem with plain text POSTs is that a clever hacker could intercept the packets of information and modify the response before sending it on to you (although we must stress this is a very complex and

difficult process). They could, for example, change a **NOTAUTHED** message to an **OK** message. To counteract this, the notification POST has a `VPSSignature` field attached to sign the POST which is an MD5 hash of the contents of the message.
Your notification script should read the `VendorTxCode` and `VPSTxId` from the POST and retrieve the relevant information about the order from your database including, most importantly, the `SecurityKey` for the transaction (which was sent back to your servers in Step 3).

Using the `SecurityKey` and the contents of the notification POST, your script can reconstruct that message and run it through an MD5 Hash algorithm. Hash algorithms are one-way functions if you pass the same data through the same algorithm you'll get the same signature value every time you run it. There is no way to regenerate the original data from the signature data, even if you know the algorithm used and the key. Hashing is a standard means of digitally signing messages in this manner.

Your script can then compare the value it has generated to the `VPSSignature` value in the POST. If they match, the message has not been tampered with. If they do not match then the message may well have been altered in some way and you can act accordingly by declining the transaction.

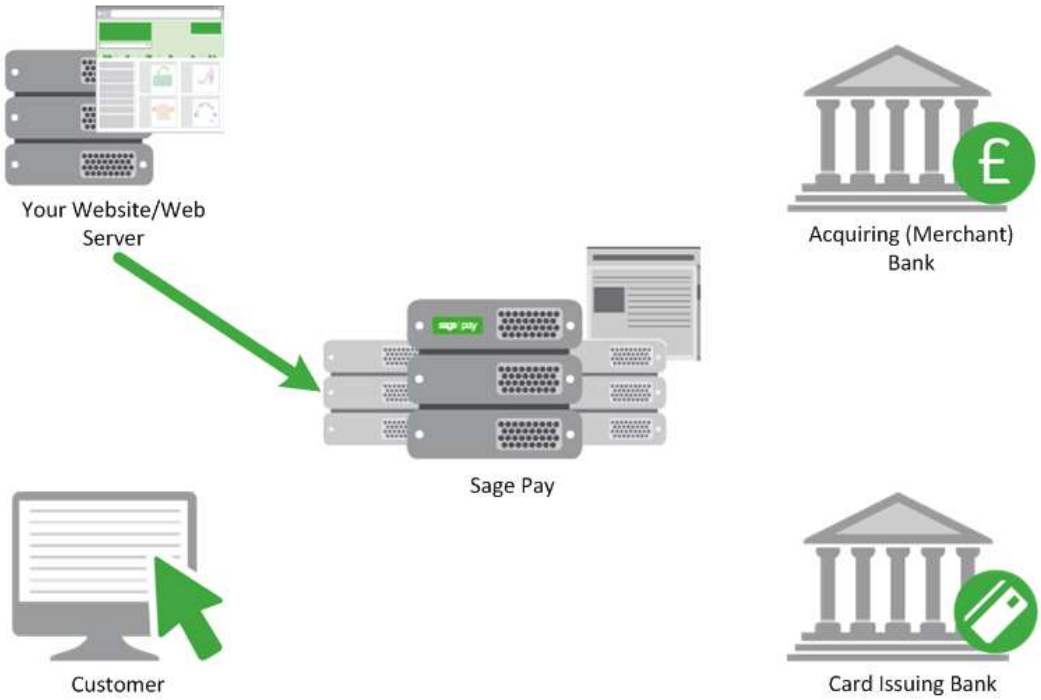
If the Hash values match you should store the `TxAuthNo` field from the notification POST in your database alongside the `VendorTxCode`, `VPSTxId` and `SecurityKey`. The `TxAuthNo` field DOES NOT contain the actual Authorisation Code sent by the bank (this is returned in the `BankAuthCode` field) but contains instead a unique reference number to that authorisation that we call the `VPSAuthCode`. This is a unique transaction ID sent to the bank during settlement so the bank may use this value to refer to your transaction if they need to contact you about it.

As mentioned above, your notification script must reply to the notification POST in all circumstances, irrespective of the `Status` of the message. Without this acknowledgement the Sage Pay Transaction Monitor will cancel the transaction and keep trying to notify you of the cancellation.
If the Sage Pay servers cannot contact your `NotificationURL` on the first attempt, it will try to notify you a further 9 times at approximately 1 second intervals in case your server is busy. If your `NotificationURL` still cannot be contacted after 10 seconds (i.e. after the 10th attempt), then if the transaction is a card payment the transaction is timed out by the Transaction Monitor and never settled, so your customer is not charged.
If the transaction is timed out, the Sage Pay servers continue to attempt to send notification POSTs to your `NotificationURL` with a `Status` of **ABORT** to inform you of the cancelled transaction.

As all PayPal transactions are settled instantly if there is a problem with Sage Pay notifying you of the transaction it is possible that your PayPal Admin area will display a transaction as successful, but MySagePay will state the transaction has failed. We strongly recommend you to log into your PayPal Admin area regularly, and cancel any transactions which are displayed as failed in MySagePay to ensure your PayPal Admin area and MySagePay reconcile.

As European Payment transactions are settled instantly if there is a problem with Sage Pay notifying you of the transaction, it will still be completed with the status indicated by PPro. This means that it is possible that the transaction may have a different `Status` on your systems than that shown in MySagePay. The correct status will be that shown in MySagePay.

Step 10: You reply to the Notification Post



Your notification script should reply to the Sage Pay Server POST with three fields: `Status`, which indicates if you wish to accept the transaction notification, `StatusDetail` to hold human readable reasons for accepting the transaction or otherwise, and `RedirectURL` which is the completion page on your own site to which the customer should be redirected by the Sage Pay Server.

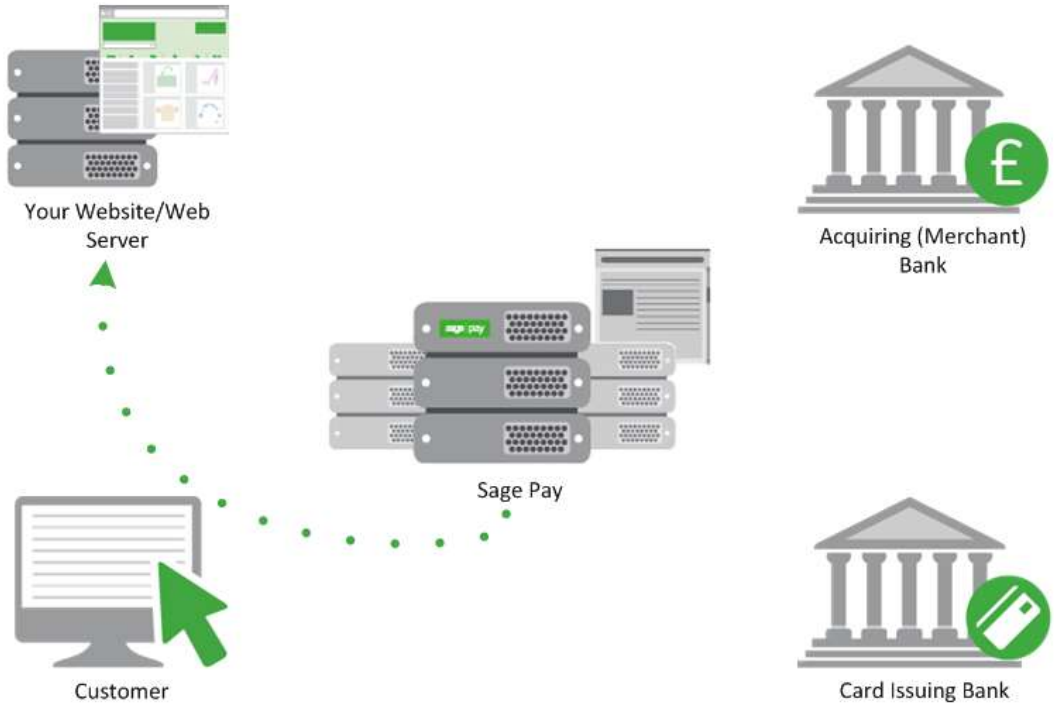
You can reply with a `Status` of either **OK**, **INVALID** or **ERROR**.
ERROR should be used very rarely, and should ONLY be sent if something unforeseen has happened on your server or database (if you receive a notification POST for a transaction you cannot find, for instance).
A `Status` of **INVALID** should be sent if you are not happy with the contents of the POST, either because the MD5 hash signatures did not match or you do not wish to proceed with the order. **OK** should be sent if you are happy with the notification and wish to proceed to charge the customer. Regardless of `Status`, the `RedirectURL` must be sent and contain a valid, fully qualified URL (i.e. an address starting https://) to the final completion page on your site to which Sage Pay will send your customer.

When the `Status` is **OK**, this is normally a page saying “Thank you for your order. Reference 123456, please visit us again.” In the case of **INVALID** or **ERROR**, the `RedirectURL` will normally point to an error page with a support telephone number.

If the `Status` field you send back to our server is anything other than **OK** then the transaction is never settled with the bank (see Step 12) and the customer is NOT charged for the goods or services. In these circumstances you should not send goods out to the customer.

If the transaction was successful with PPro then it will appear as Successful in MySagePay and the money debited from the customer’s account even if you have returned **INVALID** or **ERROR**.

Step 11: Sage Pay redirects the customer to your site

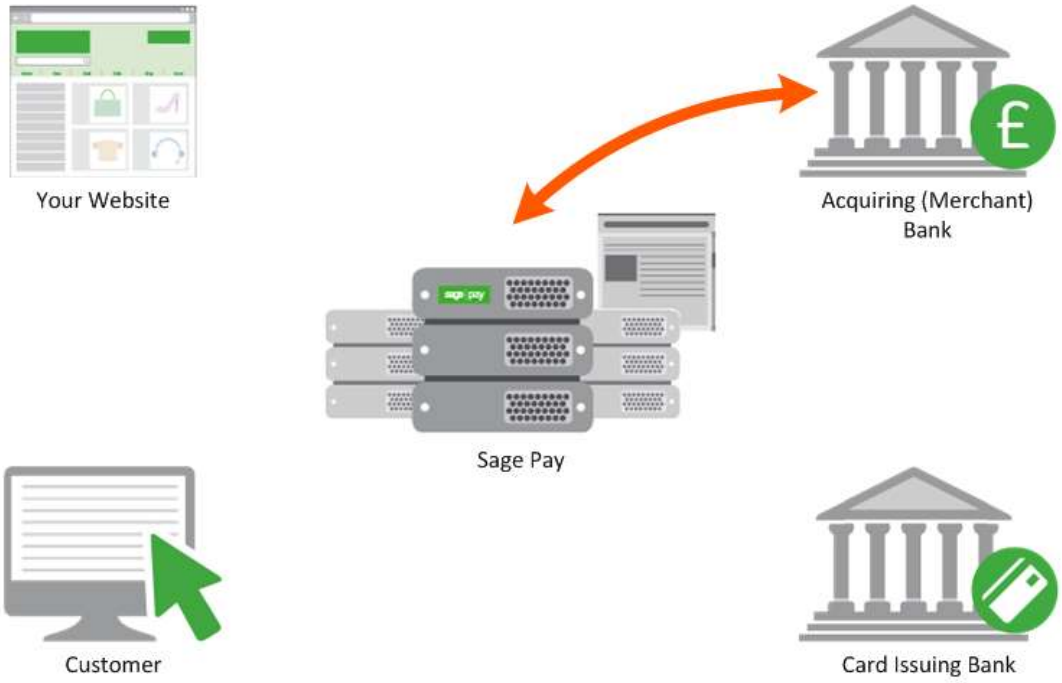


Sage Pay sends a simple HTML page to the customer's browser that redirects them to the page on your server pointed to by the `RedirectURL` field (sent in Step 10).

As before, the customer is unaware of the background POST and response process in the previous two steps. From their perspective they simply clicked "Proceed" on their payment screens, got a message saying "Authorising please wait…" and then found themselves back on your website on a completion page of some description.

The real time processing of the transaction by Sage Pay is now complete. Later in the day the final stage of the process is carried out between us and the banks without you or your site needing to do anything.

Step 12: Sage Pay sends Settlement Batch Files



Once per day, from 12.01am, the Sage Pay system batches all authorised transactions for each acquirer and creates an acquirer specific settlement file.

Transactions for ALL merchants who use the same merchant acquirer are included in this file. Every transaction (excluding PayPal and European Payment methods transactions) that occurred from 00:00:00am until 11:59:59pm on the previous day, are included in the files.

They are uploaded directly to the acquiring banks on a private secure connection. This process requires no input from you or your site. The contents of these batches and confirmation of their delivery can be found in the Settlement section of MySagePay.

Sage Pay monitors these processes to ensure files are submitted successfully, and if not, the support department correct the problem to ensure the file is sent correctly that evening or as soon as reasonably possible. Ensuring funds are available to all vendors more expediently.

The acquirers send summary information back to Sage Pay to confirm receipt of the file, then later more detailed information about rejections or errors. If transactions are rejected, we will contact you to make you aware and where possible, resubmit them for settlement.

Funds from your customers' PayPal payments are deposited into your PayPal Business account immediately, there is no settlement process. You can then withdraw or transfer the funds electronically into your specified bank account. Although PayPal transactions are included in the Settlement Reports displayed within MySagePay, as PayPal transactions are not settled by Sage Pay directly with the banks, we recommend you to log into your PayPal Admin area to obtain a report of your PayPal transactions.

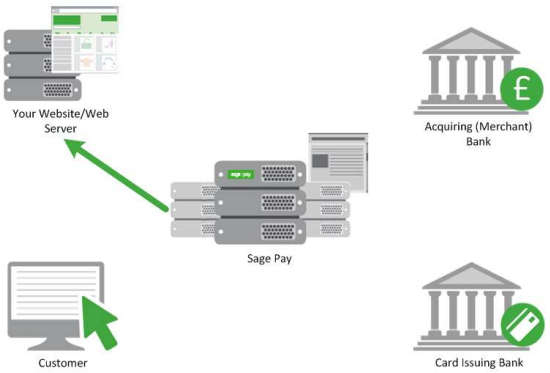
The clearing of European Payment transactions is performed on a weekly basis. From 12.01am on each Saturday a file will be received by Sage Pay that contains details of all transactions processed between Saturdays 00:00:00am to Fridays 11:59:59pm.

There may be another 2-3 days before you receive these funds as this is then transferred by PPro to your designated bank account.

4.0 The Transaction Monitor

If the Sage Pay servers are unable to inform your website, even after multiple attempts, of the status of a transaction, the transaction is placed in suspension and passed to the Transaction Monitor.

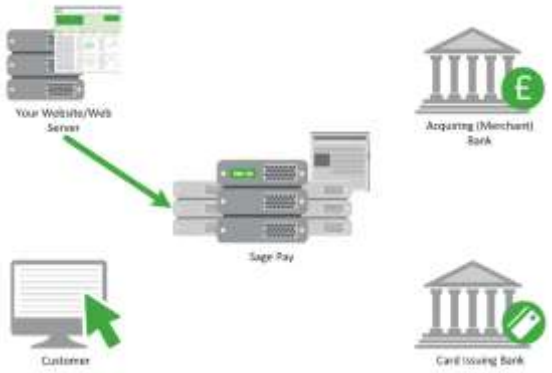
Likewise, if a customer reaches the Sage Pay payment pages changes their mind but does not click Cancel instead closes their browser, or navigates away from the payment session then the transaction is stuck in limbo and is passed to the Transaction Monitor.



Sage Pay guarantee to inform you about the success or failure of every transaction you send to us, so transactions such as those mentioned above have to be dealt with.

The Transaction Monitor is a service that runs within our secure private network, monitoring the gateway for unfinished transactions that are over 15 minutes old. When it finds one it cancels the transactions and sends a POST to your `NotificationURL` (in exactly the same manner as in Step 9 above) with a `Status` of **ABORT**.

Because the process is identical to a normal notification POST, your script should reply as it would to any **ABORT** notification POST (see step 10), with a `Status` and a `RedirectURL`. Because the user is no longer online, no redirection message will be sent to the client browser. Your site is now aware that the transaction has been cancelled and goods will not need to be shipped and the user has not been charged.



If your site does not reply to the **ABORT** Post, the service continues to try and notify you at the following intervals:

- 5 attempts at 5 minutes intervals
- 15 attempts at 15 minute intervals
- 13 attempts at 1 hour intervals
- 1 attempt per day for the next 29 days

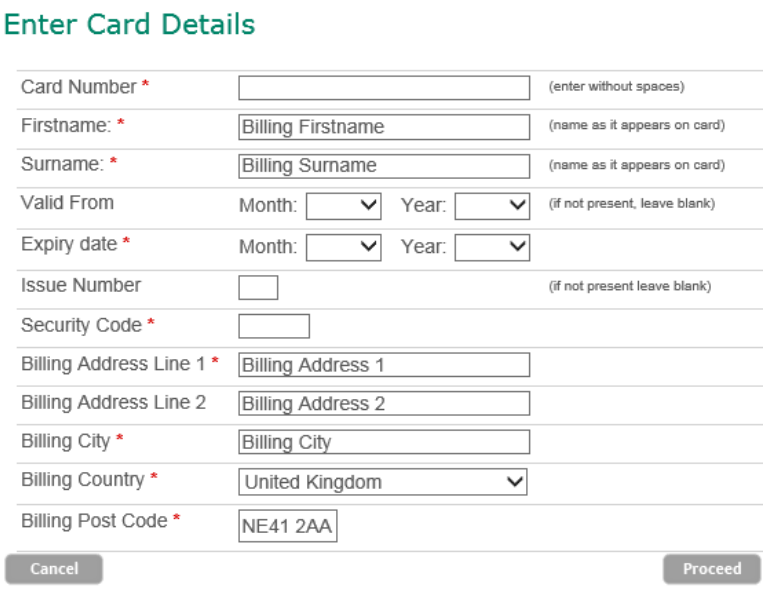
During this period the transaction is still classed as ‘active’, and therefore will not appear within MySagePay (where only completed transactions are listed). If your `NotificationURL` still cannot be contacted after 30 days, the monitor stops trying and the transaction will be marked as completed and listed as a failed transaction within MySagePay.

Transaction Monitor ignores European Payment transactions. This is because there is a guaranteed final response within 60 minutes, and so they will always have a final status.

5.0 Low Profile Payment Pages

With Server integration, you have the option of using LOW PROFILE payment pages (by sending `PROFILE`=**LOW** in your transaction registration POST). This enables you to select a less graphical, simpler set of payment pages instead of the normal default set.

Low Profile templates are designed to run inside an iframe and present simple HTML pages with no pop-ups, limited formatting and minimal graphics. This allows you to ostensibly keep the customer on your own site, whilst actually redirecting them to the Sage Pay servers to enter their card details.



To use the Server integration method in this way, you must obtain an SSL certificate for your site and serve the page containing the iframe over HTTPS. If you do not, whilst all transaction information passed between your site and the Sage Pay is encrypted using our high-security SSL certificates, from a customer's perspective the secure padlock will not display in the main browser window and they may be less likely to enter their card details into what they perceive to be an insecure site.

2.5 % surcharge for Visa

Please note that you will NOT be able to accept PayPal transactions or Local European Payments with the Low Profile templates enabled.

The Low Profile option displays a card details page to the shopper (rather than the initial 'card selection' screen); asking for the card information and billing address details.

If 3D Secure is active on your account the customer is redirected to the card issuing bank's 3D-Secure page as in normal profile, but the page will not be returned as the main content in your iframe (not wrapped with a Sage Pay screen).
Once the shopper completes 3D Authentication, (or if 3D Secure is disabled on your account), the shopper is presented with a simple scrolling progress bar, again in your iframe.

Your `NotificationURL` is contacted in the normal manner and you should reply with a `RedirectURL`.

VERY IMPORTANT: Your customer will be redirected back to the page you supply, but they will be inside your own iframe. The code on the `RedirectURL` page will need to break out of the iframe to return the customer to full screen pages on your website. You have the option of customising the Low Profile pages, so that the look and feel of the payment pages is similar to your own site. For further information about how you can customise the LOW PROFILE payment pages, please refer to the Sage Pay Custom Templates Kit on sagepay.com

6.0 Integrating with Sage Pay Server

Linking your website to Sage Pay using the Server integration method involves creating two scripts (or modifying the examples provided in the integration kits). One to register the transaction with our gateway, process the response we send back and redirect the customer across to our hosted payment pages. The other to handle the notification call-back from our servers, process the message and respond with a `Status` and `RedirectURL`.

Stage 1

The first step of the integration will be to get your site talking to Sage Pay's Test server and process all possible outcomes. This is an exact copy of the live site but without the banks attached and with a simulated 3D-Secure environment. Authorisations on the test server are only simulated, but the user experience is identical to Live, MySagePay also runs here so you can familiarise yourself with the features available to you.

The MySagePay admin system for viewing your Test transactions is at:
https://test.sagepay.com/mysagepay

Transactions from your scripts should be sent to the Sage Pay Test Server at:
https://test.sagepay.com/gateway/service/vspserver-register.vsp

Stage 2

Once you are happily processing end-to-end transactions on the test server and we can see test payments and refunds going through your account, you've completed the online Direct Debit signup and the MID has been confirmed by your Acquirer, your account on the Live Server is activated for you to start using. You will need to modify your scripts to send transactions to the live server, send through a Payment using your own credit or debit card, and then VOID it through the MySagePay Admin service so you don't charge yourself. If this works successfully, then you are ready to trade online.

The Live MySagePay admin system is at:
https://live.sagepay.com/mysagepay

Transactions from your scripts should be sent to the Sage Pay Live Server at:
https://live.sagepay.com/gateway/service/vspserver-register.vsp

7.0 Testing on the Test Server (Stage 1)

The Test Server is an exact copy of the Live System but without the banks attached and with a simulated 3D-Secure environment. This means you get a true user experience but without the fear of any funds transferring during testing.

In order to test on the Test Server, you need a Test Server account to be set up for you by the Sage Pay Support team. Your test account can only be set up once you have submitted your Sage Pay application. You can apply online here. Often when applying to trade online it takes a while for the Merchant Account to be assigned by your acquirer, so you may wish to ensure that you set those wheels in motion before you begin your integration with Sage Pay, to ensure things don't bottleneck at this stage.

The Support Team will set up an account for you on the Test Server within 48 hours of you submitting a completed application form. This will be under the same Sage Pay Vendor Name as your online application form. You will, however, be issued with different passwords for security purposes. The Support Team will let you know how to retrieve those passwords and from there how to use the MySagePay screens to look at your transactions.

To link your site to the Test Server, you need only to change your transaction registration script to send the message to the Test Server URL for Sage Pay Server. In the kits this is done simply by changing the flag in the configuration scripts to TEST. If you've been developing your own scripts, then the Test Site URL for payment registration is:
https://test.sagepay.com/gateway/service/vspserver-register.vsp

For other transaction types, the final vspserver-register.vsp section would be changed to refund.vsp, release.vsp, void.vsp etc. Please refer to the Server and Direct Shared Protocols Guide.

7.1 Registering a Payment

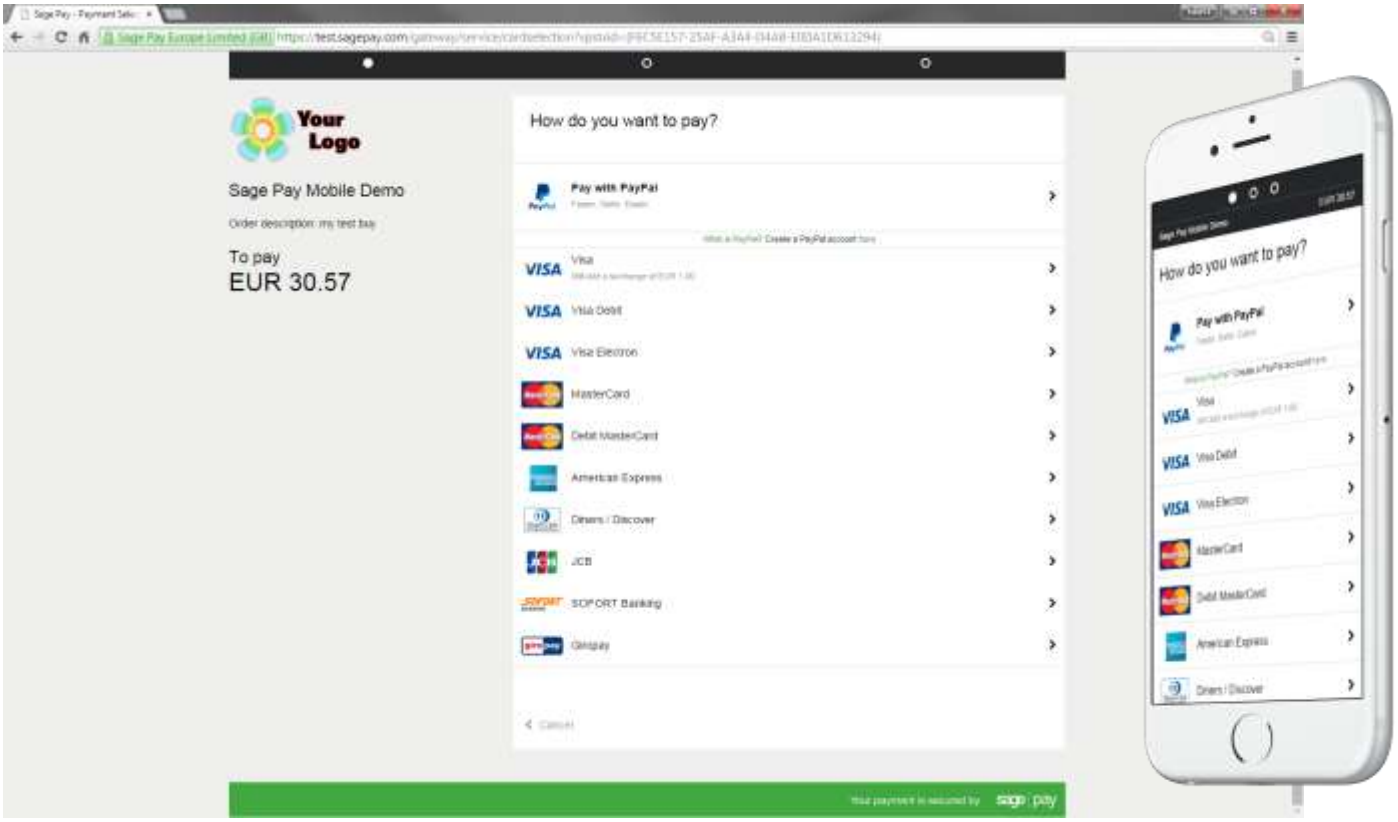
If you don't plan to implement the protocol entirely on your own, you should install the most appropriate integration kit or worked example for your platform. These can be downloaded from sagepay.com.

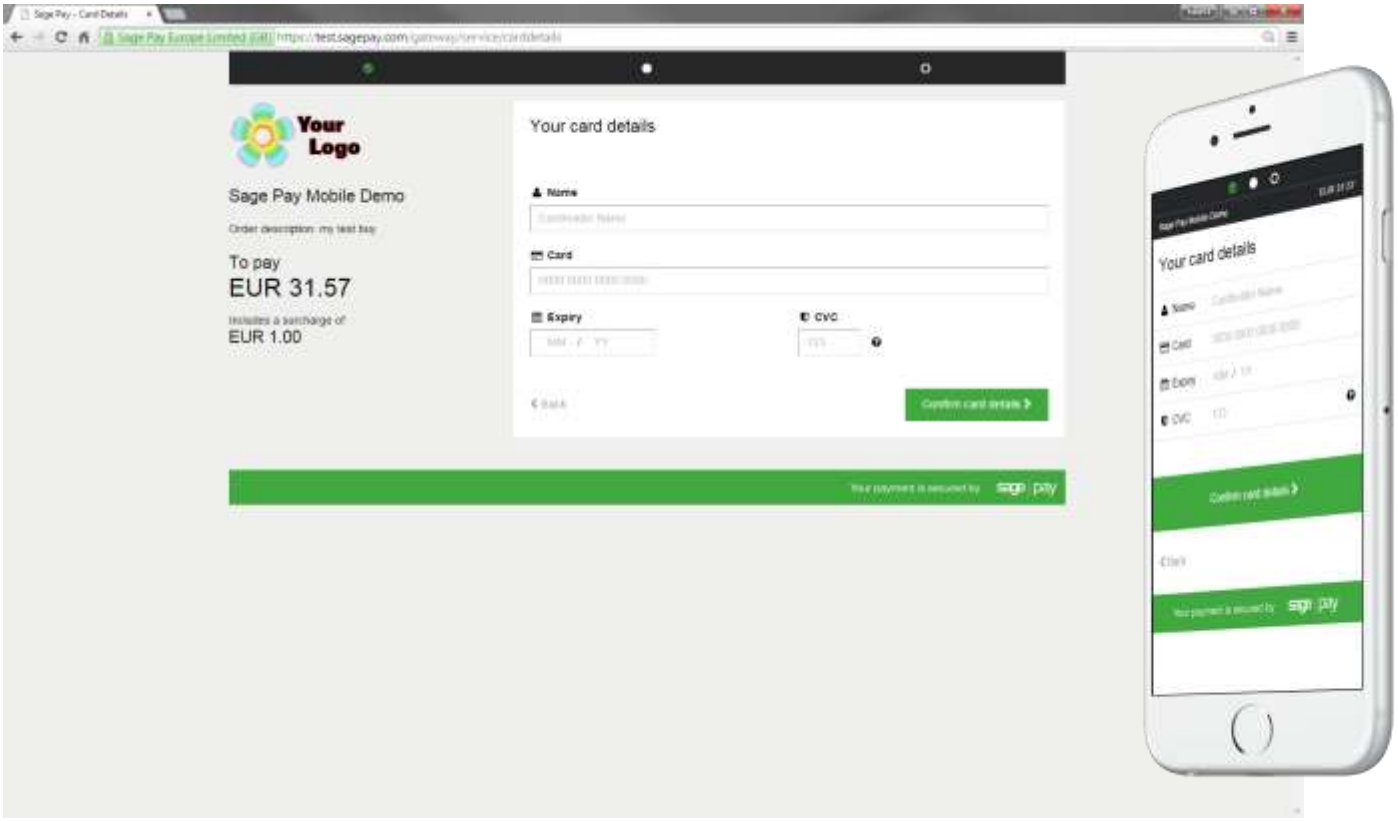
The kits will not quite run out of the box because you have to provide some specific details about your site in the configuration files before a transaction can occur, but they will provide end to end examples of registering transactions and handling notification POSTs.
The kits provide a worked example of how to construct the Transaction Registration POST (see Appendix A) and how to read the response that comes back (see Appendix B).

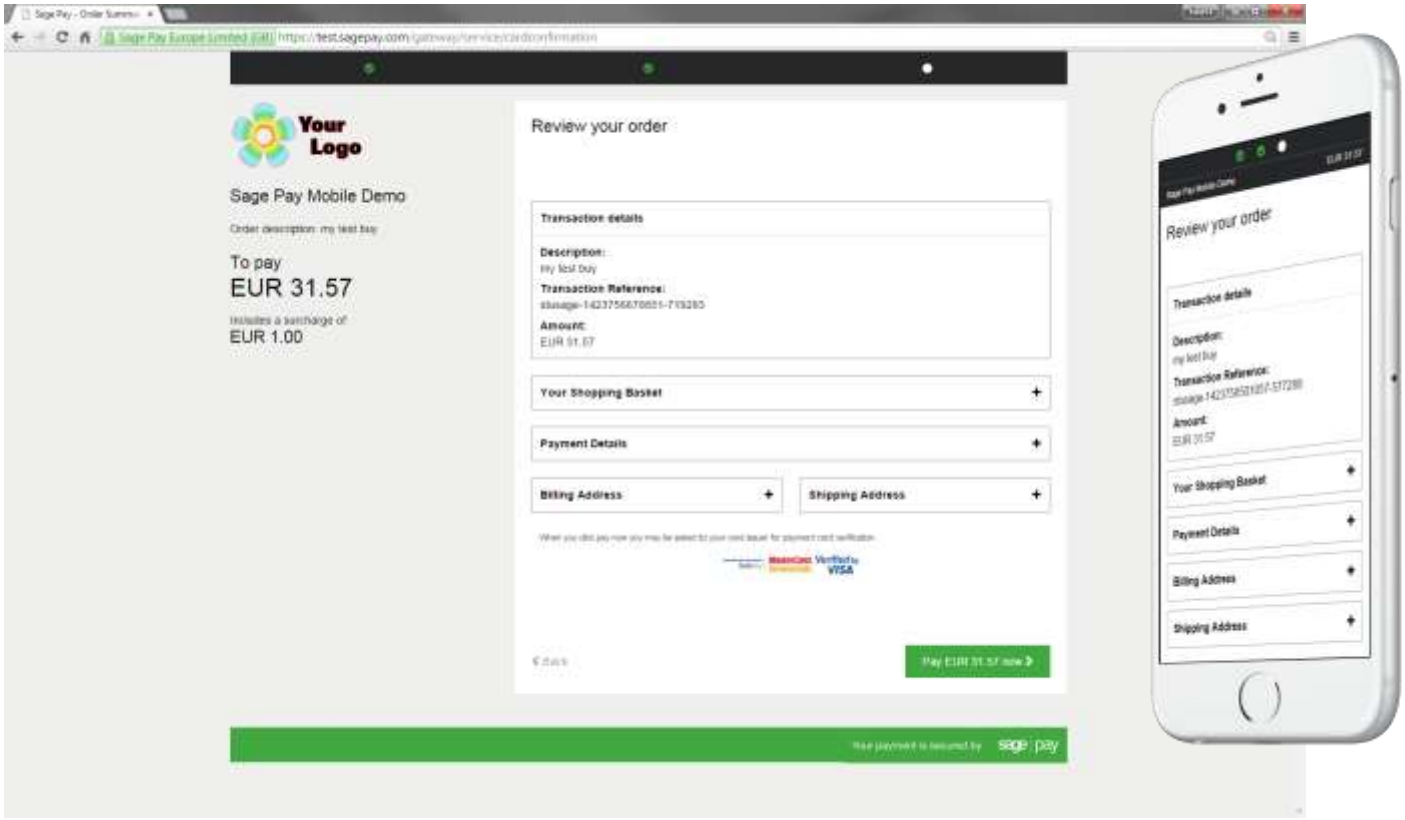
Check that this script is sending transactions to the Sage Pay test server and not the live site. Then execute this page, passing it some dummy transaction data, to send a payment registration. You may wish to modify the script at this stage to echo the results of the POST to the screen, or a file, so you can examine the `Status` and `StatusDetail` reply fields to check for errors.

Once your script can successfully register a Payment and you receive a `Status` of **OK**, you should ensure your code stores the `VPSTxId` and `SecurityKey` alongside your uniquely generated `VendorTxCode` and the order details in your own database before redirecting the browser to the URL sent by us in the `NextURL` field. When your site redirects the customer you will find yourself on the Sage Pay payment pages.

You will first be presented with our Card Selection page, where you select your payment method, then the Card Details.







If you use our iframe integration, `Profile` = **LOW** you will be presented with our one page checkout.

Enter Card Details

Card Number *		(enter without spaces)
Firstname: *	Billing Firstname	(name as it appears on card)
Surname: *	Billing Surname	(name as it appears on card)
Valid From	Month: Year:	(if not present, leave blank)
Expiry date *	Month: Year:	
Issue Number		(if not present leave blank)
Security Code *		
Billing Address Line 1 *	Billing Address 1	
Billing Address Line 2	Billing Address 2	
Billing City *	Billing City	
Billing Country *	United Kingdom	
Billing Post Code *	NE41 2AA	

Cancel Pay Now

1.5 % surcharge for MasterCard

2.00 GBP surcharge for MasterCard Debit

7.1.1 Test card numbers

You will always receive an **OK** response and an Authorisation Code from the test server if you are using one of the test cards listed below. All other valid card numbers will be declined, allowing you to test your failure pages.
If you do not use the Address, Postcode and Security Code listed below, the transaction will still authorise, but you will receive NOTMATCHED messages in the AVS/CV2 checks, allowing you to test your rulebases and fraud specific code.

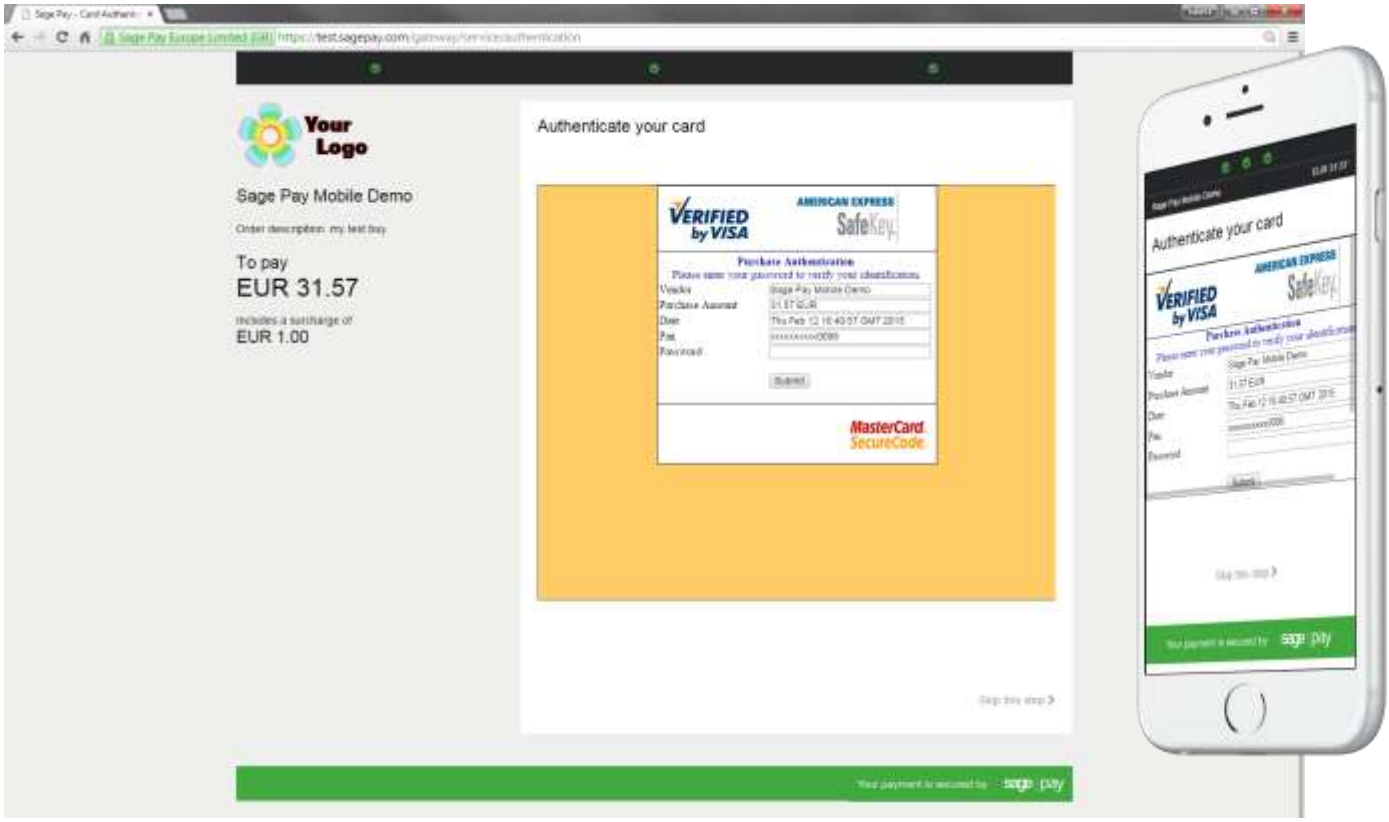
There are different cards for Visa and MasterCard to simulate the possible 3D-Secure responses.

Billing Address 1: 88
Billing Post Code: 412
Security Code: 123
Valid From: Any date in the past
Expiry Date: Any date in the future

Payment Method	Card Number	CardType Response	3D-Secure Response (VERes)
Visa	4929 0000 0000 6	VISA	Y
Visa	4929 0000 0555 9	VISA	N
Visa	4929 0000 0001 4	VISA	U
Visa	4929 0000 0002 2	VISA	E
Visa Corporate	4484 0000 0000 2	VISA	N
Visa Debit	4462 0000 0000 0003	DELTA	Y
Visa Electron	4917 3000 0000 0008	UKE	Y
MasterCard	5404 0000 0000 0001	MC	Y
MasterCard	5404 0000 0000 0043	MC	N
MasterCard	5404 0000 0000 0084	MC	U
MasterCard	5404 0000 0000 0068	MC	E
Debit MasterCard	5573 4700 0000 0001	MCDEBIT	Y
Maestro (UK Issued)	6759 0000 0000 5	MAESTRO	Y
Maestro (German Issued)	6705 0000 0000 8	MAESTRO	Y
Maestro (Irish Issued)	6777 0000 0000 7	MAESTRO	Y
Maestro (Spanish Issued)	6766 0000 0000 0	MAESTRO	Y
American Express	3742 0000 0000 004	AMEX	N/A
Diners Club / Discover	3600 0000 0000 08	DC	N/A
JCB	3569 9900 0000 0009	JCB	N/A
PayPal	Use your own PayPal Sandbox	PAYPAL	N/A

3D-Secure Response (VERes)
Y = Enrolled, will progress to PAReq (3D-Authentication)
N = Not Enrolled, will return the `3DSecureStatus` **NOTAVAILABLE**
U = Unable to verify enrolment, will return the `3DSecureStatus` **NOTAVAILABLE**
E = Error occurred during verification, will return the **ERROR**

If you have 3D-Secure set up on your test account, you can use MySagePay to switch on the checks at this stage and simulate the Verification and Authentication process.



To successfully authenticate the transaction, enter "**password**" (without the quotes) into the password field. Enter the values below (without the quotes) into the password field to simulate all other possible 3D-Secure responses:

"**A:D:06**" = Cardholder not enrolled, will return the `3DSecureStatus` **ATTEMPTONLY**
"**U:N:06**" = Authentication not available, will return the `3DSecureStatus` **INCOMPLETE**
"**E:N:06**" **=** Error occurred during authentication, will return the `3DSecureStatus` **ERROR**

Any other phrase will fail the authentication, allowing you to test your rules and 3D-Secure response handling.
The process will then continue as per the Live Servers. Only the authorisation stage is simulated.

7.2 Handling Notification response

After your site has passed the customer across to the Sage Pay payment pages, they enter their card details and the bank authorise their transaction (an **OK** response) or decline it (a **NOTAUTHED** response), or Sage Pay may reverse an authorisation if your fraud screening rules are not met (a **REJECTED** response). The customer may also change their mind and click Cancel on one of the payment pages (an **ABORT** response).

Irrespective of the `Status` Sage Pay Server needs to send you, the message is always sent to the same script on your server. We refer to this script as the Notification Script and it is pointed to by the contents of the `NotificationURL` field you sent to us in Step 2 (see Appendix A1).

This message (see Appendix B1) is POSTed to your Notification script, which should process it and reply with a `Status` and a `RedirectURL` (see Appendix B2).

Processing the Notification POST is slightly more complex because you need to validate the MD5 digital signature that is attached to the message to ensure it has not been tampered with and genuinely comes from Sage Pay. The example scripts in the Integration Kits show you how to do this, but the steps are:

1. Split the fields out of the POST to obtain the authorisation result, transaction ids and `VPSSignature` value.
2. Use the transaction ids to look up the order in your database and retrieve the `SecurityKey` passed to you during transaction registration.
3. Rebuild the Notification POST using the contents of your database and the POST itself in the order specified in the protocol (see Appendix B1).
4. Pass that data through a MD5 hashing algorithm (provided either as part of your scripting language or as part of our kits) to generate a hash value.
5. Compare that hash value to the contents of the `VPSSignature` field. If they match, the data has not been tampered with. If they do not, either the data has been modified or there is a mismatch between your data and ours, and the transaction should be cancelled.

If the signatures match, your Notification Script should respond with a `Status` of **OK** and a `RedirectURL` pointing to either an order completion page (if the `Status` was **OK**) or an appropriate order failure page (if the `Status` was **NOTAUTHED** or **ERROR**). You may wish **ABORT** messages to redirect the customer to a page providing them with alternative methods of payment, or asking them why they chose to cancel the transaction.

If the signatures do not match you should check that your code is rebuilding the message correctly. If the message is correctly built all such messages should be responded to with an **INVALID** `Status` and a `RedirectURL` pointing the user to a failure page.

If you cannot find the transaction we are notifying you about, you should return an **ERROR** `Status` and a `RedirectURL` pointing to an error page.

Your Notification URL should only respond with a `Status`, `RedirectURL` and optionally a `StatusDetail` field. No other HTML, headers, comments or text should be included either before or after these fields. Sage Pay will treat all such text as an error and fail the transaction.

For **OK** responses, you should store the `TxAuthNo` field against the other fields in your database for this transaction. This reference number uniquely identifies the transaction with your acquiring bank and they are likely to quote you this value if there are issues with it.

You should check to ensure that the notification script on your server can handle correctly all the message sent by Sage Pay (**OK**, **ABORT**, **NOTAUTHED**, **REJECTED**, **PENDING** and **ERROR**). You may also wish to add code that stores the `3DSecureStatus` and `CAVV` fields, if you plan to use 3D-Secure (Verified by Visa, MasterCard SecureCode and Amex SafeKey) and specific code that stores or reacts to the AVS and CV2 results (refer to our Fraud Prevention Guide available on sagepay.com).

Once your site can initiate transactions and handle the callbacks, then you've completed the basic Sage Pay Server integration against the test server. If you wish to link in additional processes, such as Refunds or Repeats, or the ability to Release or Abort Deferred transactions, you should continue your integration with Server & Direct Shared protocol available on sagepay.com

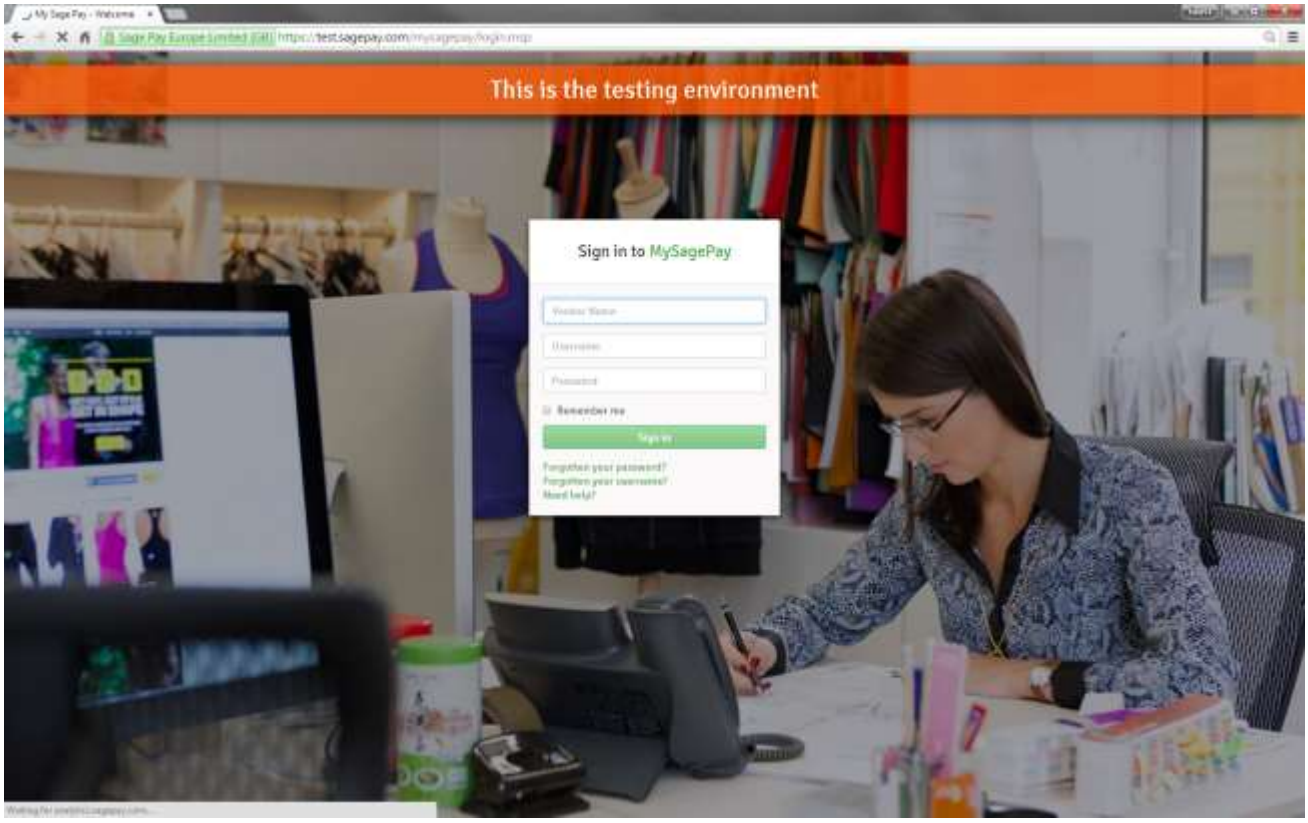
Once you've checked you can process an end-to-end transaction and tested any additional transaction types you have set up (such a Refunds and Releases) then you are almost ready to go live. Before doing so, however, you should log into MySagePay on the test servers to view your transactions and familiarise yourself with the interface.

If the transaction was successful with PPRO then it will appear as Successful in MySagePay and the money debited from the customer's account even if you have returned **INVALID** or **ERROR**.
If there was a problem with this transaction then please contact our support team on support@sagepay.com

7.2 Accessing MySagePay on Test

A Test Server version of MySagePay is available to you whilst using your test account to view your transactions, refund payments, release deferred payments, void transactions etc. You should familiarise yourself with this system on the Test Server before you go live so you know how to use the system on the Live Servers. The user guide for MySagePay can be found here.

The Test Server MySagePay can be found at: https://test.sagepay.com/mysagepay

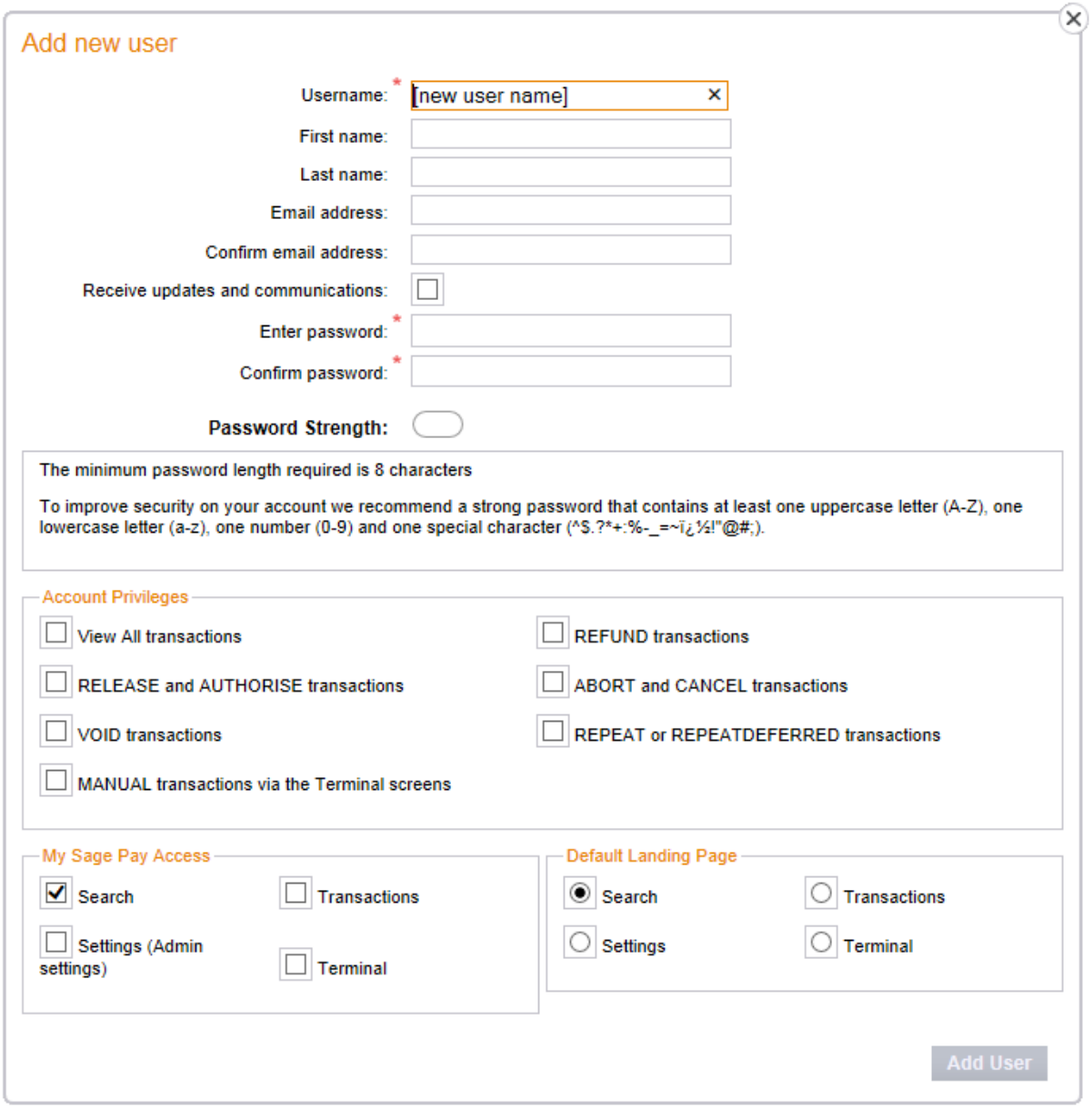


When you log in to MySagePay screens you will be asked for a Vendor Name, a Username and a Password. The first time you log in you will need to do so as your system Administrator:

- In the Vendor Name field, enter your Vendor Name, set during the application process used throughout the development as your unique Sage Pay identifier.
- In the Username field, enter the Vendor Name again.
- In the Password field, enter the MySagePay Admin password as supplied to you by Sage Pay when your test account was set up.

The administrator can ONLY access the settings Tab. You cannot, whilst logged in as administrator, view your transactions or take MO/TO payments through the online terminal.

To use those functions, and to protect the administrator account, you need to create new users for yourself and others by clicking on the 'Users' tab then the 'New User' button. You will be presented the following screen where you set the log in credentials and account privileges.



Once you have created a new user, click the Sign Out button and sign back in, this time entering:

- Your Vendor name in the Vendor Name field.
- The Username of the account you just created in the Username field.
- The password for the account you just created in the Password field.

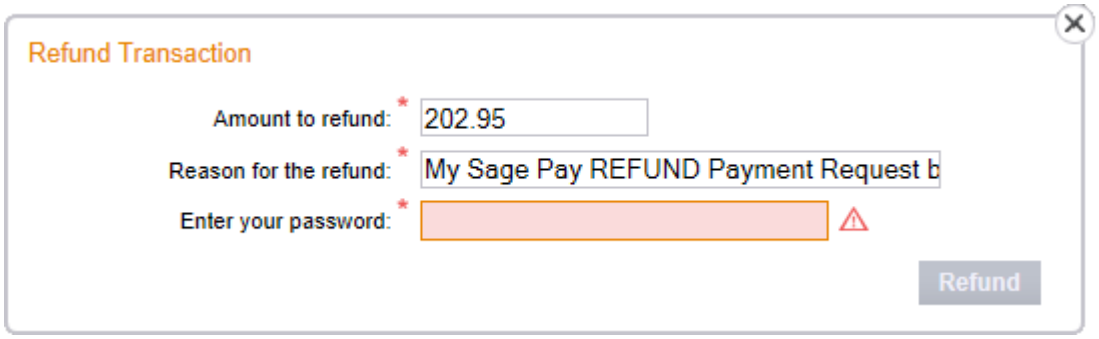
You are now logged in using your own account and can view your test transactions and use all additional functions. If you lock yourself out of your own account, you can use the Administrator account to unlock yourself or use the lost password link on the Sign In screen.

If you happen to lock out the Administrator account, you will need to contact Sage Pay to unlock it for you. Send an email to unlock@sagepay.com stating the Vendor Name and Merchant Number of the account. If you need reminding of your unique account passwords, send an email to the above and request a password retrieval link, stating the Vendor Name and Merchant Number of the account.

Detailed information on using MySagePay can be found here. Play with the system until you are comfortable with it. You cannot inadvertently charge anyone or damage anything whilst on the test server.

7.3 Refunding a transaction

Before we can set your account live, you will need to refund one of the test transactions you have already performed. This can be down by integrating with our Server & Direct Shared protocol available on sagepay.com and submitting a REFUND post. Alternatively, whilst signed in to MySagePay as a user which has privileges to refund a transaction, select the Transactions tab. Click a successful transaction and then the 'Refund' button.



You will be prompted with a screen to enter your password. You also have the opportunity to set a description for the refund and modify the amount. You cannot refund for more than the original amount.

MySagePay is also available on mobile devices.
The following features are currently available:
- List of transactions (including status)
- Transaction details
- Account activity monitoring
- Sage Pay news and alert notifications
- Sage Pay support access

8.0 Additional Transaction Types

Sage Pay supports a number of additional methods of registering a transaction and completing the payment.

8.1 DEFERRED transactions

By default a **PAYMENT** transaction type is used to gain an authorisation from the bank, and then settle that transaction early the following morning, committing the funds to be taken from your customer's card.

In some cases you may not wish to take the funds from the card immediately, merely place a 'shadow' on the customer's card to ensure they cannot subsequently spend those funds elsewhere. Then take the money when you are ready to ship the goods. This type of transaction is called a **DEFERRED** transaction and is registered in exactly the same way as a **PAYMENT**. You simply need to change your script to send a `TxType` of **DEFERRED** when you register the transaction instead of **PAYMENT**.

DEFERRED transactions are not sent to the bank for completion the following morning. In fact, they are not sent at all until you **RELEASE** them either by sending a **RELEASE** post to our servers using the Server & Direct Shared Protocol (available on sagepay.com) or by logging into MySagePay. You can release only once and only for an amount up to and including the amount of the original **DEFERRED** transaction.

If you are unable to fulfil the order, you can also **ABORT** deferred transactions in a similar manner and the customer will never be charged.

DEFERRED transactions work well in situations where it is only a matter of days between the customer ordering and you being ready to ship. Ideally all **DEFERRED** transaction should be released within 6 days. After that the shadow may disappear from the cardholders account before you settle the transaction, and you will have no guarantee that you'll receive the funds if the customer has spent all available funds in the meantime.
If you regularly require longer than 6 days to fulfil orders, you should consider using Authenticate and Authorise instead of **DEFERRED** payments.

DEFERRED transactions remain available for **RELEASE** for up to 30 days. After that time they are automatically **ABORT**ed by the Sage Pay system.

As settlement is not guaranteed to occur within 4 days for this transaction type, you may be charged a higher fee by your acquirer for ALL Deferred transactions. You should contact your Merchant Bank for more information on Pre-Authorisations.

Unlike a normal Sage Pay **DEFERRED** transaction, no shadow is placed on the customer's account for a PayPal **DEFERRED** transaction. An order is simply registered with the PayPal account and a successful authorisation for a **DEFERRED** transaction only confirms the availability of funds and does not place any funds on hold.

When you **RELEASE** a **DEFERRED** PayPal transaction, PayPal applies best efforts to capture funds at that time, but there is a possibility that funds will not be available.
We recommend that you do not ship goods until obtaining a successful release.

You cannot use the **DEFERRED** transaction type with European Payments.

8.2 REPEAT payments

If you have already successfully authorised a **PAYMENT**, a released **DEFERRED** or an **AUTHORISE** you can charge an additional amount to that card using the **REPEAT** transaction type, without the need to store the card details yourself.

If you wish to regularly **REPEAT** payments, for example for monthly subscriptions, you should ensure you have a merchant number from your bank that supports this recurring functionality (sometimes called Continuous Authority). **REPEAT** payments cannot be 3D-Secured nor have CV2 checks performed on them unless you supply this value again, as Sage Pay are not authorised to store CV2 numbers. It may be better to make use of Authenticate and Authorise if you need to vary the transaction amount on a regular basis.

You can **REPEAT** using MySagePay or by using the Server & Direct Shared Protocol. It's possible to **REPEAT** for a different `Amount` and `Currency` and supply alternative delivery address details.

The Sage Pay gateway archives all transactions that are older than 2 years old; this prevents any subsequent authorisations from being made. We therefore recommend that you repeat against the last successful authorised transaction.

You can only **REPEAT** a PayPal transaction if the initial transaction was setup as a PayPal Reference transaction, where `BillingAgreement` is set to **1**.
You will need to request approval from PayPal to enable reference transactions on your account. To request approval for a live PayPal account, contact PayPal Customer Support.
It's not possible to **REPEAT** PayPal transactions using MySagePay, you will need to submit a **REPEAT** request using the Shared Protocol.

You cannot **REPEAT** any European Payment transactions.

8.3 AUTHENTICATE and AUTHORISE

The **AUTHENTICATE** and **AUTHORISE** methods are specifically for use by merchants who are either:

- Unable to fulfil the majority of orders in less than 6 days or sometimes fulfil them after 30 days.
- Do not know the exact amount of the transaction at the time the order is placed, for example; items shipped priced by weight or items affected by foreign exchange rates.

Unlike normal **PAYMENT** or **DEFERRED** transactions, **AUTHENTICATE** transactions do not obtain an authorisation at the time the order is placed. Instead the card and cardholder are validated using the 3D-Secure mechanism provided by the card-schemes and card issuing banks, with a view to later authorise.

Your site will register the transaction with a `TxType` of **AUTHENTICATE**, and redirect the customer to the Sage Pay payment pages to enter their payment details. Sage Pay will verify the card number and contact the 3D-Secure directories to check if the card is part of the scheme. If it is not, the card details are simply held safely at Sage Pay and your `NotificationURL` is sent a `Status` of **REGISTERED**.

This also happens if you do not have 3D-Secure active on your account or have used the `Apply3DSecure` flag to turn it off.

If they have not passed authentication, your rule base is consulted to check if they can proceed for authorisation anyway. If not, your `NotificationURL` is sent a `Status` of **REJECTED**. If they failed authentication but can proceed, your `NotificationURL` is sent a `Status` of **REGISTERED**. If the customer passed authentication with their bank and a CAVV/UCAF value is returned, a `Status` of **AUTHENTICATED** and a `CAVV` value is returned, for you to store if you wish.

In all cases, the customer's card is never authorised. There are no shadows placed on their account and your acquiring bank is not contacted. The customer's card details and their associated authentication status are simply held at Sage Pay for up to 90 days (a limit set by the card schemes, 30 days for International Maestro cards) awaiting you to **AUTHORISE** or **CANCEL** via MySagePay or by using the Server & Direct Shared Protocol.

To charge the customer when you are ready to fulfil the order, you will need to **AUTHORISE** the transaction. You can authorise for any amount up to 115% of the value of the original Authentication, and use any number of Authorise requests against an original Authentication. As long as the total value of those authorisations does not exceed the 115% limit and the requests are inside the 90 days limit the transactions will be processed by Sage Pay. This is the stage at which your acquiring bank is contacted for an authorisation code. AVS/CV2 checks are performed at this stage and rules applied as normal. This allows you greater flexibility for partial shipments or variable purchase values. If the **AUTHENTICATE** transaction was **AUTHENTICATED** (as opposed to simply **REGISTERED**) all authorisations will be fully 3D-Secured.

When you have completed all your Authorisations, or if you do not wish to take any, you can **CANCEL** the **AUTHENTICAT** to prevent any further Authorisations being made against the card. This happens automatically after 90 days.

You can use the Authenticate and Authorise transaction type but the transaction will only ever be **REGISTERED** (because the transaction will never be 3D-Secured).

You cannot use the **AUTHENTICATE** transaction type with European Payments.

8.4 REFUNDs and VOIDs

Once a **PAYMENT**, **AUTHORISE** or **REPEAT** transaction has been **AUTHORISED**, or a **DEFERRED** transaction has been **RELEASE**d, it will be settled with the acquiring bank early the next morning and the funds will be moved from the customer's card account to your merchant account. The bank will charge you for this process, the exact amount depending on the type of card and the details of your merchant agreement.

If you wish to cancel that payment before it is settled with the bank the following morning, you can **VOID** a transaction using MySagePay or by using the Server & Direct Shared Protocol to prevent it from ever being settled, thus saving you your transaction charges and the customer from ever being charged. **VOID**ed transactions can NEVER be reactivated, so use this functionality carefully.

Once a transaction has been settled you can no longer **VOID** it. If you wish to return funds to the customer you need to perform a **REFUND** in MySagePay or by using the Server & Direct Shared Protocol.

You can **REFUND** any amount up to the value of the original transaction. You can even send multiple refunds for the same transaction so long as the total value of those refunds does not exceed the value of the original transaction.

The Sage Pay gateway archives all transactions that are older than 2 years old; we therefore recommend that you check the date of the original transaction which you wish to refund before processing.

You cannot **VOID** a PayPal transaction, but you are able to **REFUND** a PayPal transaction.

You cannot **VOID** any European Payment transactions, but you are able to **REFUND** them.

9.0 Applying Surcharges

The ability to apply surcharges based on the currency and payment type selected will provide a financial benefit to you by transferring the cost of these transactions to the customer.

You will have the ability to pass surcharge values (fixed amount or percentage) for all transactions except PayPal. For example, credit card = fixed fee of £2.00 or 2%.
Different surcharges can be set for each payment type/currency combination you accept.

Please note it is you responsibility to ensure that any surcharges set up comply with laws within your country.

How does it works

- You set up default surcharges for the payment types/currencies you wish to apply them to in MySagePay.
- Customers select the goods they wish to purchase from your website.
- They then select the payment type to complete the transaction.
- Alternatively you can use the `SurchargeXML` (see Appendix A1.1) to send through surcharge values that override the defaults. If the payment type selected is not sent through in the `SurchargeXML` then the default in MySagePay will be applied.

For more information, please contact our support team on support@sagepay.com

10.0 Sage 50 Accounts Software Integration

It is possible to integrate your Sage Pay account with Sage Accounting products to ensure you can reconcile the transactions on your account within your financial software.

To learn more about the integration options available and which version of Sage Accounts integrate with Sage Pay please visit sagepay.com, or email tellmemore@sagepay.com.

If you wish to link a transaction to a specific product record this can be done through the `Basket` field in the transaction registration post.

Please note the following integration is not currently available when using `BasketXML` fields.

In order for the download of transactions to affect a product record the first entry in a basket line needs to be the product code of the item within square brackets.

Example;

4:[PR001]Pioneer NSDV99 DVD-Surround Sound System:1:424.68:74.32:499.00: 499.00:[PR002]Donnie Darko Director's Cut:3:11.91:2.08:13.99:41.97:[PR003]Finding Nemo:2:11.05:1.94:12.99:25.98: Delivery:000:000:000:000:4.99

When a transaction with the `Basket` field containing the items above is imported into Sage 50 Accounts an invoice is created and product codes PR001, PR002 and PR003 are updated with the relevant activity and stock levels reduced accordingly.

For further information on the `Basket` field please see Appendix A1.2.

11.0 Going Live (Stage 2)

Once Sage Pay receives your application your account will be created and details will be sent to the bank for confirmation. The bank will be expected to confirm your merchant details within 3 to 5 working days. Once both the Direct Debit (filled out during application) and the confirmation of your merchant details reach Sage Pay, your account will become Live automatically and you will start to be billed for using our gateway.

This does not mean you will immediately be able to use your live account

You must ensure you have completed Stage 1 Testing on the Test Server, before you are granted access to your live account. Further information on testing can be found on sagepay.com.

NB – Without confirmation from the bank and without a Direct Debit submission, Sage Pay will not be able to set your account live. You will only be charged by Sage Pay when your account has valid Direct Debit details and confirmation of your merchant details from the bank.

Once your live account is active, you should point your website transaction registration scripts to the following URL:

https://live.sagepay.com/gateway/service/vspserver-register.vsp
(for other transaction types, the server-register.vsp section would be changed to refund.vsp, void.vsp, release.vsp etc.)

You should then run an end-to-end transaction through your site, ordering something relatively inexpensive from your site and paying using a valid credit or debit card. If you receive an authorisation code, then everything is working correctly.

You should then log into MySagePay on the live server https://live.sagepay.com/mysagepay.
It is worth noting here that none of the users you set up on the MySagePay system on the test server are migrated across to live. This is because many companies use third party web designers to help design the site and create users for them during testing that they would not necessarily like them to have in a live environment. You will need to recreate any valid users on the live system when you first log in as described in 5.2.

Once logged in, locate your test transaction and **VOID** it so you are not charged. At this stage the process is complete.

12.0 Congratulations, you are live with Sage Pay Server

Well done. Hopefully the process of getting here was as painless and hassle free as possible. You should contact us with any transaction queries that arise or for any help you need with MySagePay.

Here are the best ways to reach us and the best department to contact:

- If you require any information on additional services, email tellmemore@sagepay.com
- If you have a query regarding a Sage Pay invoice, email finance@sagepay.com
- If you have a question about a transaction, have issues with your settlement files, are having problems with your payment pages or MySagePay screens, or have a general question about online payments or fraud, email support@sagepay.com with your Sage Pay Vendor Name included in the mail.
- If you have any suggestions for future enhancements to the system, or additional functionality you'd like to see added, please email feedback@sagepay.com with your comments. We do take all comments on board when designing upgrades, although we may not be able to answer every mail we get.
- You can call on 0845 111 44 55, for any type of enquiry.

Your email address will be added to our group mail list used to alert you to upgrades and other pending events.

You can also always check our system availability and current issues on the Sage Pay Monitor page at www.sagepay.com/support/system-monitor.

Thanks again for choosing Sage Pay, and we wish you every success in your e-commerce venture.

13.0 Character Sets and Encoding

All transactions are simple synchronous HTTPS POSTs sent from a script on your servers to the Sage Pay gateway, with the same script reading the Response component of that POST to determine success or failure. These POSTs can be sent using any HTTPS compatible objects (such as cURL in PHP, HttpWebRequest in .NET and Apache HttpComponents in Java).

The data should be sent as URL Encoded Name=Value pairs separated with & characters and sent to the Sage Pay Server URL with a Service name set to the message type in question.

The following sections detail the contents of the POSTs and responses, between your server and ours. The format and size of each field is given, along with accepted values and characters. The legend below explains the symbols:

Aa	Letters (A-Z and a-z)	^	Caret	+	Plus
0-9	Numbers	[]	Square brackets	()	Parentheses
á	Accented characters	*	Asterisk	;	Semi-colon
&	Ampersand	'	Apostrophe (single quote)	|	Pipe
@	At sign	/\	Slash and Backslash	!	Exclamation Mark
:	Colon	-	Hyphen		Space
,	Comma	_	Underscore	~	Tilde
{}	Curly brackets	.	Full stop / Period	=	Equals
"	Quotes	$	Dollar	US	Valid 2-letter US States
#	Hash	?	Question Mark	DATE	Date in the format YYYY-MM-DD
ISO639	ISO 639-2 (2-letter language codes)	BASE64	Valid Base64 characters (A-Z,a-z,0-9,+ and /)	BOOLEAN	True or False
ISO3166	ISO 3166-1 (2-letter country codes)	CR/LF	New line (Carriage Return and Line Feed)	RFC532N	RFC 5321/5322 (see also RFC 3696) compliant email addresses
ISO4217	ISO 4217 (3-letter currency codes)	RFC1738	RFC 1738 compliant HTTP(S) URL All non-compliant characters, including spaces should be URL encoded	<HTML>	Valid HTML with no active content. Script will be filtered. Includes all valid letters, numbers, punctuation and accented characters

Appendix A: Transaction Registration

A1. You submit your transaction registration POST

This is performed via a HTTPS POST request, sent to the initial Sage Pay Payment URL service vspserver-register.vsp. The details should be URL encoded Name=Value fields separated by '&' characters.

Request format

Name	Mandatory	Format	Max Length	Allowed Values	Description
VPSProtocol	Yes	0-9 .	4 chars	**3.00**	This is the version of the protocol you are integrating with. Default or incorrect value is taken to be **3.00**.
TxType	Yes	Aa	15 chars	**PAYMENT** **DEFERRED** **AUTHENTICATE**	See companion document "Server Integration and Protocol Guidelines 3.00" for more information on the different transaction types. The value should be in UPPERCASE.
Vendor	Yes	Aa 0-9	15 chars		Used to authenticate your site. This should contain the Sage Pay Vendor Name supplied by Sage Pay when your account was created.
VendorTxCode	Yes	Aa 0-9 { } . _ -	40 chars		This should be your own reference code to the transaction. Your site should provide a completely unique VendorTxCode for each transaction.
Amount	Yes	0-9 . ,		0.01 to 100,000.00	Amount for the transaction containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.
Currency	Yes	ISO4217	3 chars	ISO 4217 Examples: **GBP**, **EUR** and **USD**	The currency the transaction is performed in. This must be supported by one of your Sage Pay merchant accounts or the transaction will be rejected.

Description	Yes	<HTML>	100 chars		Free text description of goods or services being purchased. This will be displayed on the Sage Pay Server payment page as the customer enters their card details.
NotificationURL	Yes	RFC1738	255 chars		This should be the fully qualified URL (including http:// or https:// header). It is the callback URL to which Notification POSTs are sent.
Token	No	Aa 0-9 - {}	38 chars		The Token provided during the token registration phase.
BillingSurname	Yes	Aa á ' /\ & - . ' , 0-9	20 chars		Customer billing details. All mandatory fields must contain a value, apart from the BillingPostcode. The BillingPostcode can be blank for countries that do not have postcodes (e.g. Ireland) but is required in all countries that do have them. Providing a blank field when information is required will cause an error. The BillingState becomes mandatory when the BillingCountry is set to **US**.
BillingFirstnames	Yes	Aa á ' /\ & - . ' , 0-9	20 chars		
BillingAddress1	Yes	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	100 chars		
BillingAddress2	No	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	100 chars		
BillingCity	Yes	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	40 chars		
BillingPostCode	Yes	Aa - 0-9	10 chars		
BillingCountry	Yes	ISO3166	2 chars	ISO 3166 Examples: **GB**, **IE** and **DE**	
BillingState	No	US	2 chars	Examples: **AL**, **MS** and **NY**	
BillingPhone	No	0-9 - Aa + ' ()	20 chars		
DeliverySurname	Yes	Aa á ' /\ & - . ' , 0-9	20 chars		Customer delivery details. All mandatory fields must contain a value, apart from the DeliveryPostcode. The DeliveryPostcode can be blank for countries that do not have postcodes
DeliveryFirstnames	Yes	Aa á ' /\ & - . ' , 0-9	20 chars		

DeliveryAddress1	Yes	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	100 chars		(e.g. Ireland) but is required in all countries that do have them. Providing a blank field when information is required will cause an error. The DeliveryState becomes mandatory when the DeliveryCountry is set to **US**.
DeliveryAddress2	No	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	100 chars		
DeliveryCity	Yes	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	40 chars		
DeliveryPostCode	Yes	Aa - 0-9	10 chars		
DeliveryCountry	Yes	ISO3166	2 chars	ISO 3166 Examples: **GB**, **IE** and **DE**	
DeliveryState	No	US	2 chars	Examples: **AL**, **MS** and **NY**	
DeliveryPhone	No	0-9 - Aa + ' ()	20 chars		
CustomerEMail	No	RFC532N	255 chars	Examples: me@mail1.com:me@mail2.com	The customers email address. If you wish to use multiple email addresses, you should add them using the : (colon) character as a separator. The current version of the Server integration method does not send confirmation emails to the customer. This field is provided for your records only.
Basket	No	<HTML>	7500 chars	See A1.2	You can use this field to supply details of the customer's order. This information will be displayed to you in MySagePay. If this field is supplied then the BasketXML field should not be supplied.
AllowGiftAid	No	BOOLEAN	Flag	**0** (default) **1**	This flag allows the gift aid acceptance box to appear for this transaction on the payment page. This only appears if your vendor account is Gift Aid enabled. **0 =** No Gift Aid box displayed (default) **1 =** Display Gift Aid box on payment page.

ApplyAVSCV2	No	0-9	Flag	**0** (default) **1** **2** **3**	Using this flag you can fine tune the AVS/CV2 checks and rule set you've defined at a transaction level. This is useful in circumstances where direct and trusted customer contact has been established and you wish to override the default security checks. **0 =** If AVS/CV2 enabled then check them. If rules apply, use rules (default) **1 =** Force AVS/CV2 checks even if not enabled for the account. If rules apply, use rules. **2 =** Force NO AVS/CV2 checks even if enabled on account. **3 =** Force AVS/CV2 checks even if not enabled for the account but DON'T apply any rules. This field is ignored for PayPal and European Payment method transactions.

Apply3DSecure	No	0-9	Flag	**0** (default) **1** **2** **3**	Using this flag you can fine tune the 3D Secure checks and rule set you've defined at a transaction level. This is useful in circumstances where direct and trusted customer contact has been established and you wish to override the default security checks. **0 =** If 3D-Secure checks are possible and rules allow, perform the checks and apply the authorisation rules. (default) **1 =** Force 3D-Secure checks for this transaction if possible and apply rules for authorisation. **2 =** Do not perform 3D-Secure checks for this transaction and always authorise. **3 =** Force 3D-Secure checks for this transaction if possible but ALWAYS obtain an auth code, irrespective of rule base. This field is ignored for PayPal and European Payment method transactions.
Profile	No	Aa	10 chars	**NORMAL** (default) **LOW**	A profile of **LOW** returns the simplified payment pages which have only one step and minimal formatting. Designed to run in an iframe. Omitting this field or sending **NORMAL** renders the normal card selection screen.

BillingAgreement	No	BOOLEAN	Flag	**0** **1**	If you wish to register this transaction as the first in a series of regular payments, this field should be set to **1**. If you do not have a PayPal account set up for use via Sage Pay, then this field is not necessary and should be omitted or set to **0**. **0 =** This is a normal PayPal transaction, not the first in a series of payments (default) **1 =** This is the first in a series of PayPal payments. Subsequent payments can be taken using TxType=**REPEAT**. This field is not required for non-PayPal transactions. You will need to contact PayPal directly in order to apply for Reference transactions and have the service confirmed before attempting to pass the BillingAgreement field and a value of **1** for successful repeat payments.
AccountType	No	Aa	1 char	**E** (default) **M** **C**	This optional flag is used to tell the Sage Pay gateway which merchant account to use. If omitted, the system will use E, then M, then C by default. **E =** Use the e-commerce merchant account (default). **M =** Use the mail order/telephone order account (if present). **C =** Use the continuous authority merchant account (if present). This field is ignored for PayPal transactions.

CreateToken	No	BOOLEAN	Flag	**0** (default) **1**	Use this flag to indicate you wish to have a token generated and stored in our database and returned to you for future use. **0 =** This will not create a token from the payment (default) **1 =** This will create a token from the payment if successful and return a Token.
StoreToken	No	BOOLEAN	Flag	**0** (default) **1**	Use this flag to indicate you wish to store the token being used for future use. **0 =** Do not store Token (default) **1 =** Store Token after three failed attempts or after a successful authorisation. To store a Token repeatedly a value of **1** must be passed with every use of the Token.
BasketXML	No		20000 chars	See A1.3	A more flexible version of the current basket field which can be used instead of the basket field. If this field is supplied then the Basket field should not be supplied.
CustomerXML	No		2000 chars	See A1.4	This can be used to supply information on the customer for purposes such as fraud screening.
SurchargeXML	No		800 chars	See A1.1	Use this field to override current surcharge settings in "My Sage Pay" for the current transaction. Percentage and fixed amount surcharges can be set for different payment types.
VendorData	No	Aa 0-9 _	200 chars		Use this field to pass any data you wish to be displayed against the transaction in MySagePay.
ReferrerID	No	Aa á _ /\ & – - ' , 0-9 : + () CR/LF	40 char		This can be used to send the unique reference for the Partner that referred the Vendor to Sage Pay.

Language	No	ISO639	2 chars	ISO 639-2 Examples: **EN**, **DE** and **FR**	The language the customer sees the payment pages in is determined by the code sent here. If this is not supplied then the language default of the shoppers browser will be used. If the language is not supported then the language supported in the templates will be used. Currently supported languages in the Default templates are: French, German, Spanish, Portuguese, Dutch and English.
Website	No	Aa á ' /\ & - . ' , 0-9 : + () CR/LF	100 chars		Reference to the website this transaction came from. This field is useful if transactions can originate from more than one website. Supplying this information will enable reporting to be performed by website.
FIRecipientAcctNumber	No	Aa 0-9	10 chars		This should either be the first 6 and the last 4 characters of the primary recipient PAN (no spaces). Where the primary recipient account is not a card this will contain up to 10 characters of the account number (alphanumeric), unless the account number is less than 10 characters long in which case the account number will be present in its entirety. This field is only required for UK merchants who have a merchant category code of 6012 (Financial Institutions)
FIRecipientSurname	No	Aa	20 chars		This is the surname of the primary recipient. No special characters such as apostrophes or hyphens are permitted. This field is only required for UK merchants who have a merchant category code of 6012 (Financial Institutions)
FIRecipientPostcode	No	Aa 0-9			This is the postcode of the primary recipient. This field is only required for UK merchants who have a merchant category code of 6012 (Financial Institutions)
FIRecipientDoB	No	0-9			This is the date of birth of the primary recipient in the format YYYYMMDD This field is only required for UK merchants who have a merchant category code of 6012 (Financial Institutions)

A1.1 SurchargeXML

Use this field to override the default surcharge in MySagePay for the current transaction. You can set a different surcharge value for each payment type (except PayPal). The value can either be a percentage or fixed amount.
If a surcharge amount for the payment type selected is NOT included in the Surcharge XML, then the default value for that payment type will be used from MySagePay.
If you wish to remove the surcharge value currently set in MySagePay for a payment type then you should send through the payment type with a surcharge value of 0 in the Surcharge XML. The XML tags should follow the order stated in the table.

Surcharge XML elements

Node/Element	Mandatory	Format	Max Length	Allowed Values	Description
`<surcharges>`	No	Node			The root element for all other surcharge elements.
└`<surcharge>`	Yes	XML container element			At least one must occur in the xml file. There can be multiple `<surcharge>` elements but each must have a unique `<paymentType>`.
└`<paymentType>`	Yes	Aa	15 chars	**VISA** **MC** **MCDEBIT** **DELTA** **MAESTRO** **UKE** **AMEX** **DC** **JCB**	**VISA** is Visa **MC** is MasterCard **MCDEBIT** is Debit MasterCard **DELTA** is Visa Debit **MAESTRO** is Domestic and International issued Maestro **UKE** is Visa Electron **AMEX** is American Express **DC** is Diners Club International and Discover **JCB** is Japan Credit Bureau The value should be in UPPERCASE.
└`<percentage>`	Yes unless a `<fixed>` element supplied	0-9 , .	Maximum 3 digits to 2 decimal places		The percentage of the transaction amount to be included as a surcharge for the transaction for the payment type of this element.
└`<fixed>`	Yes unless a `<fixed>` element supplied	0-9 , .			Amount of the surcharge containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.

View example Surcharge XML snippets on sagepay.com

A1.2 Basket

The shopping basket contents can be passed in a single, colon-delimited field, in the following format:

```
Number of lines of detail in the basket field:
Item 1 Description:
Quantity of item 1:
Unit cost item 1 without tax:
Tax applied to item 1:
Cost of Item 1 including tax:
Total cost of item 1 (Quantity x cost including tax):
Item 2 Description:
Quantity of item 2:
....
Cost of Item including tax:
Total cost of item
```

- The line breaks above are included for readability only. No line breaks are needed; the only separators should be the colons.
- The first value "The number of lines of detail in the basket" is **NOT** the total number of items ordered, but the total number of rows of basket information. In the example below there are 6 items ordered, (1 DVD player and 5 DVDs) but the number of lines of detail is 4 (the DVD player, two lines of DVDs and one line for delivery).

Example:

Items	Quantity	Item value	Item Tax	Item Total	Line Total
Pioneer NSDV99 DVD-Surround Sound System	1	424.68	74.32	499.00	499.00
Donnie Darko Director's Cut	3	11.91	2.08	13.99	41.97
Finding Nemo	2	11.05	1.94	12.99	25.98
Delivery	---	---	---	---	4.99

```
4:Pioneer NSDV99 DVD-Surround Sound System:1:424.68:74.32:499.00: 499.00:Donnie Darko Director's Cut:3:11.91:2.08:13.99:41.97:
Finding Nemo:2:11.05:1.94:12.99:25.98: Delivery:---:---:---:---:4.99
```

If you wish to leave a field empty, you must still include the colon. E.g. `DVD Player:1:199.99:::199.9`

A1.3 BasketXML

The basket can be passed as an XML document with extra information that can be used for:

1. Displaying to the customer when they are paying using PayPal.
2. Displaying in MySagePay to give you more detail about the transaction.
3. Displaying on the payment page. It is possible to send through a delivery charge and one or more discounts. The discount is at the order level rather than item level and is a fixed amount discount. You can however add multiple discounts to the order.
4. More accurate fraud screening through ReD. Extra information for fraud screening that can be supplied includes; details of the items ordered, and also the shipping details and the recipient details. Any information supplied will be sent to ReD to enable them to perform more accurate fraud screening.
5. The supplying of TRIPs information. However this information will only be of use to you if your acquiring bank is Elavon. TRIPs information which can be supplied includes details of airlines, tours, cruises, hotels and car rental. If your acquiring bank is Elavon this information will be sent in the daily settlement file.

NB : Please note if your customer is buying more than one service from you (i.e. more than one of following ; airlines, tours, cruises, hotels and car rental) you will need to send the information through as separate transactions.

No validation is performed on the totals of the basket, it is your responsibility to ensure that the amounts are correct and that the total of the basket matches the transaction amount sent in the Registration

Both the `Basket` field and the `BasketXML` field are optional. If basket information is to be supplied, you cannot pass both the `Basket` and the `BasketXML` field, only one of them needs to be passed.

The XML tags should follow the order stated in the table.

Basket XML elements

Node/Element	Mandatory	Format	Max Length	Allowed Values	Description
<basket>	No	Node			The root element for all other basket elements.
└<agentId>	No	Aa 0-9 _ +	16 chars		The ID of the seller if using a phone payment.

└<item>		XML container element			There can be as many Items are you like in the BasketXML, each holding a different item and recipient. The sum of all <TotalGrossAmount> in all item elements and the <deliveryGrossAmount> amount must match the Amount field sent with the transaction
└<description>	Yes	Aa á / \ - . ' , 0-9 : + ()	100 chars		Description of the item
└<productSku>	No	Aa - 0-9 +	12 chars		Item SKU. This is your unique product identifier code.
└<productCode>	No	Aa - 0-9 +	12 chars		Item product code.
└<quantity>	Yes	0-9 .	12 chars		Quantity of the item ordered
└<unitNetAmount>	Yes	0-9 .	14 chars		Cost of the item before tax containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.
└<unitTaxAmount>	Yes	0-9 .	14 chars		Amount of tax on the item containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.
└<unitGrossAmount>	Yes	0-9 .	14 chars		<unitNetAmount> + <unitTaxAmount>
└<totalGrossAmount>	Yes	0-9 .	14 chars		<unitGrossAmount> x <quantity>
└<recipientFName>	No	Aa / \ - . ' + ()	20 chars		The first name of the recipient of this item.
└<recipientLName>	No	Aa / \ - . ' + ()	20 chars		The last name of the recipient of this item.

└<recipientMName>	No	Aa	1 char		The middle initial of the recipient of this item.
└<recipientSal>	No	Aa	4 chars		The salutation of the recipient of this item.
└<recipientEmail>	No	RFC532N	45 chars		The email of the recipient of this item.
└<recipientPhone>	No	0-9 - Aa + ( )	20 chars		The phone number of the recipient of this item.
└<recipientAdd1>	No	Aa /\ - . ' , 0-9 : + ( ) CR/LF	100 chars		The first address line of the recipient of this item.
└<recipientAdd2>	No	Aa /\ - . ' , 0-9 : + ( ) CR/LF CR/LF	100 chars		The second address line of the recipient of this item.
└<recipientCity>	No	Aa /\ - . ' , 0-9 : + ( ) CR/LF CR/LF	40 chars		The city of the recipient of this item.
└<recipientState>	No	US	2 chars		If in the US, the 2 letter code for the state of the recipient of this item.
└<recipientCountry>	No	ISO3166	2 chars		The 2 letter country code (ISO 3166) of the recipient of this item.
└<recipientPostCode>	No	Aa - 0-9	9 chars		The postcode of the recipient of this item.
└<itemShipNo>	No	Aa 0-9 + -	19 chars		The shipping item number.
└<itemGiftMsg>	No	Aa 0-9 +	160 chars		Gift message associated with this item.
└<deliveryNetAmount>	No	0-9 .	14 chars		Cost of delivery before tax containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.

└<deliveryTaxAmount>	No	0-9 .	14 chars		Amount of tax on delivery containing minor digits formatted to 2 decimal places where appropriate. e.g. 5.10 or 3.29. Values such as 3.235 will be rejected. Minimum for no minor unit currencies like JPY is 1. Amounts must be in the UK currency format. The period must be used to indicate the decimal place. The comma must only be used to separate groups of thousands.
└<deliveryGrossAmount>	No	0-9 .	14 chars		<deliveryNetAmount> + <deliveryTaxAmount>
└<discounts>	No				The root element for all other discount elements.
└<discount>	Yes				There can be multiple discount elements.
└<fixed>	Yes	0-9 .	14 chars	Zero or greater	This is the amount of the discount. This is the monetary value of the discount. The value sent will be subtracted from the overall total
└<description>	No	Aa á / \ - . ' , 0-9 : + () @ {} ; _ ^ " ~ [] ° $ = ! # | ?	100 chars		This is the description of the discount. This will appear on the payment pages, MySagePay and the PayPal checkout pages if appropriate.
└<shipId>	No	Aa + 0-9	16 chars		The ship customer ID.
└<shippingMethod>	No	Aa	1 char	**C**- Low Cost **D** – Designated by customer **I** – International **M** – Military **N** – Next day/overnight **O** – Other **P** – Store pickup **T** – 2 day service **W** – 3 day service	The shipping method used.
└<shippingFaxNo>	No	0-9 - Aa + ()	20 chars		The Fax Number
└<hotel>	No				Used to provide hotel information for settlement. There can be only one hotel element.

└<checkIn>	Yes	DATE			Check in date for hotel.
└<checkOut>	Yes	DATE			Check out date for hotel.
└<numberInparty>	Yes	0-9	3 chars		Number of people in the hotel booking.
└<folioRefNumber>	No	Aa 0-9 +	10 chars		Folio reference number for hotel.
└<confirmedReservation>	No	Aa		Y N	Flag to indicate whether a guest has confirmed their reservation **Y**= Confirmed Reservation **N** = Unconfirmed Reservation
└<dailyRoomRate>	Yes	0-9 . Aa	15 chars		Daily room rate for the hotel.
└<guestName>	Yes	Aa 0-9 +	20 chars		Name of guest
└<cruise>	No				Used to provide cruise information for settlement. There can be only one cruise element.
└<checkIn>	Yes	DATE			Start date for cruise.
└<checkOut>	Yes	DATE			End date for cruise.
└<cardRental>	No				Used to provide car rental information for settlement. There can be only one car rental element.
└<checkIn>	Yes	DATE			Check in date for car rental.
└<checkOut>	Yes	DATE			Check out date for car rental.
└<tourOperator>	No				Used to provide tour operator information for settlement. There can be only one tour operator element.
└<checkIn>	Yes	DATE			Check in date for tour operator.
└<checkOut>	Yes	DATE			Check out date for tour operator.
└<airline>	No				Used to provide airline information for settlement. There can be only one airline element
└<ticketNumber>	Yes	Aa 0-9	11 chars		The airline ticket number
└<airlineCode>	Yes	0-9	3 chars		IATA airline code
└<agentCode>	Yes	0-9	8 chars		IATA agent code
└<agentName>	Yes	Aa 0-9	26 chars		Agency name
└<flightNumber>	No	Aa 0-9	6 chars		Flight number
└<restrictedTicket>	Yes	BOOLEAN			Can be 0, 1, true or false.

└<passengerName>	Yes	Aa 0-9	29 chars		Name of passenger
└<orginatingAirport>	Yes	Aa	3 chars		IATA airport code
└	Yes				Contains other elements detailing the segment At least one segment element must be supplied under the airline element, but can supply up to 4 segments.
└<carrierCode>	Yes	Aa	3 chars		IATA carrier code
└<class>	Yes	Aa 0-9	3 chars		Class of service
└<stopover>	Yes	BOOLEAN			Can be 0,1, true or false to indicate a stopover
└<legDepartureDate>	Yes	DATE			Departure date of the segment.
└<destination>	Yes	Aa	3 chars		IATA airport code of destination
└<fareBasis>	No	Aa 0-9	6 chars		Fare basis code
└<customerCode>	No	Aa 0-9	20 chars		Airline customer code
└<invoiceNumber>	No	Aa 0-9	15 chars		Airline Invoice Number
└<dinerCustomerRef>	No	Aa 0-9	15 chars		Diners customer reference Can include up to 5 elements

View example Basket XML snippets on sagepay.com

A1.4 CustomerXML

The extra fields detailed below can be passed as an xml document for more accurate fraud screening. The XML tags should follow the order stated in the table.

Customer XML elements

Node/Element	Mandatory	Format	Max Length	Allowed Values	Description
<customer>	No	Node			The root element for all other customer elements.
└<customerMiddleInitial>	No	Aa	1 char		The middle initial of the customer.
└<customerBirth>	No	DATE	19 chars		The date of birth of the customer.
└<customerWorkPhone>	No	0-9 - Aa + ( )	19 chars		The work phone number of the customer.
└<customerMobilePhone>	No	0-9 - Aa + ( )			The mobile number of the customer.
└<previousCust>	No	BOOLEAN			Whether the customer is a previous customer or new.
└<timeOnFile>	No	0-9 + -	16 chars	Min Value 0	The number of days since the card was first seen.
└<customerId>	No	Aa 0-9	1 char		The ID of the customer

View example Customer XML snippets on sagepay.com

A2. Server response to the transaction registration POST

This is the plain text response part of the POST originated by your servers in A1. Encoding will be as Name=Value pairs separated by carriage return and linefeeds (CRLF).

Response format

Name	Mandatory	Format	Max Length	Allowed Values	Description
VPSProtocol	Yes	0-9 .	4 chars	**3.00**	Protocol version used by the system. Same as supplied in A1.
Status	Yes	Aa	15 chars	**OK** **OK REPEATED** **MALFORMED** **INVALID** **ERROR**	If the Status is not **OK**, the StatusDetail field will give more information about the problem. **OK =** Process executed without error. **OK REPEATED =** If the VendorTxCode passed in A1 has been used before, but that transaction is still active, then details of that transaction are passed back in this POST and the suffix **REPEATED** is appended to the Status. Your system must be able to handle **REPEATED** messages from Sage Pay. **MALFORMED =** Input message was missing fields or badly formatted – normally will only occur during development. **INVALID =** Transaction was not registered because although the POST format was valid, some information supplied was invalid. E.g. incorrect vendor name or currency. **ERROR =** A problem occurred at Sage Pay which prevented transaction registration. Please notify Sage Pay if a Status of **ERROR** is seen, together with your Vendor, VendorTxCode and the StatusDetail.

StatusDetail	Yes	Aa 0-9 . _ ( ) , :	255 chars		Human-readable text providing extra detail for the Status message. Always check StatusDetail if the Status is not **OK**
VPSTxId	Yes	Aa 0-9 - { }	38 chars		The Sage Pay ID to uniquely identify the transaction on our system. Only present if Status is **OK** or **OK REPEATED**.
SecurityKey	Yes	Aa 0-9	10 chars		A Security key which Sage Pay uses to generate a MD5 Hash for to sign the Notification message (B3 below). The signature is called VPSSignature. This value is used to allow detection of tampering with notifications from the Sage Pay gateway. It must be kept secret from the customer and held in your database. Only present if Status is **OK** or **OK REPEATED**.
NextURL	Yes	RFC1738	255 chars		This is the URL to which the Vendor must redirect the Customer to continue the transaction. Only present if Status is **OK** or **OK REPEATED**. Note that the full URL must be used for the redirect, including any appended parameters.

Appendix B: Notification of Transaction Results

B1. Sage Pay Notification POST

The Sage Pay Server will send notification in the request part of a POST to the Notification URL provided in A1. The request will be URL encoded, with Name=Value fields separated by '&' characters.

Request format

Name	Mandatory	Format	Max Length	Allowed Values	Description
VPSProtocol	Yes	0-9 .	4 chars	**3.00**	Same as supplied in A1
TxType	Yes	Aa	15 chars	**PAYMENT** **DEFERRED** **AUTHENTICATE**	Same as supplied in A1
VendorTxCode	Yes	Aa 0-9 { } . _ -	40 chars		Same as supplied in A1
Status	Yes	Aa	15 chars	**OK** **NOTAUTHED** **PENDING** **ABORT** **REJECTED** **AUTHENTICATED** **REGISTERED** **ERROR**	If the Status is not **OK**, the StatusDetail field will give more information about the problem. **OK =** Process executed without error. **NOTAUTHED =** The Sage Pay gateway could not authorise the transaction because the details provided by the customer were incorrect, or insufficient funds were available. However the transaction has completed. **PENDING =** This only affects European Payment methods. Indicates a transaction has yet to fail or succeed. This will be updated by Sage Pay when we receive a notification from PPRO. The updated status can be seen in MySagePay. **ABORT =** The Transaction could not be completed because the user clicked the CANCEL button on the payment pages, or went inactive for 15 minutes or longer. **REJECTED =** The Sage Pay System rejected the transaction

					because of the fraud screening rules you have set on your account. Note: The bank may have authorised the transaction but your own rule bases for AVS/CV2 or 3D-Secure caused the transaction to be rejected. **AUTHENTICATED =** The 3D-Secure checks were performed successfully and the card details secured at Sage Pay. Only returned if `TxType` is **AUTHENTICATE**. **REGISTERED =** 3D-Secure checks failed or were not performed, but the card details are still secured at Sage Pay. Only returned if `TxType` is AUTHENTICATE. **ERROR =** A problem occurred at Sage Pay which prevented transaction registration. Please notify Sage Pay if a `Status` of **ERROR** is seen, together with your `Vendor`, `VendorTxCode` and the `StatusDetail`.
`StatusDetail`	Yes		255 chars		Human-readable text providing extra detail for the `Status` message. Always check `StatusDetail` if the `Status` is not **OK**
`TxAuthNo`	No		10 chars		Sage Pay unique Authorisation Code for a successfully authorised transaction. Only present if `Status` is **OK**.
`AVSCV2`	Yes		50 chars	**ALLMATCH** **SECURITY CODE MATCH ONLY** **ADDRESS MATCH ONLY** **NO DATA MATCHES** **DATA NOT CHECKED**	This is the response from AVS and CV2 checks. Provided for Vendor info and backward compatibility with the banks. Rules set up in MySagePay will accept or reject the transaction based on these values. More detailed results are split out in the next three fields. Not present if the Status is **AUTHENTICATED** or **REGISTERED.**
`AddressResult`	Yes		20 chars	**NOTPROVIDED** **NOTCHECKED** **MATCHED**	The specific result of the checks on the cardholder's address numeric from the AVS/CV2 checks. Not present if the `Status` is **AUTHENTICATED** or **REGISTERED**

				NOTMATCHED	
PostCodeResult	Yes	Aa	20 chars	**NOTPROVIDED** **NOTCHECKED** **MATCHED** **NOTMATCHED**	The specific result of the checks on the cardholder's Postcode from the AVS/CV2 checks. Not present if the Status is **AUTHENTICATED** or **REGISTERED**
CV2Result	Yes	Aa	20 chars	**NOTPROVIDED** **NOTCHECKED** **MATCHED** **NOTMATCHED**	The specific result of the checks on the cardholder's CV2 code from the AVS/CV2 checks. Not present if the Status is **AUTHENTICATED** or **REGISTERED**
GiftAid	Yes	BOOLEAN	1 char	**0** **1**	This field is always present even if GiftAid is not active on your account. **0** = The Gift Aid box was not checked this transaction. **1** = The customer checked the Gift Aid box on the payment page

3DSecureStatus	Yes	Aa	50 chars	**OK** **NOTCHECKED** **NOTAVAILABLE** **NOTAUTHED** **INCOMPLETE** **ATTEMPTONLY** **ERROR**	This field details the results of the 3D-Secure checks (where appropriate) **OK** - 3D-Secure checks carried out and user authenticated correctly. **NOTCHECKED** – 3D-Secure checks were not performed. This indicates that 3D-Secure was either switched off at an account level, or disabled at transaction registration. **NOTAVAILABLE** – The card used was either not part of the 3D -Secure Scheme, or the authorisation was not possible. **NOTAUTHED** – 3D-Secure authentication checked, but the user failed the authentication. **INCOMPLETE** – 3D-Secure authentication was unable to complete. No authentication occurred. **ATTEMPTONLY**– 3D-Secure attempted but cardholder was not enrolled. **ERROR** - Authentication could not be attempted due to data errors or service unavailability in one of the parties involved in the check.
CAVV	No	Aa 0-9	32 chars		The encoded result code from the 3D-Secure checks (CAVV or UCAF). Only present if the 3DSecureStatus field is **OK** or **ATTEMPTONLY**
AddressStatus	Yes	Aa	20 chars	**NONE** **CONFIRMED** **UNCONFIRMED**	**PayPal Transactions Only.** If AddressStatus is **CONFIRMED** and PayerStatus is **VERIFIED**, the transaction may be eligible for PayPal Seller Protection. To learn more about PayPal Seller Protection, please contact PayPal directly or visit paypal.com
PayerStatus	Yes	Aa	20 chars	**VERIFIED** **UNVERIFIED**	

CardType	Yes	Aa	15 chars	**VISA** **MC** **MCDEBIT** **DELTA** **MAESTRO** **UKE** **AMEX** **DC** **JCB** **PAYPAL**	**VISA** is Visa **MC** is MasterCard **MCDEBIT** is Debit MasterCard **DELTA** is Visa Debit **MAESTRO** is Domestic and International issued Maestro **UKE** is Visa Electron **AMEX** is American Express **DC** is Diners Club International and Discover **JCB** is Japan Credit Bureau PAYPAL
Last4Digits	Yes	0-9	4 chars		The last 4 digits of the card number used in this transaction. PayPal transactions have 0000 This field is supplied to allow merchants using wallet systems to identify the card to their customers
VPSSignature	Yes	Aa 0-9	100 chars	MD5 signature of the concatenation of the values of: {VPSTxId }+ VendorTxCode + Status + TxAuthNo + VendorName + AVSCV2 + SecurityKey + AddressResult + PostCodeResult + CV2Result + GiftAid + 3DSecureStatus + CAVV + AddressStatus + PayerStatus + CardType + Last4Digits + DeclineCode + ExpiryDate + FraudResponse + BankAuthCode	To detect any possible tampering with messages, your site should compute the same MD5 signature (which incorporates the SecurityKey provided in A2) and check it against VPSSignature. You can then decide what to do with transactions that appear to have been tampered with. MD5 value is returned in UPPER CASE. If a field is returned without a value this should not be included in the string. Please ensure the VendorName is lower case prior to hashing.

FraudResponse	No	Aa	10 chars	**ACCEPT** **CHALLENGE** **DENY** **NOTCHECKED**	**ACCEPT** means ReD recommends that the transaction is accepted **DENY** means ReD recommends that the transaction is rejected **CHALLENGE** means ReD recommends that the transaction is reviewed. You have elected to have these transactions either automatically accepted or automatically denied at a vendor level. Please contact Sage Pay if you wish to change the behaviour you require for these transactions **NOTCHECKED** means ReD did not perform any fraud checking for this particular transaction
Surcharge	No	0-9 . ,		0.01 to 100,000.00	Returns the surcharge amount charged and is only present if a surcharge was applied to the transaction.
DeclineCode	No	0-9	2 chars		The decline code from the bank. These codes are specific to the bank. Please contact them for a description of each code. e.g. 00
ExpiryDate	Yes	0-9	4 chars		Expiry date of the card used, in the format MMYY.
BankAuthCode	No	Aa 0-9	6 chars		The authorisation code returned from the bank. e.g T99777
Token	No	Aa 0-9 - {}	38 chars		The token generated by Sage Pay.

B2. You acknowledge receipt of the Notification POST

This is the plain text response part of the POST originated by the Server in the step above. Encoding must be as Name=Value fields separated by carriage-return-linefeeds (CRLF).

Response format

Name	Mandatory	Format	Max Length	Allowed Values	Description
`Status`	Yes	Aa	15 chars	**OK** **INVALID** **ERROR**	**OK =** Send this if you successfully received the Notification Post in B1 and were able to match the `VPSSignature`. **INVALID =** Send this if the details you received in the A3 post were inconsistent with expectations for this transaction. The `RedirectURL` must still be provided, and Sage Pay will still redirect the customer back to your site, but the transaction will NOT be settled with the bank. Only send this result if you want to cancel the transaction. **ERROR =** An error has occurred during your Notification processing. The Sage Pay system will check for a `RedirectURL`, and if one is provided the Customer will be redirected to your site, but the transaction will NOT be settled with the bank. Only send this result if you want to cancel the transaction and report an **ERROR** to Sage Pay.
`RedirectURL`	Yes	RFC1738	255 chars		Fully qualified URL (including http:// or https:// header) to which you'd like the customer redirected on completion of the transaction. If you wish to pass parameters back to your own site (such as the session id or transaction code), these should be included in `RedirectURL`.
`StatusDetail`	No	Aa 0-9 . ( ) , :	255 chars		Human-readable text providing extra detail for the `Status` message.

Before writing the three fields above to the Response object of the POST, please ensure you clear your response buffer to remove any header code, comments or HTML. The Sage Pay Server is expecting “Status=” to be the first characters in the response. If it does not see these, it treats the response as though it is an error and fails the transaction. All POSTs must be communicated through ports 80 and 443.

14.0 URLs

The table below shows the complete set of web addresses (URLs) to which you send the transaction registration post.

Environment	URL
TEST	https://test.sagepay.com/gateway/service/vspserver-register.vsp
LIVE	https://live.sagepay.com/gateway/service/vspserver-register.vsp

Please ensure that your firewalls allow outbound Port 443 (HTTPS only!) and inbound ports 443 (and optionally 80 HTTP) access in order to communicate with our servers (on Test and Live).