Sage Pay Direct
Integration and Protocol Guidelines 3.00

Published: 01/08/2014

Table of Contents

Document Details 4
Version History 4
Legal Notice 4

1.0 Introduction 5

2.0 Overview of Direct Integration 6
2.1 Non 3D-Secure transactions 6
2.2 3D-Secure transactions 7
2.3 Direct and PayPal 8

3.0 Direct Integration in Detail (none PayPal) 9
Step 1: The customer orders from your site 9
Step 2: Your server registers the payment with Sage Pay 10
Step 3: Sage Pay checks for 3D-Secure enrolment 12
Step 4: Sage Pay replies to your registration POST 13
Step 5: You redirect your customer to their Issuing Bank 14
Step 6: 3D-Authenticaiton and your site called back 16
Step 7: Your site POSTs the 3D-Secure results to Sage Pay 17
Step 8: Sage Pay servers request card authorisation 18
Step 9: Sage Pay reply to your POST 19
Step 10: Sage Pay sends Settlement Batch Files 20

4.0 Direct Integration (PayPal) 21
4.1 Direct PayPal Message Flow 23

5.0 Integrating with Sage Pay Direct 24

6.0 Testing on the Test Server (Stage 1) 25
6.1 Registering a Payment 25
6.1.1 3D-Authenticated Transactions 26
6.1.2 Test card numbers 26
6.2 Direct PayPal transactions 29
6.3 Accessing MySagePay on Test 31
6.4 Refunding a transaction 33

7.0 Additional Transaction Types 34
7.1 DEFERRED transactions 34
7.2 REPEAT payments 35
7.3 AUTHENTICATE and AUTHORISE 35
7.4 REFUNDs and VOIDs 36

8.0 Applying Surcharges 37

9.0 Sage 50 Accounts Software Integration 38

10.0 Going Live (Stage 2) 39

11.0 Congratulations, you are live with Sage Pay Direct 40

12.0 Character Sets and Encoding 41

Appendix A: Direct Protocol 42
A1. You submit your transaction registration POST 42
A1.1 SurchargeXML 51
A1.2 Basket 52
A1.3 BasketXML 53
A1.4 CustomerXML 59
A2. Sage Pay response to the Transaction Registration or Callback POSTs 60
A3. Sage Pay response to the transaction registration POST (3D-Secure) 65
A4. 3D-Authentication Results POST from your Terminal URL to Sage Pay (3D-Secure) 66
A5. Sage Pay response to the Transaction Registration POST (PayPal) 67
A6. Sage Pay Callback after PayPal Authentication (PayPal) 68
A7. Complete a PayPal Transaction (PayPal) 70

13.0 URLs 71

Document Details

Version History

Date	Change	Page
19/07/2013	Document published.	---
	Added Expiry Date as a returned field.	64
	Basket XML includes Discounts.	56
	Allowed characters in BankAuthCode now Alphanumeric.	64
16/09/2013	New screenshots.	---
	References to Sage Pay website updated.	---
	European Payment Information updated.	---
	Removed reference to Laser Cards.	---
	Surcharge XML clearer.	37
	Added StoreToken field.	49
01/08/2014	Rebranded.	---
	Included additional fields for Financial Institutions (MCC 6012).	50
	Information on pre-authorisations.	34
	Sage Software.	38
	3D-Secure simulation.	27
	XML snippets moved to sagepay.com	---
	Updated Test Cards.	27
	Added PPro / PayPal indicators.	---
	Basket XML Amendments.	53

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 Direct integration.

Sage Pay's Direct integration provides a secure, simple means of authorising credit and debit card transactions from your website.

Sage Pay's Direct is designed to enable you to take payment on your own secure servers and pass them across to us for authorisation and secure storage in a server-to-server session that does not involve redirecting the customer to the Sage Pay hosted pages. This enables you to white-label the payment process. Your customer never leaves your site (unless you are using PayPal or the 3D-Secure authentication processes) and they do not necessarily know that Sage Pay is authorising the transaction on your behalf (although in practice many merchants choose to tell their customers in case they have concerns about card data security).

To use the Direct method you will need a 128-bit SSL certificate to secure your payment pages. These can be obtained from a number of sources, including VeriSign. You will also need to be able to make HTTPS POSTs from scripts on your server (using something like OpenSSL on Linux platforms, or the WinHTTP object in Win32). If you are hosting with a third party company we recommend you talk to them about these requirements before committing to use Direct. If you cannot install a certificate for your payment pages, we would recommend using the Sage Pay Server integration instead. If you cannot perform HTTPS POSTs from your scripts, we would recommend the Sage Pay Form integration.

If you wish to support 3D-Secure (Verified by Visa, MasterCard SecureCode and Amex Safe), Direct provides a wrapper for these systems, removing the need for you to purchase and support your own Merchant Plug-In. All the messages will be created for you, and you'll simply need to redirect your customer to their issuing bank, and then send on the results of their 3D-Authentication back to Sage Pay to complete the payment process. Just like non 3D-Secure Direct transactions, the customer is never directed to Sage Pay. They leave your site to authenticate with their bank and then return to your site when they have finished.

This document explains how your Web servers communicate with Sage Pay using the Direct method, and explains how to integrate with our test and live environments. It also contains the complete Payment Protocol in the Appendix.

Since card data will be collected via your site, you will be obliged to comply with the Payment Card Industry Data Security Standard (PCI-DSS). We have been working with our own data security partner, Trustwave, to set up a program for Sage Pay customers to make PCI DSS compliance easy and cost effective. For further information please visit sagepay.com.

Indicates additional information specific to European Payment method transactions.

Indicates additional information specific to PayPal transactions.

2.0 Overview of Direct Integration

Direct payment requests are very simple. The interaction with your customer is entirely yours. The customer will select items or services to purchase and fill up a shopping basket. When they are ready to pay, you will first collect their name, billing and delivery address, contact details (telephone number, email address and so forth) and perhaps allow them to sign up for quicker purchases in future. You will total the contents of the basket and summarise its contents for them before asking them to continue.

Your scripts should then store everything about the transaction and customer in your database for future reference. You will not need to store any card details because Sage Pay will hold those securely for you.

You will then present your customers with a payment page, secured with your 128-bit SSL certificate. This page will ask the customer for:

- The Cardholder Name as it appears on the card
- The Card Type (Visa, MasterCard, American Express etc.)
- The full Card Number without spaces or other separators
- The Expiry Date
- The Card Verification Value (called CVV or CV2 value. The extra three digits on the signature strip for most cards, or the 4 numbers printed on the front of an American Express card).
- The Cardholder's Billing Address, including the Postcode (if you have not already asked for it and stored it in your database).

This page is submitted to a script on your server that retrieves and pre-validates those values (checking all fields are present, expiry dates are not in the past, the card number field only contains numbers etc.) before constructing an HTTPS POST containing your own unique reference to the transaction, the `VendorTxCode` (which should be stored alongside the order details in your database) and the correctly formatted data from your form. This HTTPS POST is sent to the Sage Pay gateway.

2.1 Non 3D-Secure transactions

Sage Pay validates the data sent to us, checking that it has come from a valid source and that all required information is present, before creating a transaction in our database to securely hold all the data passed to us, contacting the bank for authorisation and replying to you, in real-time, in the response part of the same HTTPS POST. In practice this takes about 2-3 seconds to complete.

The same script on your server that initiated the POST simply reads the Response from that POST to determine whether the transaction was authorised or not. It then updates your database with transaction reference values and the authorisation code (where appropriate) before displaying either a completion page to your customer, or an error page explaining why the payment was not accepted.

Your own database will contain all the necessary information about the transaction, the basket contents and the customer, but you will NOT need to store the card details because the transaction IDs passed to you by the Direct system will enable you to perform all other actions against that card (refunds, additional payments, cancellations and so on). This allows you to be certain that even if your server is compromised, no card details can be gleaned from your database.

The following sections explain the integration process in more detail. The Direct Payment protocol is attached in the Appendix, providing a detailed breakdown of the contents of the HTTPS message sent between your servers and ours during a payment.

A companion document 'Server and Direct Shared Protocol' gives details of how to perform other transaction-related POSTs, such as **REFUND**s, **REPEAT** payments and the **RELEASE** / **ABORT** mechanisms for **DEFERRED** transactions.

2.2 3D-Secure transactions

Direct payments with 3D-Authentication are a little more complicated because your customer has to be forwarded to their card issuer to authenticate themselves before a card authorisation can occur. You must have 3D-Secure active on your account before you can process this type of transaction. Contact support@sagepay.com for more information about setting this up (Depending on your Merchant Acquirer your account maybe set up with 3D-secure by default. If this is the case you'll just need to enable it in your MySagePay admin area. This will be covered later.)

The process of obtaining a 3D-Secured authorisation begins in the same manner as non-authenticated transactions. Your customer fills up a shopping basket on your site, you collect their details, then present them with a payment page secured with your 128-bit SSL certificate. This page POSTs to a script on your site which pre-validates the data and formats a normal server-side Direct Transaction Registration POST (see Appendix A1) which is sent to Sage Pay.

As in a non-authenticated Direct transaction, the information you POST to us is validated against your IP address list and the data checked for range errors, but if everything appears in order, rather than immediately sending the card details to your acquiring bank for authorisation, the details are instead used to send a query to the 3D-Secure directory servers. These check to see if the card and the card-issuer are enrolled in the 3D-Secure scheme.

If the card or the issuer is NOT part of the scheme, Direct checks your 3D-Secure rule base (which you can modify in our MySagePay screens) to determine if you wish to proceed with the authorisation in such circumstances. If the card or the issuer is not part of the scheme and your rule base allows authorisation to proceed, the card details are sent to the acquiring bank and the results of that process returned to your site in the Response object of your POST (just like a non-3D-authenticated Direct transaction, but with an additional `3DSecureStatus` field informing you about the results of the card lookup).

If authorisation cannot proceed because your rules do not allow it, a **REJECTED** message is sent back in the Response object of your POST, outlining the reason for the transaction rejection.

If, however, the card and issuer are part of the 3D-Secure scheme, Direct does not attempt to obtain an authorisation from your acquiring bank. Instead it formats and encrypts a 3D-Secure request message called a `PAReq` and replies to your Direct POST in the Response object with this message, a unique transaction code called the `MD`, and the URL of the 3D-Secure authentication pages at the cardholder's Issuing Bank (in a field called `ACSURL`). You can store the `MD` value if you wish to, but the `ACSURL` and `PAReq` values should NEVER be stored in your database.

Your server creates a simple, automatically-submitted HTML form that POSTs the user, the `MD` and the `PaReq` fields across to the `ACSURL`, along with an additional field called the `TermUrl` which points to a page on your site to which the bank will return the customer when they have been authenticated.

From the user's perspective, they will have entered their card details on your payment page, clicked submit, and will find themselves transferred to their card issuer to validate their 3D-Secure credentials.

Once the user has completed their 3D-authentication, their Issuing Bank will redirect the customer back to a script on your site pointed to by the `TermUrl`. The user returns to your site along with the `MD` of the transaction and the results of their authentication in an encrypted field called the `PaRes`. Like before, Direct takes care of decrypting and decoding this information for you, so your `TermUrl` page simply needs to format a server side HTTPS POST containing the `MD` and the `PaRes` fields (all correctly URL Encoded) and send it to Direct. You do not need to store the `MD` or `PaRes` fields in your database.

Direct examines the `PaRes` to determine if authentication was successful. If it was, it retrieves all the details from your original Direct POST and goes on to obtain an authorisation from your acquiring bank. It then replies with the results in the Response object of your `TermUrl` POST in the same format as a non-3D Secured transaction, but with two additional fields for you to store (the `3DSecureStatus` and the `CAVV` value; a unique value which indicates that the Authentication was successful).

If Direct examines your `PaRes` and finds that authentication was NOT successful, it again checks your 3D-Secure rule base to determine if you wish to proceed. Like the original Transaction Registration POST, if you wish to obtain authorisations for non-3D-authenticated transactions, Direct requests an authorisation from your acquiring bank and replies as normal; if not, Direct returns a **REJECTED** message and does not obtain an authorisation.

Your `TermUrl` should update your database with the results of the authorisation (or lack thereof) and display a completion page to your customer.

Although more complex than a non-3D-authenticated Direct transaction, this process does remove a huge amount of the complexity involved in using your own Merchant Plug-In. Moreover, transactions which fully authenticate offer you the protection of a liability shift for card-related misuse, which is extremely valuable if you sell products or services that are likely to attract fraud.

2.3 Direct and PayPal

Sage Pay has integrated with PayPal Express Checkout, giving you the opportunity to add PayPal as a payment option on your payment pages.

This facility is available to merchants who are a certified PayPal Business Account holder. If you do not already have a PayPal Business Account, you can apply by contacting sales@sagepay.com.

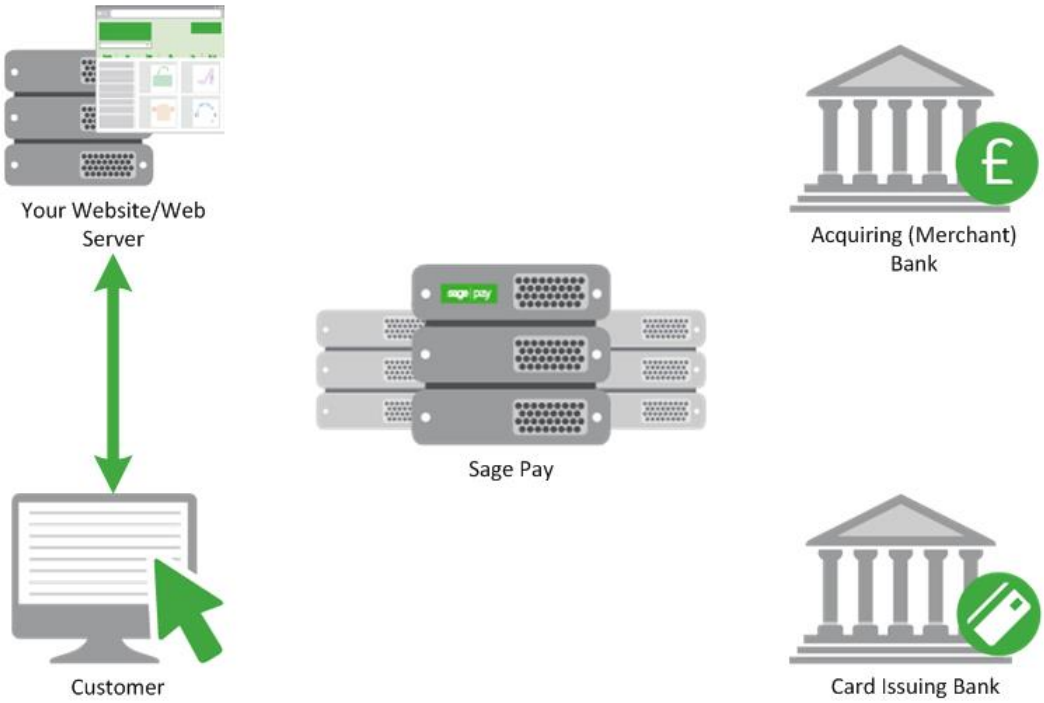
This additional service can be included in your package at no additional cost (standard PayPal transaction fees will apply)
Sage Pay will only charge you our standard transaction rates, according to the Sage Pay package you choose.

To support PayPal Express Checkout using the Direct method involves a little more integration work at your site, but nothing more complex than is currently required for 3D-Authentication.

There is an initial server-to-server POST with Sage Pay, then a redirection to the PayPal logon URL. After that, there is a call back to your servers from Sage Pay, and an additional server-to-server POST to confirm the transaction and complete the process.

3.0 Direct Integration in Detail (none PayPal)

Step 1: The customer orders from your site



A payment begins with the customer ordering goods or services from your site. 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 the Direct system puts no requirement on you to collect any specific set of information at this stage.

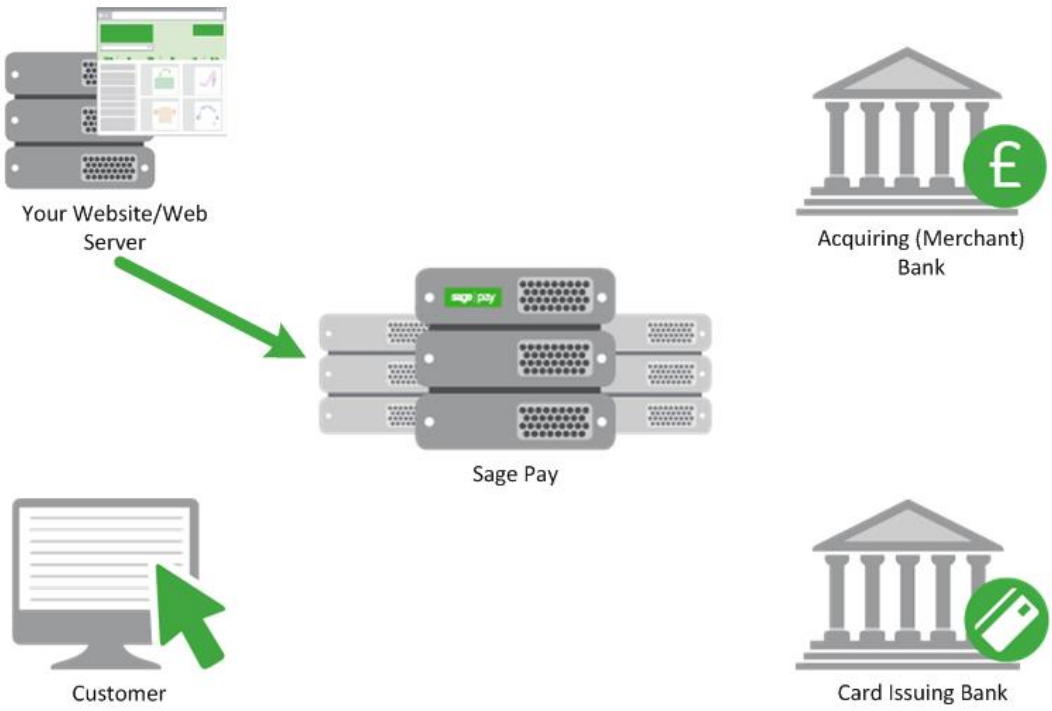
It is generally a good idea to identify the customer by name, email address, delivery and billing address and telephone number. It is also helpful to have your server record the IP Address from which the customer is accessing your system. You should store these details in your database alongside details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

You then present a 128-bit SSL secured payment page into which the customer can enter their card and billing address details. This page should contain the following fields.

- The Cardholder Name as it appears on the card
- The Card Type (**VISA**, **MC**, **MCDEBIT**, **DELTA**, **MAESTRO**, **UKE**, **AMEX**, **DC**, **JCB**)
- The full Card Number without spaces or other separators
- The Expiry Date
- The Card Verification Value (called CVV or CV2 value: The extra three digits on the signature strip for most cards, or the 4 numbers printed on the front of an American Express card).
- The Cardholder's Billing Address, including the Postcode (if you have not already asked for it and stored it in your database).

If you wish to provide a list box for the Expiry Date, please be aware that Visa now issue cards valid for up to 20 years.

Step 2: Your server registers the payment with Sage Pay



Once the customer has decided to proceed, a script on your web server will construct a payment registration message (see Appendix A1) and POST it via HTTPS to the Direct payment URL.

This POST contains your Vendor Name (assigned to you by Sage Pay when your account was created) 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 `Amount` and `Currency` of the payment, and billing and delivery address details for the customer. You can specify a brief `Description` of the goods bought to appear in your reports, plus the entire `Basket` contents if you wish. The card details themselves are passed in dedicated fields whose format can be found in Appendix A1. You can also pass contact numbers and email addresses, flags to bypass or force fraud checking for this transaction and 3D-Secure reference numbers and IDs where such checks have been carried out.

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 Direct method, you can be assured that the information you send us cannot be tampered with or understood by anyone other than us. Your script sends the payment registration message in the Request object of the HTTPS POST and the response from Direct (see Steps 4 and 9 below) is in the Response object of the same POST.

On receipt of the POST, the Sage Pay gateway begins by validating its contents.

It 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 normally only happens during development stage whilst you are refining your integration.

If 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 Direct 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. The `Amount` field is validated. Flag fields are checked, in fact, every field is 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.

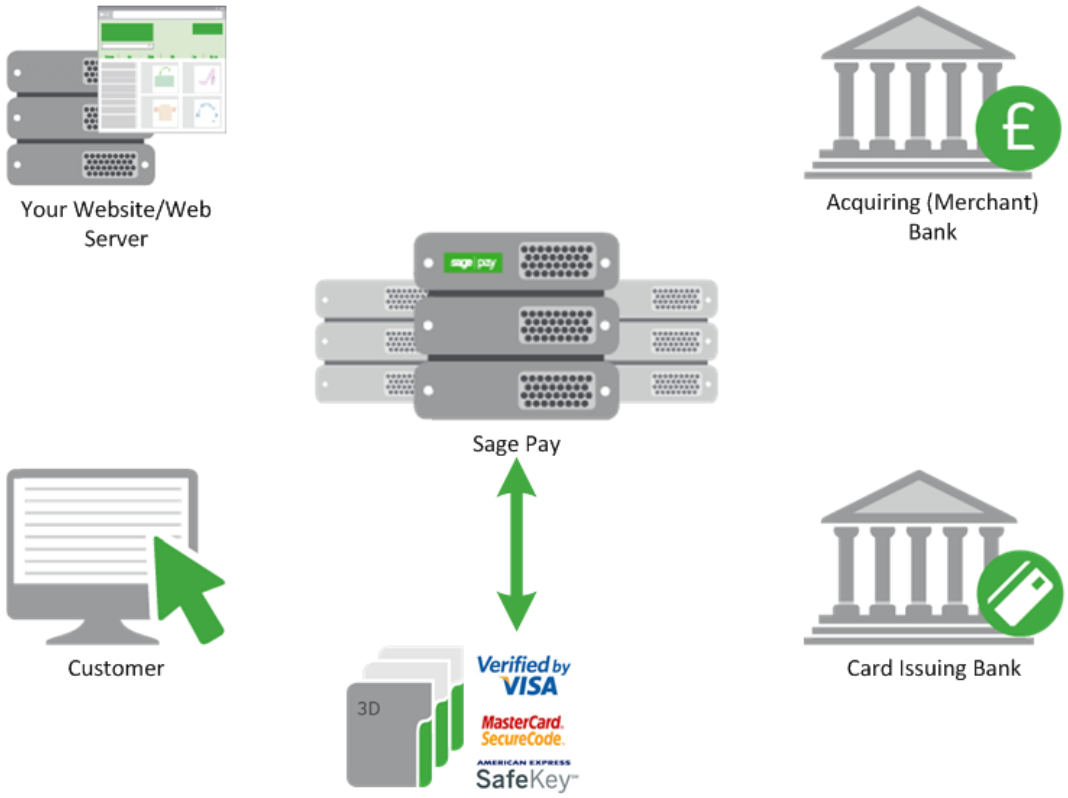
If you receive either a **MALFORMED** or **INVALID** message you should use the detailed response in the `StatusDetail` error message to help debug your scripts. If you receive these messages on your live environment, you should inform your customer that there has been a problem registering their transaction, then flag an error in your back-office systems to help you debug. You can email the Sage Pay Support team (support@sagepay.com) for help with your debugging issues.

The Integration Kits we provide contain scripts in a variety of languages that illustrate how you compose and send this message from your server to ours. These can be downloaded from sagepay.com.

When your transaction is registered with the Sage Pay gateway, a new transaction code is generated that is unique across ALL vendors using the Sage Pay systems, not just unique to you. This code, the `VPSTxId`, is our unique reference to the transaction and is returned to you in the response part of the POST after we've requested authorisation for you. This reference, whilst not the most easily remembered number, will allow us to immediately find your transaction if you have a query about it.

If your Sage Pay account is not set up with 3D-Secure or 3D-Authentication is not active for this transaction, the next step is for the system to obtain an authorisation, so skip ahead to Step 8. If, however, 3D-Secure is active on your account, continue at Step 3.

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



The Sage Pay gateway sends the card details provided in your post 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 the card and card issuer are part of the 3D-Secure scheme.

The servers send a verification response, VERes, back to the Sage Pay MPI where it is decoded and informed of the inclusion or exclusion of the card.

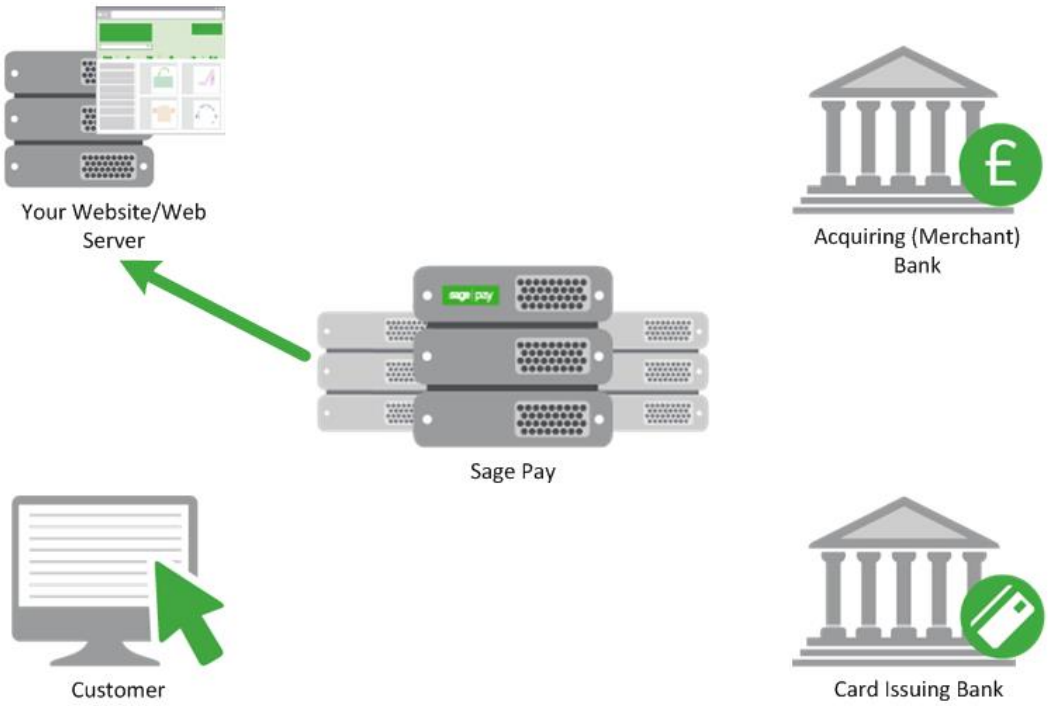
If the card or the issuer is not part of the scheme, or if an MPI error occurs, our server will check your 3D-Secure rule base to determine if authorisation should occur. For information regarding 3D Secure rule bases please refer to the Sage Pay Fraud Prevention Advice Guide, which can be downloaded from sagepay.com. By default your account will not have a rule base established and transactions that cannot be 3D-authenticated will still be forwarded to your acquiring bank for authorisation.

If your rule base rejects the transaction due to your criteria not being reached, the gateway replies with a `Status` of **REJECTED** and a `StatusDetail` indicating why. The `3DSecureStatus` field will contain the results of the 3D-Secure lookup. **REJECTED** transactions will never be authorised and the customer's card never charged, so your code should redirect your customer to an order failure page, explaining why the transaction was aborted.

If your rule base does allow authorisation to occur for non-3D-authenticated transactions, the Sage Pay gateway continues as though 3D-Secure is not active on your account. Jump ahead to Step 8.

If the card and the card issuer are both part of the scheme, the Sage Pay gateway continues with 3D-Authentication by replying to your post with a `Status` of **3DAUTH**.

Step 4: Sage Pay replies to your registration POST



The Sage Pay servers store all the information from your Transaction Registration POST in our secure database before replying (see Appendix A3). The `Status` field will be set to **3DAUTH** with a `StatusDetail` informing you to redirect your customer to their Issuing Bank to complete 3D-Authentication.

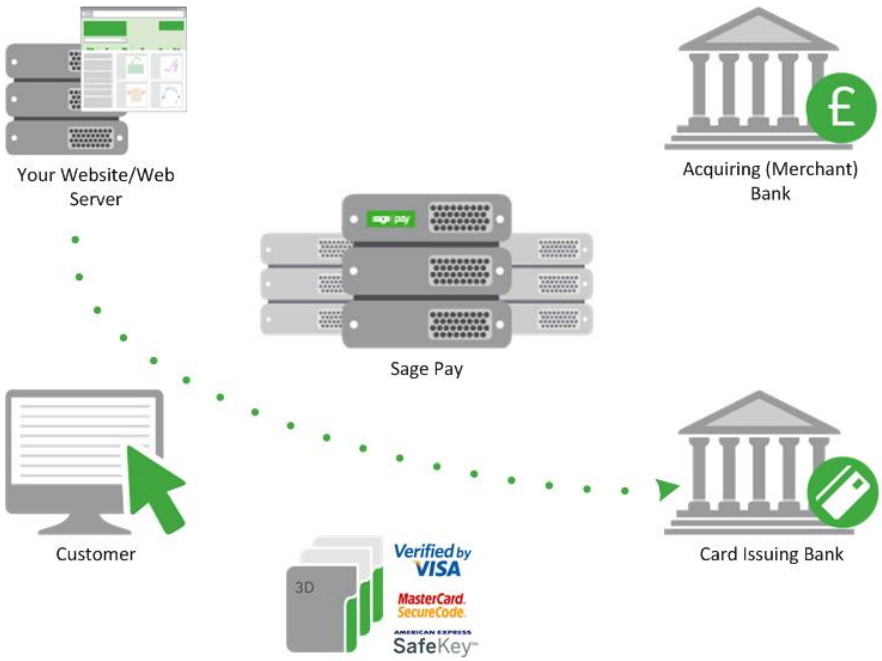
A unique identifier to your transaction called the `MD` is passed along with a preformatted, encrypted field called `PAReq`. This is the 3D-Secure message that the customer's card Issuing Bank decodes to begin the 3D-authentication process. The `PAReq` is created and encrypted by the Sage Pay MPI and you should not attempt to modify it. If you do, the 3D-Secure authentication step will fail and this, in turn, will fail your transaction.

A field called `ACSURL` (Access Control Server URL) contains the fully qualified address of the customer's card Issuing Bank's 3D-Secure module, as provided by the directory service (see Step 3). The last field is the `3DSecureStatus` field, which will always contain **OK** for transactions ready for 3D-authentication.

You do not need to store any of these values in your database. You can store the `MD` value if you wish, but the `ACSURL` and `PAReq` values should NEVER be stored. These values need only be used in the next step to redirect your customer to their Issuing Bank and should then be discarded.

The first step of the Direct transaction is now complete. You have registered a 3D-Secure transaction with Sage Pay; we have stored your payment details and replied with everything you need to send your customer for 3D-Authentication. The next parts of the process, Steps 5 and 6, are out of our control and rely on a communication between you, your customer and your customer's card Issuing Bank.

Step 5: You redirect your customer to their Issuing Bank

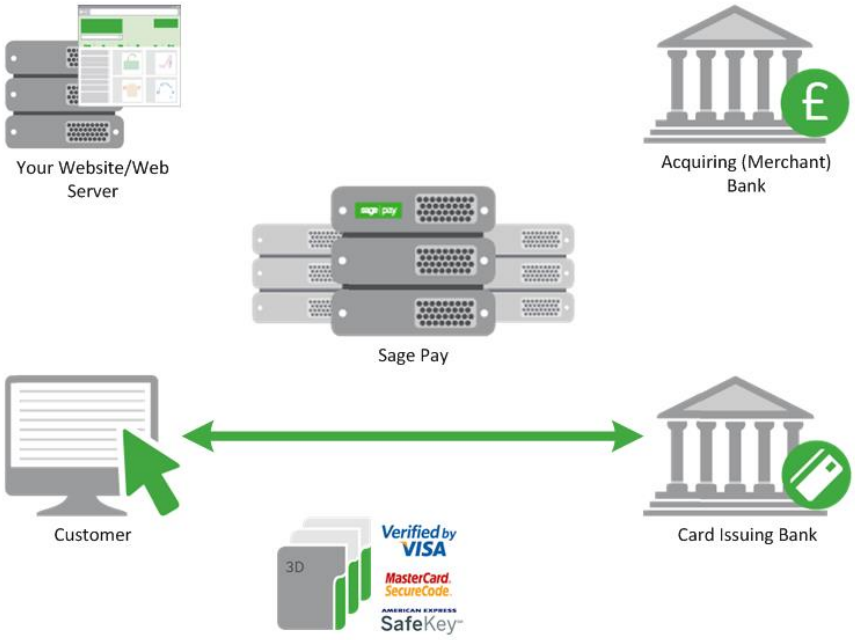


The registration page code on your server should check the `Status` field, and when a **3DAUTH** status is found, build a simple, auto-submitting form (see the example below) which sends the `MD`, `PaReq` and an additional field, the `TermUrl`, to the address specified in the `ACSURL`, and send this form to your customer's browser.

This has the effect of redirecting your customer to their card Issuer's 3D-Authentication site whilst sending to that site all the information required to perform authentication.

The `TermUrl` field is a fully qualified URL which points to the page on your servers to which the customer is sent once the 3D-authentication is completed (see Step 6).

Example code for this page is included in the integration kits provided by Sage Pay; see the example asp code below using an Iframe.



The values in `red` are those extracted from the Sage Pay response and built by your script. If your user has Javascript enabled, they simply redirect to their Issuing Bank site. If not, they will be presented with the message in the NOSCRIPT section and need to click it to go to their Issuing Bank.

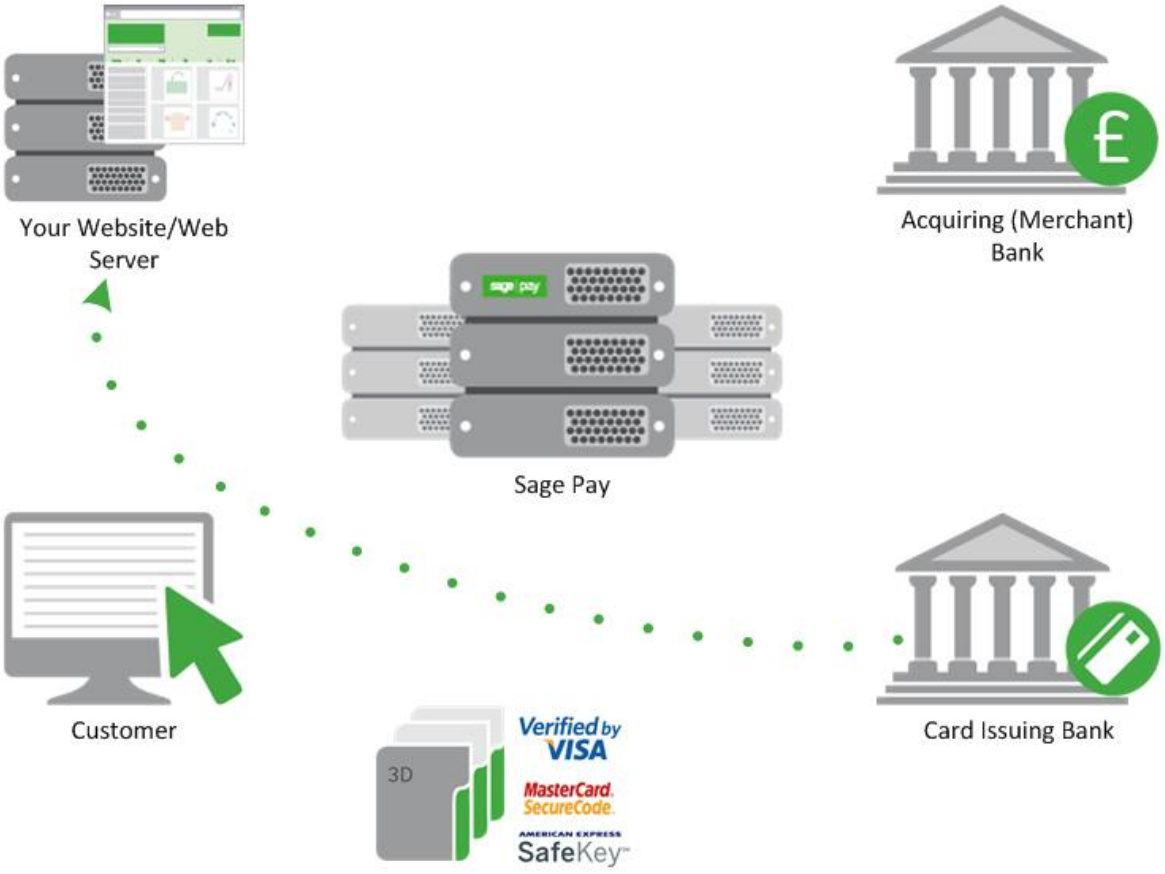
At this stage the customer has left your site, and you must wait for them to be sent back to you by the Issuing Bank.
You can either redirect the customer's entire browser page to their Issuing Bank `ACSURL`, or more commonly, use an inline frame to redirect them. Visa recommend using inline frames for continuity of customer experience, but if you do so, remember to add code to support IFRAME incapable browsers (see below).

```
<IFRAME SRC="3DRedirect.asp" NAME="3DIFrame" WIDTH="100%" HEIGHT="500"
FRAMEBORDER="0">
	<% 'Non-IFRAME browser support
	response.write "<SCRIPT LANGUAGE=""Javascript""> function OnLoadEvent() {
document.form.submit(); }</" & "SCRIPT>"
	response.write "<html><head><title>3D Secure Verification</title></head>"
	response.write "<body OnLoad=""OnLoadEvent();"">"
	response.write "<FORM name=""form"" action=""" & strACSURL &"""
method=""POST"">"
	response.write "<input type=""hidden"" name=""PaReq"" value=""" & strPAReq
&"""/>"
	response.write "<input type=""hidden"" name=""TermUrl"" value=""" &
strYourSiteFQDN & strVirtualDir & "/3DCallback.asp?VendorTxCode=" & strVendorTxCode
& """/>"
	response.write "<input type=""hidden"" name=""MD"" value=""" & strMD &"""/>"

	response.write "<NOSCRIPT>"
	response.write "<center><p>Please click button below to Authenticate your
card</p><input type=""submit"" value=""Go""/></p></center>"
	response.write "</NOSCRIPT>"
	response.write "</form></body></html>"%>
</IFRAME>
```

When you forward the `PAReq` field to the `ACSURL` please ensure you pass the `PAReq` value that we send you, in a field called `PaReq` (note the lower case ‘a’). Many `ACSURL` pages are case sensitive, and will not see the data in your pass an upper case ‘A’.

Step 6: 3D-Authenticaiton and your site called back



Your customer completes the 3D-authentication process at their Issuing Bank's website.

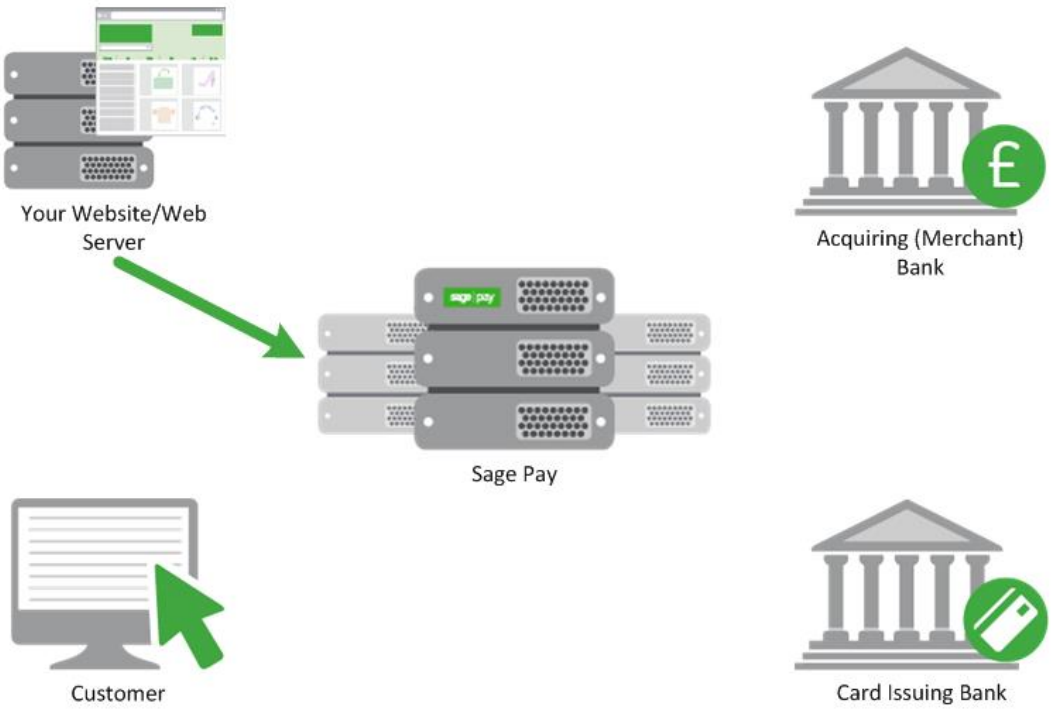
Once complete (either successfully or not), the bank will redirect your customer back to the page supplied in the `TermUrl` field you sent in Step 5.

Along with this redirection, two fields are also sent:
The `MD` value, to uniquely identify the transaction you are being called back about,
The `PaRes`, the encrypted and encoded results of your customer's 3D-authentication.

Like the `PAReq` value sent to your site by Sage Pay in Step 5, you should NOT store the `PaRes` field in your database. Also, because it is strongly encrypted, only the Sage Pay MPI can decode this for you, so you should not attempt to modify it or the authentication process will fail.

At this stage the customer is back on your site and you have completion information for the 3D-Authentication process. You now need to send those through to Sage Pay to decode the results and, where appropriate, obtain a card authorisation from your acquiring bank.

Step 7: Your site POSTs the 3D-Secure results to Sage Pay



The code in your `TermUrl` call-back page should format a simple HTTPS, server-side POST, which it sends to the Sage Pay Direct 3D-Callback page.

This POST needs to contain the `MD` and `PARes` fields sent back to your site by the cardholder's Issuing Bank (suitably URL Encoded for safe transit across the Web).

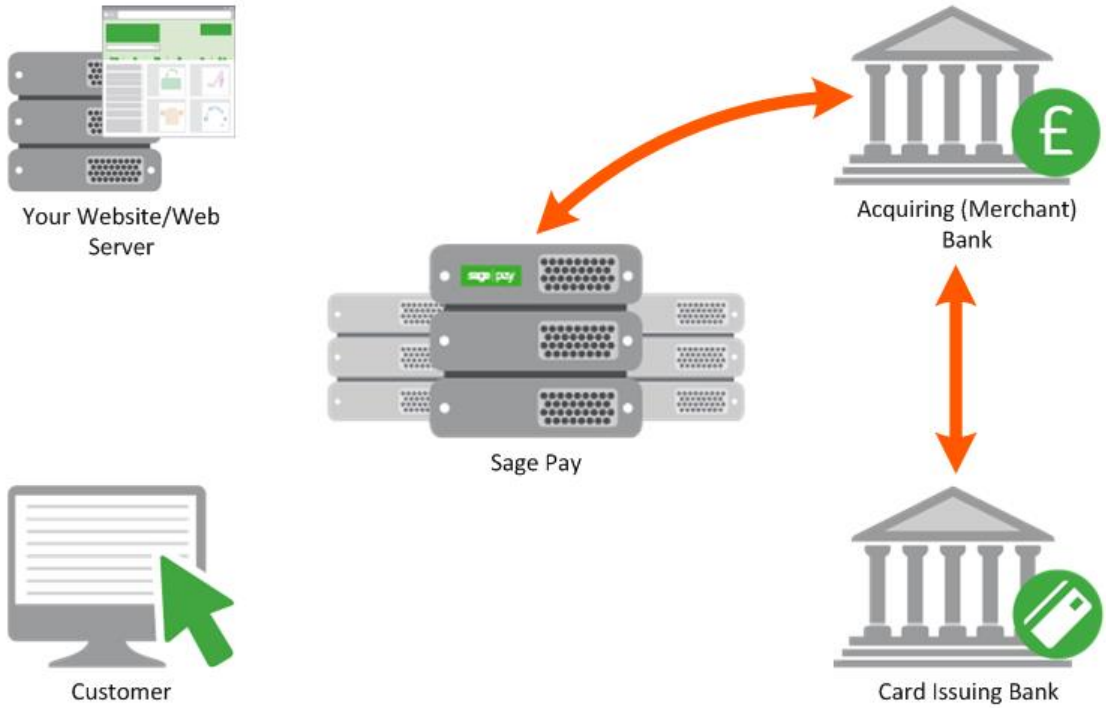
No other information is necessary because the Sage Pay system can use these values to retrieve all the transaction information you originally supplied.

If the decoded `PARes` indicates that the 3D-Authentication was successful, the Sage Pay gateway goes on to obtain an authorisation. If not, the system examines your 3D-Secure rule base to see if authentication should be attempted. By default 3D-Authentication failures are NOT sent for authorisation, but all other message types are. Refer to our Fraud Prevention Guide available on sagepay.com for more information.

Transactions not sent for authorisation are returned with a **REJECTED** `Status`.

Similarly to the note in Step 5, the encrypted and encoded results of your customer's 3D-authentication (the `PARes`) will be returned to you from the Issuing bank in a field called `PaRes` (lower case 'a'), but you must forward this value to Sage Pay in a field called `PARes`.

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.

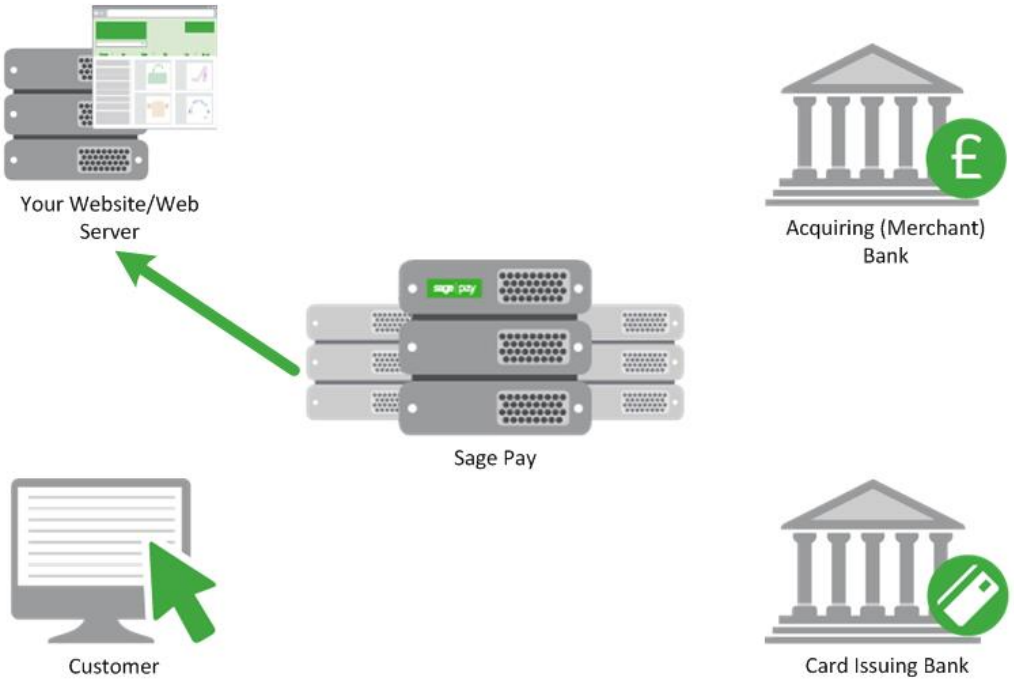
The request is normally answered within a second or so with either an authorisation code, or a declined message. This is obtained directly from the issuing bank by the acquiring bank in real time.

This process happens whilst the script on your server is waiting for a response from our servers. Depending on the response from the acquirer, the Sage Pay gateway prepares either an **OK** response with an authorisation code, a **NOTAUTHED** response if the bank declined the transaction, or an **ERROR** if something has gone wrong (you will very rarely receive these, since they normally indicate an issue with bank connectivity).

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 reverse 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 reply to your POST



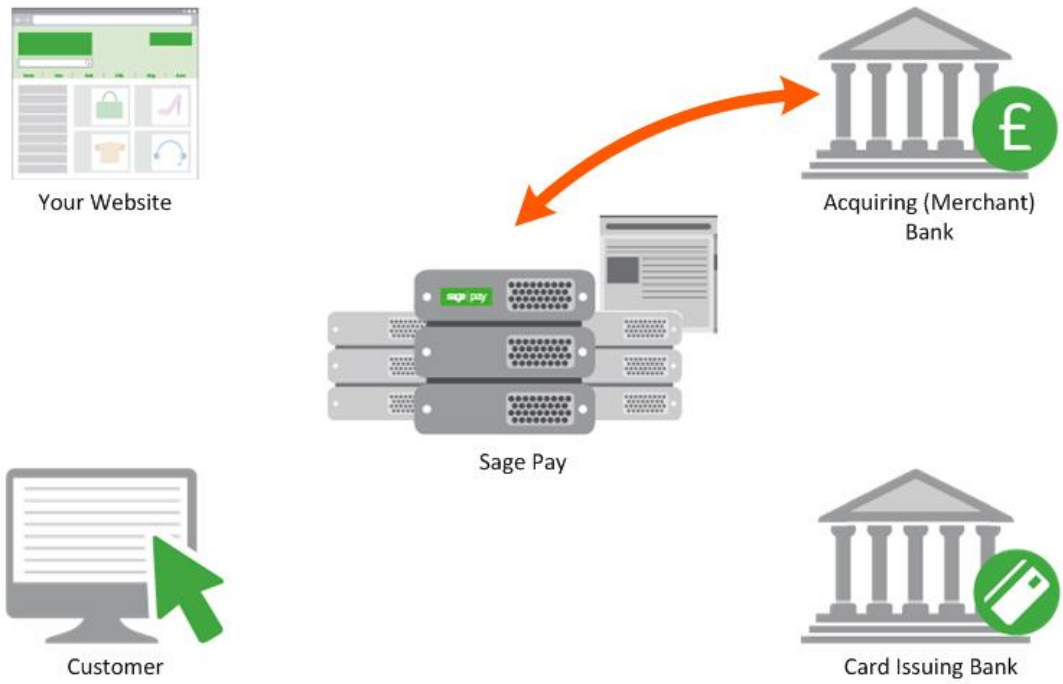
Irrespective of the `Status` being returned, the Sage Pay gateway always replies in the Response section of the POST that your server sent to us. This will either be in response to the Transaction Registration POST for non-3D-authenticated transactions, or in the response to the Terminal URL POST if 3D-Authentication was attempted.

If the transaction was registered successfully, you will always receive the `VPSTxId`, the unique transaction reference mentioned above. You will also receive a `SecurityKey`, a 10-digit alphanumeric code that is used in digitally signing the transaction. Whilst not used in the Direct transaction messages, you do need to know this value if you wish to **REFUND** the transaction, or perform any other automated actions on it using the Sage Pay Direct interface. Therefore this value should be stored alongside the `VPSTxId`, the order details and the `VendorTxCode`, in your database.

If the transaction was authorised and the `Status` field contains **OK**, you will also receive a field called `TxAuthNo`. 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 the transaction ID sent to the bank during settlement (we cannot use your `VendorTxCode` because it is too long and might contain unacceptable characters) so the bank will use this value to refer to your transaction if they need to contact you about it. You should store this value in your database along with all the other values returned to you.
The `TxAuthNo` field is only present if the transaction was authorised by the bank. All other messages are authorisation failures of one type or another (see Appendix A2 for full details of the fields and errors returned) and you should inform your customer that their payment was not accepted.

If you do receive an **OK** `Status` and a `TxAuthNo`, you should display a completion page for your customer thanking them for their order. Having stored the relevant transaction IDs in your database, your payment processing is now complete.

Step 10: 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.

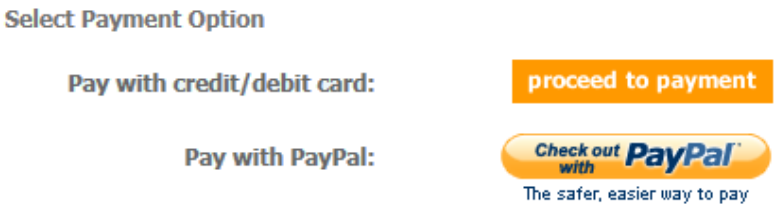
Funds from your customers' PayPal payments are deposited into your PayPal Business account immediately, there is no settlement process. You can then withdraw or transfer the funds electronically into your specified bank account. Although PayPal transactions are included in the Settlement Reports displayed 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.

4.0 Direct Integration (PayPal)

The steps involved in using PayPal with Direct are detailed below and summarised at the end in a diagram.

1. The customer shops at your site and fills up a shopping basket with items.

2. At the point the customer wishes to check-out, BEFORE they enter any address or customer details, your site can optionally allow the customer to select to pay either with PayPal, or another payment process. This is the 'Express Checkout' option and should be presented similar to the example below (available here).



   This is optional as you may wish to offer PayPal as a payment method alongside the card types (after address details have been collected).
   If the customer selects this button, the process jumps to section 6.

3. Since the customer has not selected this button (or has not had the option to do so), your site presents the normal customer detail entry screens, requesting name, email address and billing address in the following format:

   Name (compulsory - 32 chars max)
   Street (compulsory - 100 chars max)
   Street2 (optional - 100 chars max)
   City (compulsory - 40 chars max)
   Zip (compulsory - 20 chars max)
   Country (compulsory – 2 digit ISO 3166-1 code)
   State (compulsory for US Addresses only)
   Phone (optional – 20 characters)

   This structure is required to allow PayPal to validate the addresses against those held in their database.

4. Once the customer has entered their address details, they select their card type, as in a normal Direct payment, with the addition of the PayPal Logo.

   Although still part of the 'Express Checkout' flow, this is referred to as 'Mark' integration. From a Sage Pay perspective, the process is almost identical.

5. If the customer selects a method other than PayPal, then the normal Direct process with 3D-Authentication continues from this point onwards, as detailed in the Direct payment process above, i.e. the customer enters the card number, expiry date, CV2 etc. and the full server-to-server POST is sent.

6. If the customer has selected PayPal, either Mark or Express Checkout, the new process begins at this stage.

   The Direct registration message (see Appendix A1) is sent with the `CardType` field set to **PAYPAL** (no other card details should be sent). Mark implementations will also require the full `Billingxxxx` AND `Deliveryxxxx` sections to be completed as detailed above, but Express Checkouts will leave these empty. This POST also includes a `PayPalCallbackURL` field which points to a script on your site to handle the completion process (explained in stage 11).

7. The information is POSTed to the Direct Transaction Registration URL and the POST is validated as normal. If all fields are validated and the information is correct, the Sage Pay servers construct a message to send to the PayPal servers; for 'Express Checkouts', as you have not collected customer details on your own pages first, a message is sent to ensure the customer enters their address once they reach the PayPal screens. 'Mark' checkouts will already have the address information provided, and therefore the customer will not have the option to select an alternative address once on the PayPal screens.

8. The PayPal servers respond to Sage Pay with a unique token. The transaction is updated in the Sage Pay Database to record this token against the transaction, before returning the Direct response to your servers (Appendix A5).

9. Your site redirects the customer's browser to the `PayPalRedirectURL` value returned in the Direct response (Appendix A5).

10. The customer logs into PayPal and selects their chosen payment method. For Express Checkouts they will also enter/select their delivery address. For Mark, this address selection is disabled.

11. Once the shopper confirms their details on the PayPal screens, PayPal exchange information with Sage Pay, and then Direct builds a response message containing the fields listed in Appendix A6. This data is POSTed via the customer's browser to the `PayPalCallbackURL` (which you provided as part of the original Direct POST Appendix A1). This URL is also the place to which the customer's browser is redirected in the event of any errors.

12. Your site can check the information in the message to determine if you wish to proceed with the transaction. If the `AddressStatus` is **UNCONFIRMED**, and the `PayerStatus` is **UNVERIFIED**, for example, you may not wish to continue without PayPal Seller Protection. If you do NOT wish to proceed, you should build a Direct PayPal Capture message with the `Accept` field set to **NO** (Appendix A7) and POST it to the Direct PayPal Completion URL. You can then redirect the customer back to select a different payment method at this stage, and begin the Direct process again.

13. If you DO wish to proceed, you should store the delivery address details in your database (if they differ from those supplied), then build a Direct PayPal Capture message with the `Accept` field set to **YES** (Appendix A7) and POST it to the Direct PayPal Completion URL.

14. Direct will validate the POST and, if correct, forward that to PayPal.

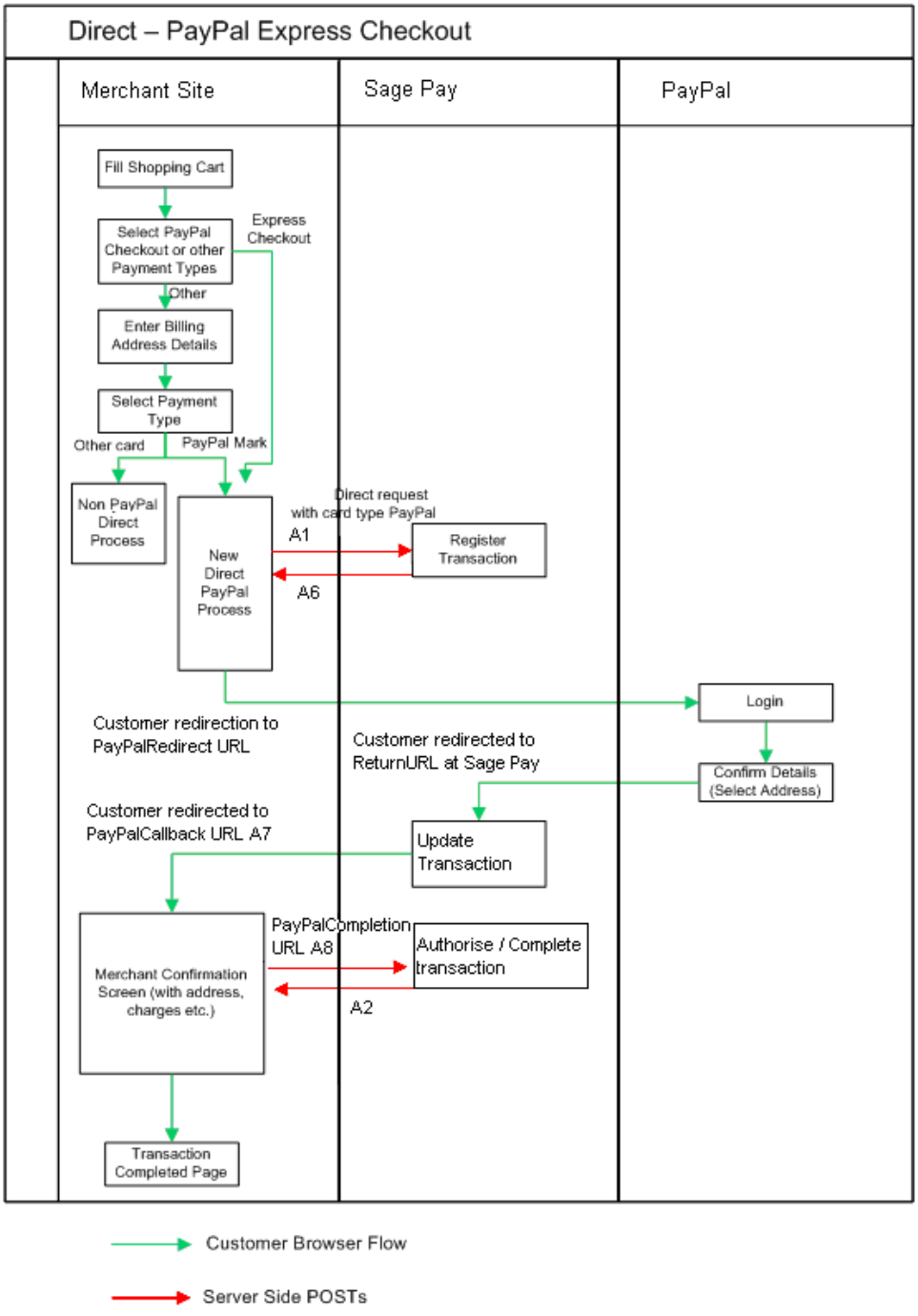
15. PayPal will complete the transaction and return the details to Direct. Sage Pay will update the transaction with the required IDs and build a completion response.

16. Direct replies to the POST sent to the PayPal Completion URL with the Direct Completion message (Appendix A2).

17. You display a completion page to the customer.

4.1 Direct PayPal Message Flow

The diagram below shows the message and customer flow for a Direct PayPal payment.



5.0 Integrating with Sage Pay Direct

Linking your Website to Sage Pay with Direct involves creating one script (or modifying the example provided in the integration kits), which both registers the transaction with our servers and processes the response we send back. If you wish to support 3D-Secure Authentication, you will also need to create or modify a second script to handle the call back from the Issuing Bank. If you wish to integrate with PayPal, additional coding is also required to redirect to a PayPal logon URL. After that, there is a call back to your servers from Sage Pay, and an additional server-to-server POST to confirm the transaction and complete the process.

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, and a version of MySagePay also runs here so you can familiarise yourself with the features available to you.

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

Transactions from your scripts should be sent to the Test Site at:
https://test.sagepay.com/gateway/service/vspdirect-register.vsp

3D-secure callback POSTS should be sent to the following URL:
https://test.sagepay.com/gateway/service/direct3dcallback.vsp

PayPal Completion POSTS should be sent to the following URL:
https://test.sagepay.com/gateway/service/complete.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, AND you've completed the online Direct Debit signup and your Merchant Account details have been confirmed, your account will be set up on our Live servers. You then need to redirect your scripts to send transactions to the Live service, send through a Payment using your own credit card, then VOID it through the MySagePay service so you don't charge yourself. If this works successfully, then you are ready to trade online.

The MySagePay system for viewing your Live transactions is at:
https://live.sagepay.com/mysagepay

Transactions from your scripts should be sent to the Live Site at:
https://live.sagepay.com/gateway/service/vspdirect-register.vsp

3D-secure callback POSTS should be sent to the following URL:
https://live.sagepay.com/gateway/service/direct3dcallback.vsp

PayPal Completion POSTS should be sent to the following URL:
https://live.sagepay.com/gateway/service/complete.vsp

6.0 Testing on the Test Server (Stage 1)

The Test Server is an exact copy of the Live System but without the banks attached. This means you get a true user experience but without the fear of any money being taken from your cards 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 : https://support.sagepay.com/apply/. 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 under the same Vendor Name as your online application form within 48 hours of submitting a completed 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 to view 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 the Direct integrated payment method. In many kits this is done simply by changing the strConnectTo string in the includes file, 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/vspdirect-register.vsp

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

6.1 Registering a Payment

If you do not 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 the transactions and handling the notification POSTs. Ensure you've completed all configuration in the includes file as detailed in the kit instructions, then locate the Transaction Registration script (called transactionRegistration).

This script provides a worked example of how to construct the Transaction Registration POST (see Appendix A1) and how to read the response that comes back (Appendix A2).

If you plan to implement 3D-Secure Authentication, the kit also provides a Terminal URL example page which implements section A3 of the attached protocol.

Check that this script is sending transactions to the Sage Pay Test server and not the live site then execute this script. You may need to develop a simple payment page that allows you to enter card details and passes them to this script if this page is not included in your kit. Use the script to send a

payment registration to the Test server. 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, SecurityKey and TxAuthNo fields alongside your uniquely generated VendorTxCode and the order details in your own database. You may wish to store the 3DSecureStatus field if you plan to support 3D-Secure.

Your script should then redirect the customer to a completion page thanking them for their order.

In the real world, the bank will either authorise the transaction (an **OK** response) or fail it (a **NOTATUHED** response), or Sage Pay may reverse an authorisation if your fraud screening rules are not met (a **REJECTED** response). You should make sure your code can handle each message appropriately. Normally **NOTAUTHED** messages would prompt the user to try another card and **REJECTED** messages would ask them to check their Address and CV2 details are correct and resubmit, or to try another card. You may wish to store the VPSTxId and SecurityKey of the failed transaction against your VendorTxCode and generate a new VendorTxCode for the retry attempt if you wish to keep a history of the failed transactions as well as the successful one.

You should test each type of error message (**MALFORMED**, **INVALID** and **ERROR**) with your payment script to check that all message types are handled correctly. **MALFORMED** messages should only occur during development when the POST may be incorrectly formatted, and **INVALID** messages can be avoided by pre-validating the user input. In the case of **ERROR**, your code should present the customer with a page saying that online payment was not currently available and offering them an alternative contact telephone number for payment or request them to come back later.

6.1.1 3D-Authenticated Transactions

If you plan to support 3D-Secure, you should now go on to test that your scripts can handle these messages.

Send a transaction registration POST and rather than receiving an **OK** Status, your script will receive a **3DAUTH** Status instead. A simulated MD, PAReq and ACSURL will be provided and you should ensure that your script builds the simple, automatically-submitting, HTML FORM code and redirects your browser to the 3D-Authentication page.

You need to ensure that the Terminal URL you have provided points to the fully qualified URL of the callback page provided in your script. This should begin with https:// (since the Terminal URL must be secured) and provide the full path to the page.

Your Terminal URL code (normally a page called 3DCallback in the kits) should be modified to store the result fields in your database (as you did for your transaction registration code in Step 3), including the 3DSecureStatus field and, for 3D-Authenticated transactions, the CAVV field (the unique signature for a validated 3D-Secure transaction).

You can then direct your customer to the relevant completion page, depending on the Status of the transaction. Like non-authenticated transactions, a Status of **OK** should redirect the user to a success page, and **ERROR**, **NOTAUTHED**, **REJECTED**, **MALFORMED** or **INVALID** to various error handling pages.

6.1.2 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 return the `Status` **3DAUTH** and `3DSecureStatus` **OK**
N = Not Enrolled, will return the `3DSecureStatus` **NOAUTH**
U = Unable to verify enrolment, will return the `3DSecureStatus` **CANTAUTH**
E = Error occurred during verification, will return the `3DSecureStatus` **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.

6.2 Direct PayPal transactions

You should ONLY begin to test your PayPal integration once you are happy that your site can correctly send and process the messages exchanged between your site and ours for a standard Direct transaction.

PayPal is now available to be used within the Sage Pay test environment allowing you to test your integration and ensure that it is working smoothly without having to use a real live PayPal account.

In order to test PayPal integration with Sage Pay, you will need to create an account and login to https://developer.paypal.com. Under the Sandbox accounts you must create a:

- Personal (buyer account)
- Business (merchant account)

You will be given an Email address for each of the accounts you create.

You will need to log into the Business (merchant account) and under API Access, add the following Third Party Permission Username. You should grant all available permissions.

ppdev_1256915571_biz_api1.sagepay.com
(Please note that this is different to the live API account)

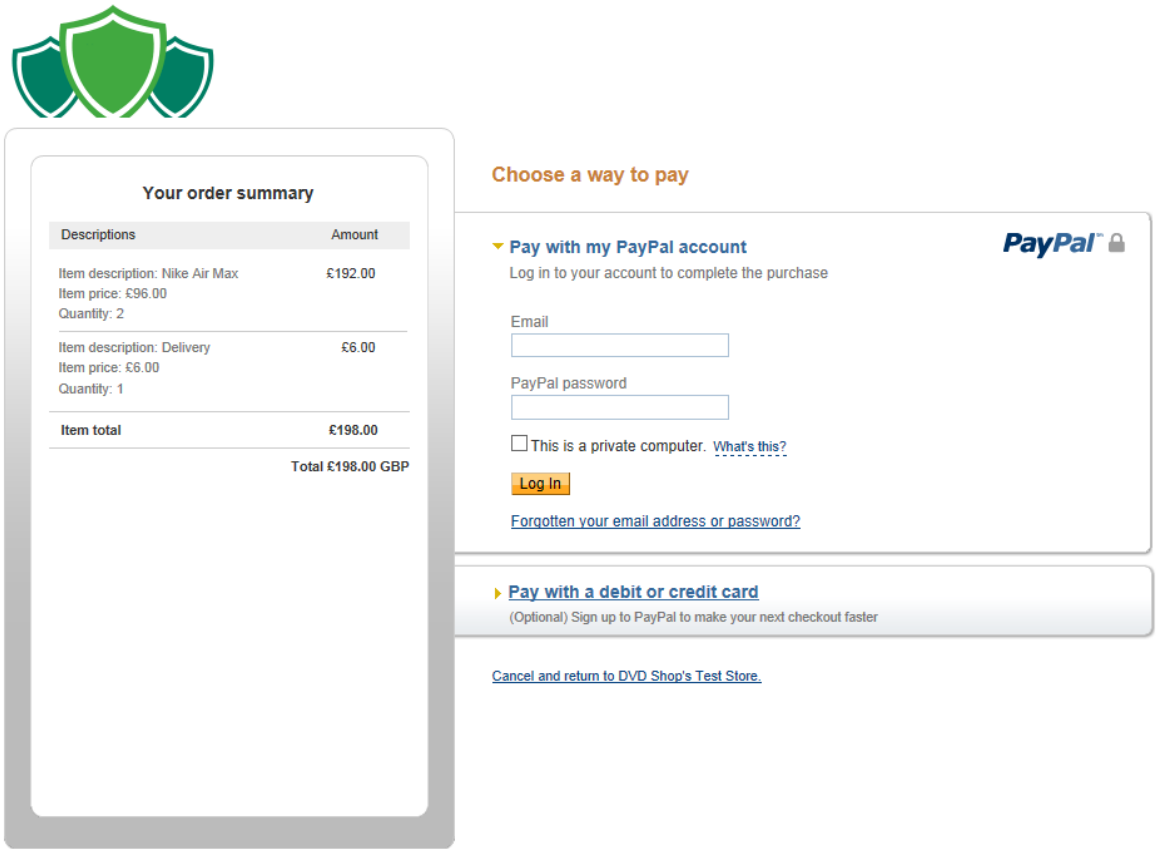
To test your PayPal integration you will need to log into your Test MySagePay and add the Email address which corresponds to the Business (merchant account) created above.

Whichever option you choose, either Express Checkout or Mark, you should send the Transaction Registration post with the `CardType` set to **PAYPAL**.

A `Status` of **PPREDIRECT** and a simulated `PayPalRedirectURL` will be provided in the Sage Pay response to your Transaction Registration Post.

Your code should store the `VPSTxId` and redirect the customer's browser to the PayPal Sign In page.

If provided the BasketXML information will be shown, along with your company logo.
You should login using the Personal (buyer account) previously created to complete the transaction process with PayPal.



Sage Pay will send a message to your `PayPalCallbackURL` along with the customer; you must ensure your script can handle a **PAYAPLOK** `Status`.

You now have the opportunity to `Accept` the transaction based on the `PayerStatus` and or `AddressStatus` (as the result of these fields can dictate if the transaction is eligible for PayPal Seller Protection).
You can also modify the `Amount` by +/- 15% of the original value (if the delivery price changes as a result of the address selected).

If you wish to proceed with the transaction, you send a POST to the PayPal Completion URL with a value of **YES** in the `Accept` field (see Appendix A7). This will return a `Status` of **OK** in the final response to your servers.

If the `AddressStatus` was **UNCONFIRMED**, and the `PayerStatus` **UNVERIFIED**, you may not wish to continue. If you do NOT wish to proceed, you would still need to send a POST to the Sage Pay servers to complete the transaction, but enter a value of **NO** in the `Accept` field to cancel the transaction (see Appendix A7). This will return a `Status` of **NOTAUTHED** in the final response to your servers.

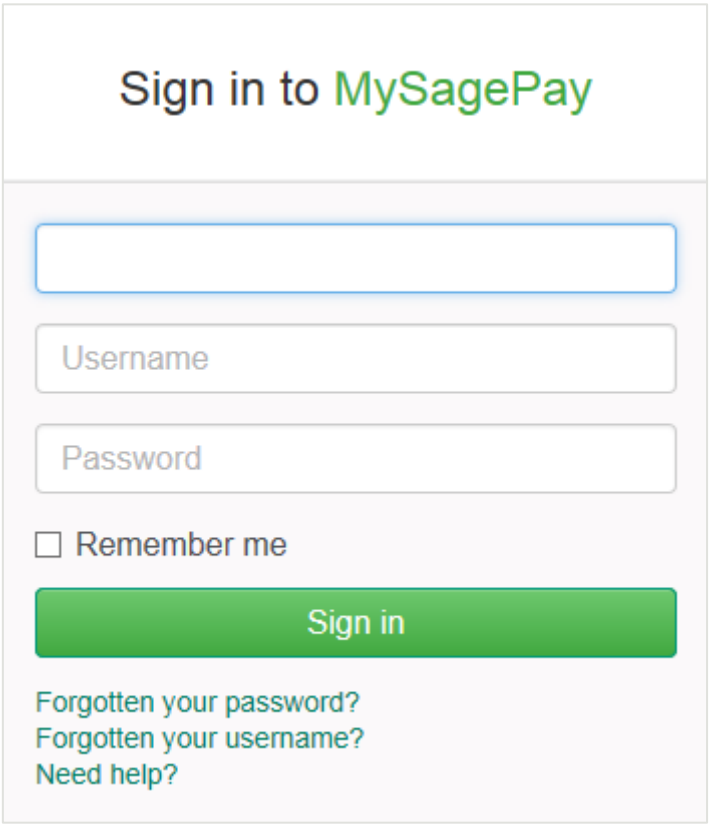
When you receive this final response from Direct, (see Appendix A2), you should redirect your customer to the relevant completion page on your site, depending on the `Status` of the transaction. Like standard transactions, a `Status` of **OK** should redirect the user to a success page, and **ERROR**, **NOTAUTHED**, **REJECTED**, **MALFORMED** or **INVALID** to various error handling pages.

Please visit PayPal's website for localised Seller Protection terms and conditions.

6.3 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

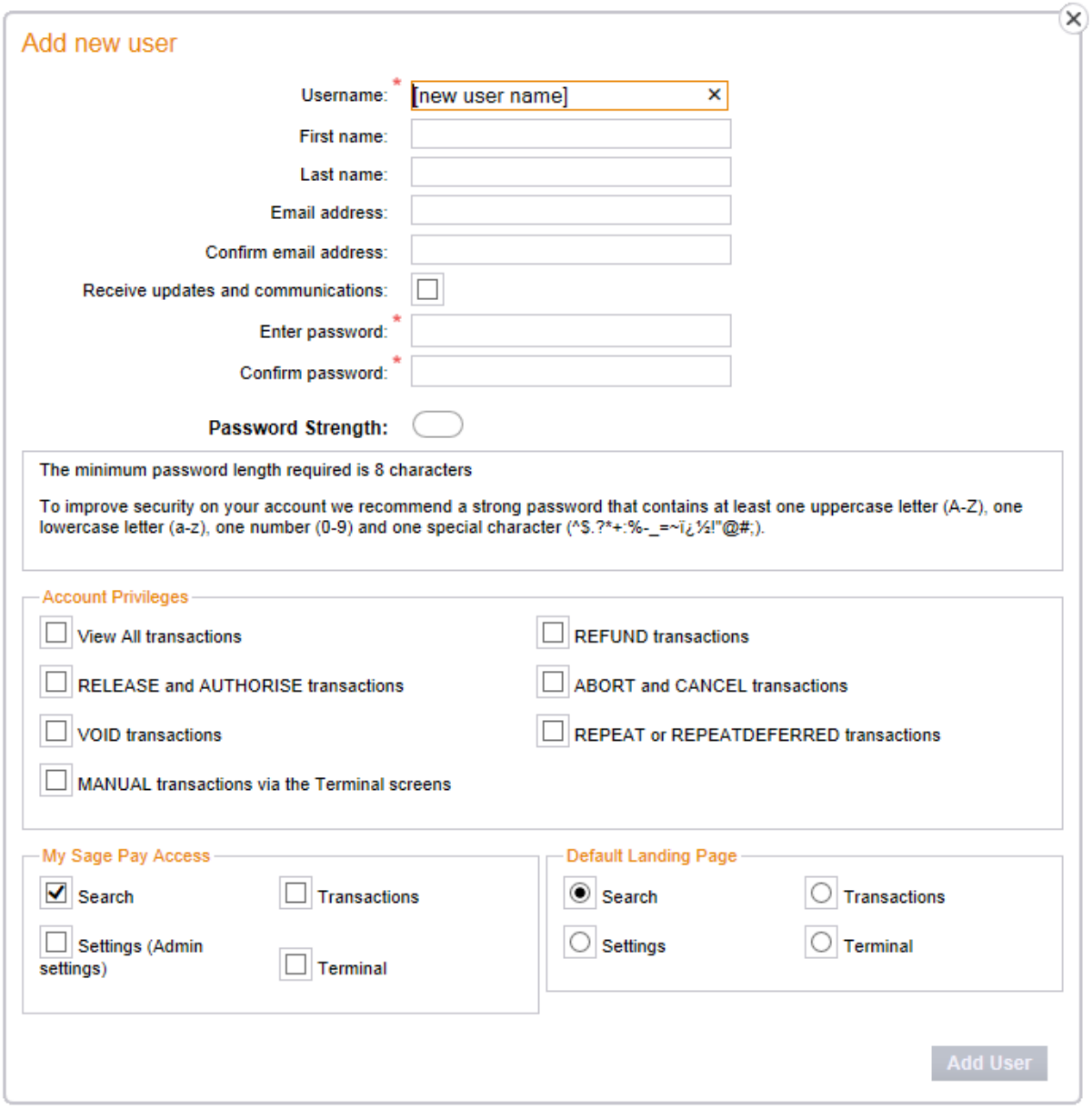


When you log in to MySagePay screens you will be asked for a Vendor Name, a Username and a Password. The first time you log in you will need to do so as your system Administrator:

- 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.

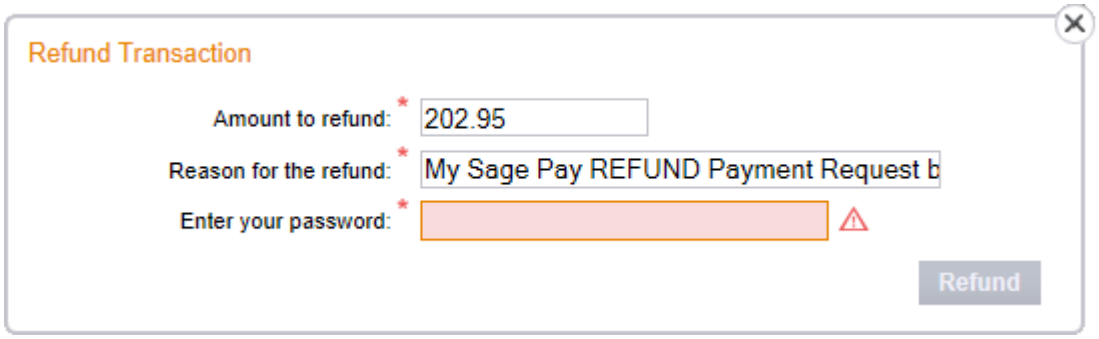
You are now logged in using your own account and can view your test transactions and use all additional functions. If you lock yourself out of your own account, you can use the Administrator account to unlock yourself or use the lost password link on the Sign In screen.

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.

6.4 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

7.0 Additional Transaction Types

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

7.1 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.

7.2 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.

7.3 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).

7.4 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.

8.0 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

9.0 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.

10.0 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/vspdirect-register.vsp

(for other transaction types, the vspdirect-register.vsp section would be changed to refund.vsp, void.vsp, release.vsp etc.)

The 3D-Secure Callback URL becomes:
https://live.sagepay.com/gateway/service/direct3dcallback.vsp

The PayPal Completion URL becomes:
https://live.sagepay.com/gateway/service/complete.vsp

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 6.3.

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

11.0 Congratulations, you are live with Sage Pay Direct

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.

12.0 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: Direct Protocol

A1. You submit your transaction registration POST

This is performed via a HTTPS POST request, sent to the initial Sage Pay Payment URL service vspdirect-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.
CardHolder	Yes	Aa á ' / \ & - . ‘	50 chars		This should be the name displayed on the card. Not required if CardType=**PAYPAL**
CardNumber	Yes	0-9	20 chars		The full card number is required. Not required if CardType=**PAYPAL**
ExpiryDate	Yes	0-9	4 chars		The expiry date of the card in the format of **MMYY** Not required if CardType=**PAYPAL**
CV2	No	0-9	4 chars		The extra security 3 digits on the signature strip of the card, or the extra 4 digits on the front for American Express Cards If AVS/CV2 is ON for your account this field becomes compulsory. Not required if CardType=**PAYPAL**
CardType	Yes	Aa	15 chars	**VISA** **MC** **MCDEBIT** **DELTA** **MAESTRO** **UKE** **AMEX** **DC** **JCB** **LASER** **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 **LASER** is Laser (withdrawn as of 28th February 2014) **PAYPAL** The value should be in UPPERCASE.
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

BillingFirstnames	Yes	Aa á ␣ /\ & - . ' , 0-9	20 chars		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**.
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 (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**.
DeliveryFirstnames	Yes	Aa á ␣ /\ & - . ' , 0-9	20 chars		
DeliveryAddress1	Yes	Aa á ␣ /\ & - . ' , 0-9 : + () CR/LF	100 chars		
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		
PayPalCallbackURL	No	RFC1738	255 chars	Must begin http:// or https://	Fully qualified domain name of the URL to which customers are redirected upon completion of a PayPal transaction. Only required if CardType=**PAYPAL**
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.
GiftAidPayment	No	0-9	Flag	**0** (default) **1**	Setting this field means the customer has ticked a box on your site to indicate they wish to donate the tax. **0** = This transaction is not a Gift Aid charitable donation (default) **1** = This payment is a Gift Aid charitable donation and the customer has AGREED to donate the tax. Only of use if your vendor account is Gift Aid enabled

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 transactions.
ClientIPAddress	No	0-9 .	15 chars		The IP address of the client connecting to your server making the payment. This should be a full IP address which you can obtain from your server scripts. We will attempt to Geolocate the IP address in your reports and fraud screening.

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 transactions.
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.

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.
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.

Validation is performed on the totals of the basket and the transaction will fail if the following amounts do not add up:

<unitGrossAmount> = <unitNetAmount> + <unitTaxAmount>

<totalGrossAmount> = <unitGrossAmount> x <quantity>

Amount (sent in transaction Registration) = <totalGrossAmount> + <deliveryGrossAmount> - <fixed> (discounts)

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 + ()	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. Sage Pay response to the Transaction Registration or Callback POSTs

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** **NOTAUTHED** **REJECTED** **AUTHENTICATED** **REGISTERED** **3DAUTH** **PPREDIRECT** **MALFORMED** **INVALID** **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. Also returned for PayPal transactions in response to the PayPal Completion Post (if `Accept`=**NO** was sent to complete PayPal transaction, see Appendix A8). **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. **3DAUTH** = The customer needs to be directed to their card issuer for 3D-Authentication. GO TO APPENDIX A3. **PPREDIRECT** = The customer needs to be redirected to PayPal. GO TO APPENDIX A6. **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**.

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**.
TxAuthNo	No	0-9	10 chars		Sage Pay unique Authorisation Code for a successfully authorised transaction. Only present if Status is **OK**.
AVSCV2	Yes	Aa	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 **3DAUTH, AUTHENTICATED, PPREDIRECT** or **REGISTERED.**
AddressResult	Yes	Aa	20 chars	**NOTPROVIDED** **NOTCHECKED** **MATCHED** **NOTMATCHED**	The specific result of the checks on the cardholder's address numeric from the AVS/CV2 checks. Not present if the Status is **3DAUTH, AUTHENTICATED, PPREDIRECT** or **REGISTERED.**
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 **3DAUTH, AUTHENTICATED, PPREDIRECT** 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 **3DAUTH, AUTHENTICATED, PPREDIRECT** or **REGISTERED.**
3DSecureStatus	Yes	Aa	50 chars	**OK** **NOTCHECKED** **NOTAUTHED** **INCOMPLETE**	This field details the results of the 3D-Secure checks (where appropriate) **OK** = 3D Secure checks carried out and user

				ERROR **ATTEMPTONLY** **NOAUTH** **CANTAUTH** **MALFORMED** **INVALID**	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 with a setting like Apply3DSecure=2 **NOTAUTHED** = 3D-Secure authentication checked, but the user failed the authentication. **INCOMPLETE** = 3D-Secure authentication was unable to complete. No authentication occurred. **ERROR** = Authentication could not be attempted due to data errors or service unavailability in one of the parties involved in the check. **ATTEMPTONLY** = The cardholder attempted to authenticate themselves but the process did not complete. A `CAVV` is returned; therefore a liability shift may occur for non-Maestro cards. Check your Merchant Agreement. **NOAUTH** = This means the card is not in the 3D-Secure scheme. **CANTAUTH** = This normally means the card Issuer is not part of the scheme. **MALFORMED / INVALID** = These statuses indicate a problem with creating or receiving the 3D-Secure data. These should not occur on the live environment.

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**
Token	No	Aa 0-9 - {}	38 chars		The token generated by Sage Pay.
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
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
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.

A3. Sage Pay response to the transaction registration POST (3D-Secure)

If 3D-Authentication is available on your account and the Card AND the Card Issuer are (or can be) part of the scheme, this is the plain text response part of the POST originated by your servers in A1. Encoding will be as Name=Value fields separated by carriage-return-linefeeds (CRLF).

Response format

Name	Mandatory	Format	Max Length	Allowed Values	Description
Status	Yes	Aa 0-9	15 chars	**3DAUTH**	**3DAUTH** = Only returned if 3D-Authentication is available on your account AND the directory services have issued a URL to which you can progress.
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**
3DSecureStatus	Yes	Aa 0-9	20 chars	**OK**	**OK** = If a Status of **3DAUTH** is returned at this stage, the only value you will receive for the 3DSecureStatus is **OK**.
MD	Yes	Aa 0-9	35 chars		A unique reference for the 3D-Authentication attempt.
ACSURL	Yes	RFC1738	7500 chars		A fully qualified URL that points to the 3D-Authentication system at the Cardholder's Issuing Bank.
PAReq	Yes	BASE64	7500 chars		A Base64 encoded, encrypted message to be passed to the Issuing Bank as part of the 3D-Authentication. When forwarding this value to the ACSURL, pass it in a field called PaReq (note the lower case a). This avoids issues with case sensitive ACSURL code.

At this point your server builds an auto-submitting form which sends the PaReq, MD and an additional field TermUrl to the address specified in the ACSURL. Sending this form to your customer's browser will redirect them to their Card Issuers 3D-Authetnication site.

Results will be sent to your TermUrl in an encryoted field called PaRes, you then forward this to Sage Pay in Appendix A4.

A4. 3D-Authentication Results POST from your Terminal URL to Sage Pay (3D-Secure)

This is performed via a HTTPS POST request, sent to the Direct 3D-Secure Callback URL. The details should be URL encoded Name=Value fields separated by '&' characters.

Request format

Name	Mandatory	Format	Max Length	Allowed Values	Description
MD	Yes	Aa 0-9	35 chars		A unique reference for the 3D-Authentication attempt. This will match the MD value passed back to your site in response to your transaction registration POST.
PARes	Yes	BASE64	7500 chars		A Base64 encoded, encrypted message sent back by Issuing Bank to your Terminal URL at the end of the 3D-Authentication process. This field must be passed back to Direct along with the MD field to allow the Sage Pay MPI to decode the result. You will receive this value back from the Issuing Bank in a field called PaRes (lower case a"), but should be passed to Sage Pay as PARes.

The response from the 3D Callback service is identical to that of the initial registration POST. See Appendix A2.

A5. Sage Pay response to the Transaction Registration POST (PayPal)

If you supplied PayPal as a CardType in A1 above and PayPal is active on your account, this response is returned from the server. Encoding will be as Name=Value fields separated by carriage-return-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 0-9	15 chars	**PPREDIRECT**	**3DAUTH** = Only returned if 3D-Authentication is available on your account AND the directory services have issued a URL to which you can progress.
StatusDetail	Yes	Aa 0-9	255 chars		Human-readable text providing extra detail for the Status message.
VPSTxId	Yes	Aa 0-9 - {}	38 chars		The Sage Pay ID to uniquely identify the transaction on our system.
PayPalRedirectURL	Yes	RFC1738	255 chars		A fully qualified domain name URL to which you should redirect the customer. Contains the PayPal token which should not be stripped out.

A6. Sage Pay Callback after PayPal Authentication (PayPal)

After redirecting your customer to the PayPalRedirectURL in step A6 above, this message sent to your PayPalCallbackURL, along with the customer, after they have completed their PayPal authentication and payment method selection.

It provides all relevant information about the transaction to allow you to decide if you wish to proceed with the payment (see A8 below). The information will be in the form of 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**	Protocol version used by the system. Same as supplied in A1.
Status	Yes	Aa 0-9	15 chars	**PAYPALOK** **MALFORMED** **INVALID** **ERROR**	**PAYPALOK** = The customer has selected a payment type and the transaction is ready to be taken **MALFORMED** = Input message was missing fields or badly formatted – normally will only occur during development and vendor integration. **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 completion. Please notify Sage Pay if a Status of **ERROR** is seen, together with your VendorTxCode and the StatusDetail text.
StatusDetail	Yes	Aa 0-9	255 chars		Human-readable text providing extra detail for the Status message.
VPSTxId	Yes	Aa 0-9 - {}	38 chars		The Sage Pay ID to uniquely identify the transaction on our system.

PayerStatus	Yes	RFC1738	255 chars	**VERIFIED** **UNVERIFIED**	**VERIFIED** lets other members know the customer is a confirmed PayPal member with a current, active bank account, it also means the transaction may be eligible for PayPal Seller Protection. Contact PayPal for more information.
DeliverySurname	Yes	Aa á ' - . ' , 0-9	20 chars		If the customer modified their delivery details whilst on PayPal' site, the updated details are returned to you in these fields. Otherwise the delivery details supplied in your registration post will be returned.
DeliveryFirstnames	Yes	Aa á ' - . ' , 0-9	20 chars		
DeliveryAddress1	Yes	Aa á ' - . ' , 0-9 : + () CR/LF	100 chars		
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		
DeliveryState	No	US	2 chars	Examples: **AL**, **MS** and **NY**	
DeliveryPhone	No	0-9 - Aa + ()	20 chars		
AddressStatus	Yes	Aa 0-9	20 chars	**NONE** **CONFIRMED** **UNCONFIRMED**	**CONFIRMED** = A buyer's Confirmed Address is checked against the credit card billing address maintained by his or her credit card company, or is verified by PayPal. It also means the transaction may be eligible for PayPal Seller Protection. Contact PayPal for more information.
CustomerEMail	Yes	RFC532N	255 chars	Examples: me@mail1.com:me@mail2.com	The customer's email address registered at PayPal.
PayerID	Yes	Aa 0-9	15 chars		Unique PayPal User Reference ID

A7. Complete a PayPal Transaction (PayPal)

If you wish to complete a PayPal transaction you must send a completion POST to the Sage Pay servers.

This is performed via an HTTPS POST request, sent to the Direct PayPal Completion URL. 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**	Protocol version used by the system. Same as supplied in A1.
TxType	Yes	Aa 0-9	15 chars	**COMPLETE**	
VPSTxId	Yes	Aa 0-9 - {}	38 chars		The Sage Pay ID to uniquely identify the transaction on our system.
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. The amount can vary from the original POST in A1 by +/- 15% of the original amount (for example, if delivery prices change as a result of the address selected). 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.
Accept	Yes	Aa	3 chars	**YES** **NO**	**YES** = You wish to proceed with the PayPal transaction. **NO** = You wish to cancel based on the information returned.

The response to the completion POST is identical to that of the initial registration POST. See Appendix A2.

13.0 URLs

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

POST	Environment	URL
REGISTRATION	**TEST**	https://test.sagepay.com/gateway/service/vspdirect-register.vsp
REGISTRATION	**LIVE**	https://live.sagepay.com/gateway/service/vspdirect-register.vsp
3D-SECURE CALLBACK	**TEST**	https://test.sagepay.com/gateway/service/direct3dcallback.vsp
3D-SECURE CALLBACK	**LIVE**	https://live.sagepay.com/gateway/service/direct3dcallback.vsp
PAYPAL COMPLETION	**TEST**	https://test.sagepay.com/gateway/service/complete.vsp
PAYPAL COMPLETION	**LIVE**	https://live.sagepay.com/gateway/service/complete.vsp

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