



Direct Protocol and Integration Guideline (Protocol v3.00)

Published Date 16/09/2013

Version History	3
LEGAL NOTICE	3
Welcome to the Sage Pay Direct integration method	4
Overview of how Direct integrated payments work	5
Direct and 3D-Secure	6
Direct and PayPal	8
Step 1: The customer orders from your site.	9
Step 2: Your server registers the payment with Sage Pay.	10
Step 3: Sage Pay Direct checks 3D-Secure enrolment.	12
Step 4: Direct replies to your registration POST.	13
Step 5: You redirect your customer to their Issuing Bank.	14
Step 6: 3D-Authentication is carried out and your site called back.	15
Step 7: Your site POSTs the 3D-Secure results to Sage Pay.	16
Step 8: Direct requests card authorisation.	17
Step 9: Sage Pay reply to your server's POST.	18
Step 10: Sage Pay sends settlement batch files to confirm payments.	19
Applying Surcharges to Transactions	20
Direct Payments using PayPal	21
Direct PayPal Message Flow	23
Integrating with Direct	25
Stage 1: Testing on the Test Server	26
Registering a payment	26
3D-Authenticated Transactions	28
Direct PayPal transactions	29
The Test Server MySagePay interface	34
Additional Transaction Types	36
Stage 2: Going Live	40
Congratulations, you are Live with the Sage Pay.	41
Appendix A - The Sage Pay Direct 3.00 Protocol	42
A1: Transaction Registration	43
A1.1 SurchargeXML field	51
A1.2 Basket Contents	52
A1.3 BasketXML field	55
A1.4 CustomerXML field	64
A2: Sage Pay Response to the Registration or Callback POSTs	66
A3: Sage Pay Response to the Registration POST (3D Secure)	71
A4: 3D-Authentication Results POST from your Terminal URL to Sage Pay. .	72
A5: Response to 3D Callback POSTs	72
A6: Sage Pay Response to the Registration POST (for PayPal transactions) .	73
A7: Sage Pay Callback after PayPal Authentication	74
A8: Complete PayPal transaction	77
A9: Direct Full URL Summary	78
A10: Sample XML	78
Surcharge XML - Example 1	78
Basket XML - Example 2	79
Customer XML - Example 3	89

Version History

A * in the first column of the Appendices below indicates this is a change since the last published version of the document

Date	Change	Page
19/07/13	Update to Basket XML to include Discounts Change allowed characters in BankAuthCode to Alphanumeric	60 70
27/08/13	Updated to say that Diners and AMEX may require additional merchants accounts Updated to make the usage of the Surcharge XML clearer Added StoreToken Field Updated description of amount field to make it clearer that it should not include any surcharge applied	44 20, 51 50 43
16/09/13	Added ExpiryDate as a returned field in the A2 : Response To Registration.	70

LEGAL NOTICE

This Protocol and Integration Guideline 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.

Copyright © Sage Pay Europe Limited 2013. All rights reserved.

Welcome to the Sage Pay Direct integration method

The Sage Pay payment system provides a secure, simple means of authorising credit and debit card transactions from your website.

Direct is designed to enable you to take card details 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 pages. This enables you to white-label the payment process. Your customer never leaves your site (unless you are using PayPal, European Payment Types 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 number 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 Verified by Visa and MasterCard SecureCode (the cardholder authentication systems collectively known as 3D-Secure) or AMEX SafeKey, 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 testing and live environments. It also contains the complete Payment Protocol in the Appendix.

NOTE: Since card details 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 the link below:

www.sagepay.com/pci-dss-compliance

Overview of how Direct integrated payments work

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.

Non 3D-Secured 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 Protocols", gives details of how to perform other transaction-related POSTs, such as Refunds, Repeat payments and the Release/Abort mechanisms for Deferred transactions.

Direct and 3D-Secure

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 more 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 Terminal URL 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.

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 for a PayPal Account by visiting the PayPal website:

<https://www.paypal-marketing.co.uk/merchantservices/sagepay/>

This additional service can be included in your package at no additional cost (standard PayPal transaction fees will apply, please visit the following link for further details:

https://www.paypal.com/uk/cgi-bin/webscr?cmd=_display-receiving-fees-outside).

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.

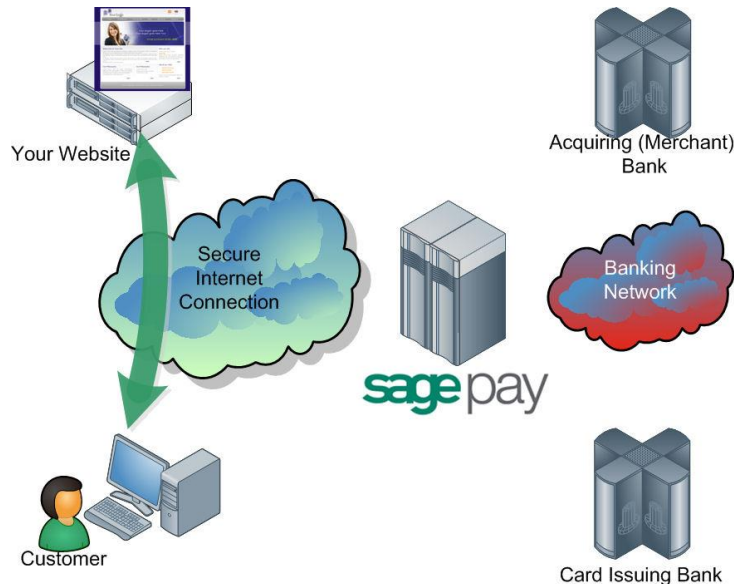
There is an initial server-to-server POST with Sage Pay, then a redirection to the PayPal logon URL. After that, there is a call back to your servers from Sage Pay, and an additional server-to-server POST to confirm the transaction and complete the process.

The Direct PayPal transaction process is detailed in the Direct PayPal Payments section later in this document.

The Direct payment process in detail (non-PayPal payments)

This section defines the messages exchanged between your web servers and the Sage Pay Direct system.

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.

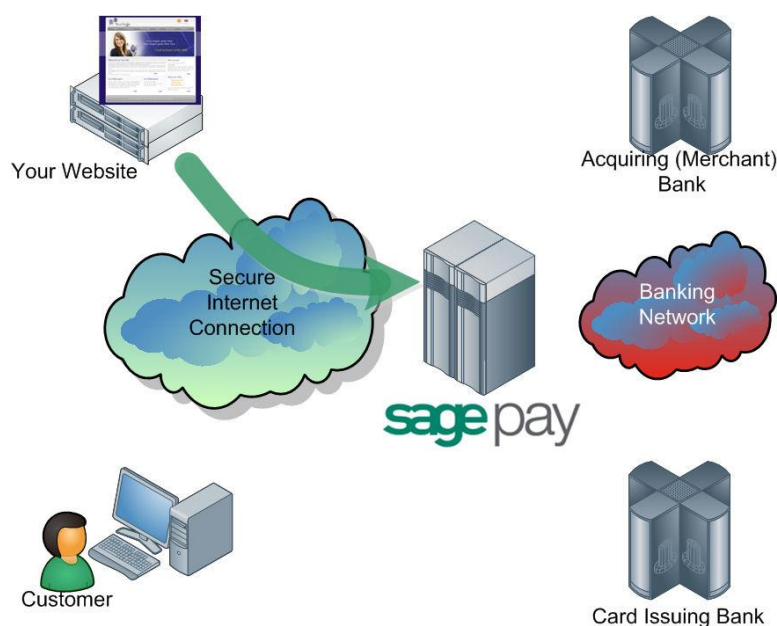
It is generally a good idea to identify the customer by name, email address, delivery and billing address and telephone number. It is also helpful to have your server record the IP Address from which the customer is accessing your system. You should store these details in your database alongside details 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, LASER)
- 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 list boxes for Start and Expiry Dates, 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 total value 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 name 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.

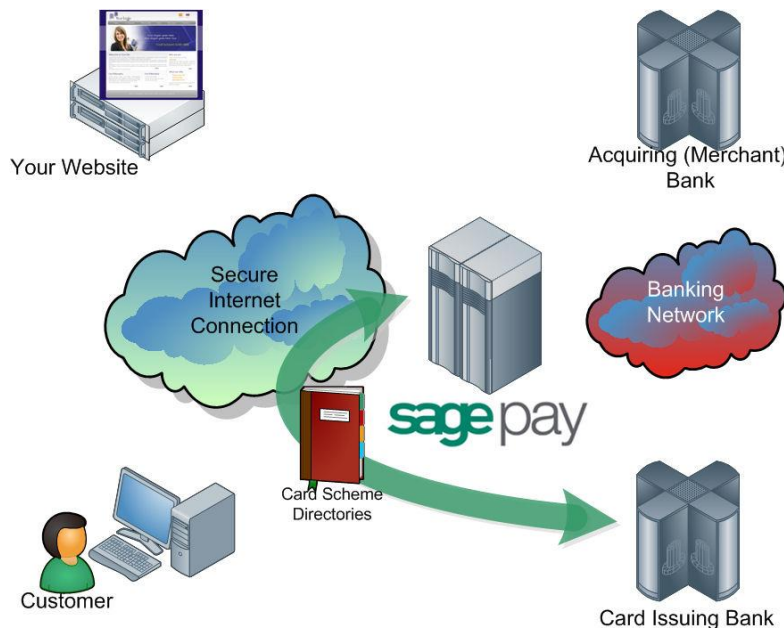
The integration kits we provide contain scripts in a variety of languages that illustrate how you compose and send this message from your server to ours. These can be downloaded as part of the application process or obtained from the download area on our website: <http://www.sagepay.com/help/downloads>.

Please note you will need to log into the website to access these files.

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 Direct checks 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.

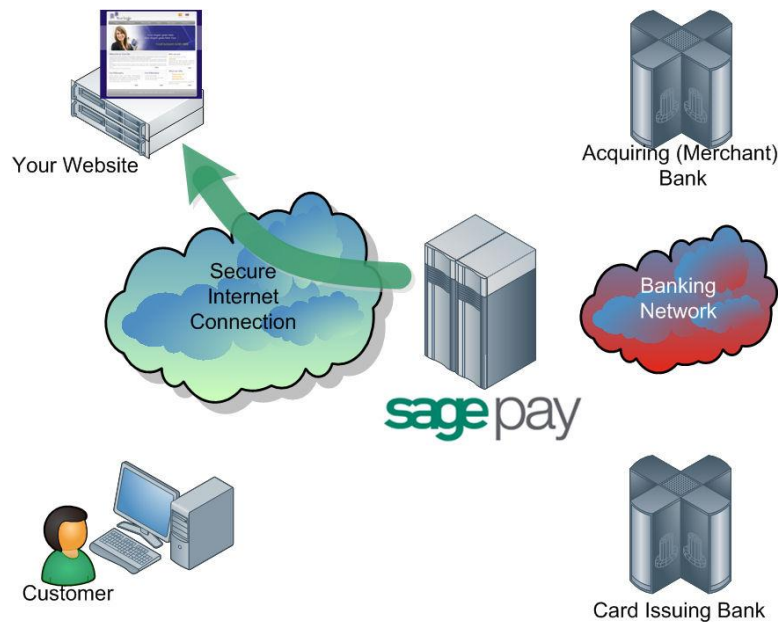
If the card or the issuer is not part of the scheme, or if an MPI error occurs, our server will check your 3D-Secure rule base to determine if authorisation should occur. For information regarding 3D Secure rule bases please refer to the Sage Pay Fraud Prevention Advice Guide, which can be downloaded from our website: <http://www.sagepay.com/help/downloads>. 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 rulebase 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](#) in these circumstances, for the authorisation stage.

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** (see the [next step](#)).

Step 4: Direct 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.

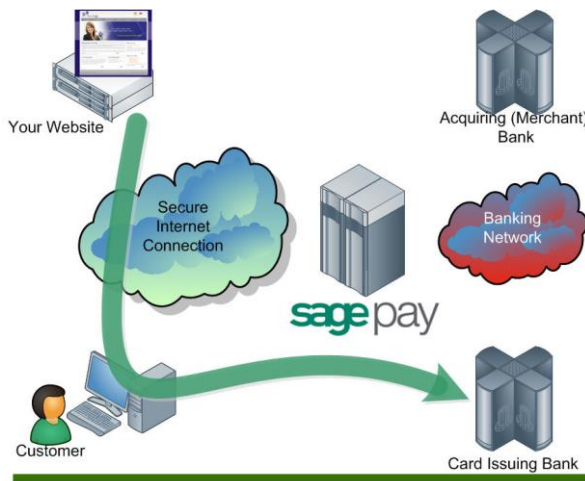
A unique identifier to your transaction called the **MD** is passed along with a preformatted, encrypted field called **PAReq**. This is the 3D-Secure message that the customer's card Issuing Bank decodes to begin the 3D-authentication process. The PAReq is created and encrypted by the Sage Pay MPI and 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](#) above). 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.

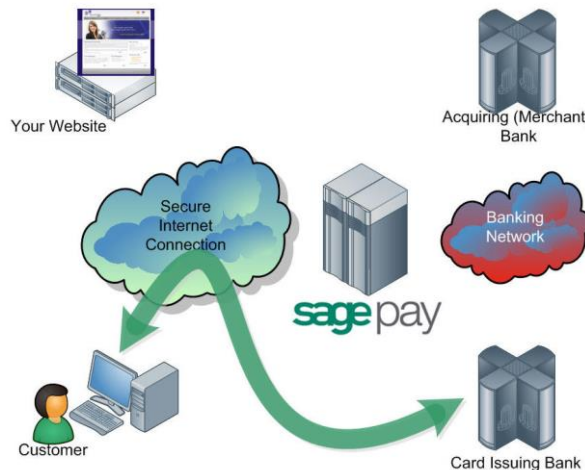
The first step of the Direct transaction is now complete. You have registered a 3D-Secure transaction with Sage Pay; we have stored your payment details and replied with everything you need to send your customer for 3D-Authentication. The next parts of the process, [steps 5 and 6](#), are out of our 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 the 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 below). Example code for this page is included in the integration kits provided by Sage Pay; see the example asp code below using an IFrame:

```
<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 & "\"/>"

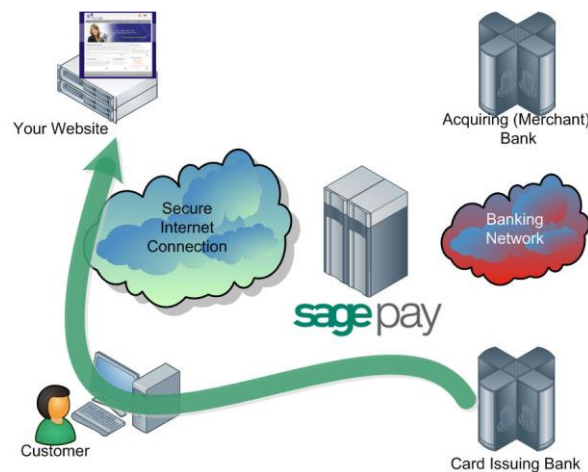
  response.write "<NOSCRIPT>"
  response.write "<center><p>Please click button below to Authenticate your
  card</p><input type='\"submit\"' value='\"Go\"'/></p></center>"
  response.write "</NOSCRIPT>"
  response.write "</form></body></html>\"">
</IFRAME>
```

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. NOTE: 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 (as above).

ADDITIONAL NOTE: 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 if you pass an upper case A. See the example code above for how to submit the data.

Step 6: 3D-Authentication is carried out and your site called back.



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

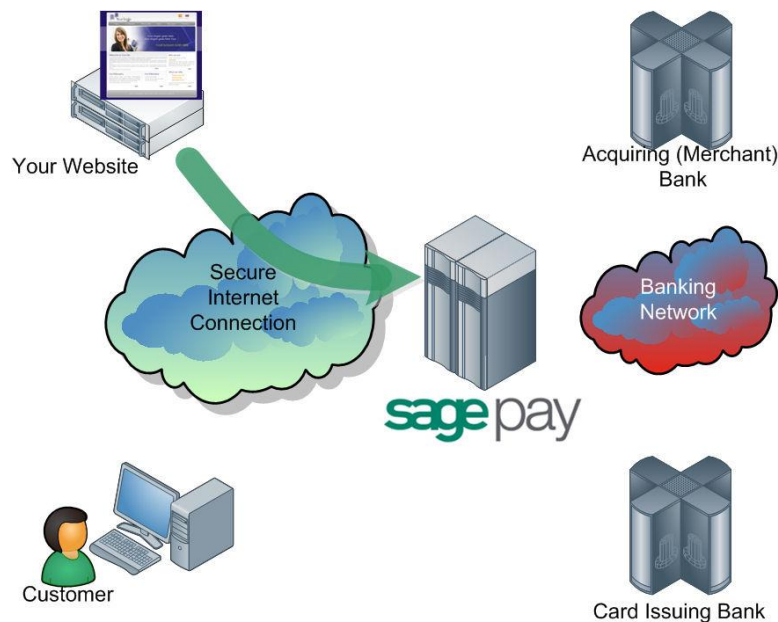
Once complete (either successfully or not), the bank will redirect your customer back to the page supplied in the **TermUrl** field you sent in step 5 above.

Along with this redirection, two fields are also sent: The **MD** value, to uniquely identify the transaction you are being called back about, and 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 file 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).

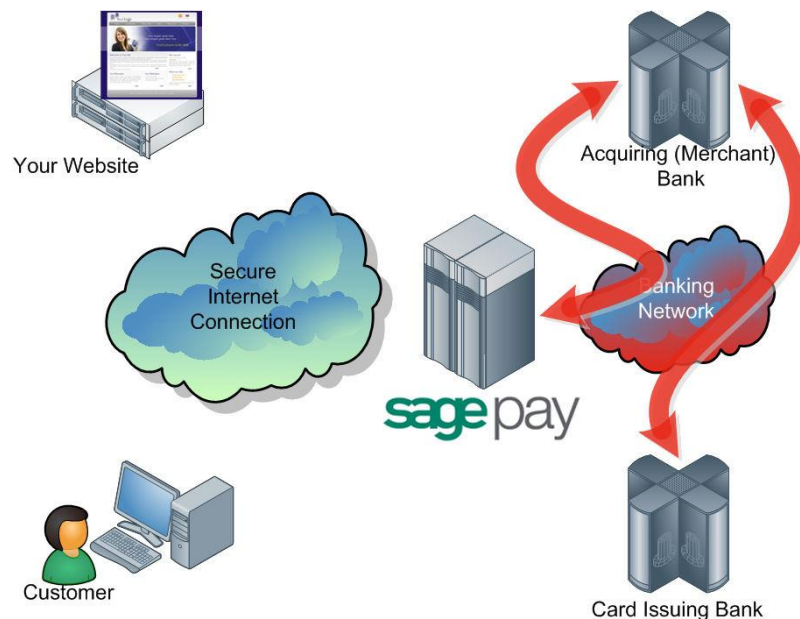
No other information is necessary because the Sage Pay system can use these values to retrieve all the transaction information you originally supplied.

If the decoded PARes indicates that the 3D-Authentication was successful, the Sage Pay gateway goes on to obtain an authorisation (see the next step). 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 the Sage Pay Rulebase Guide for more information about using 3D-Secure and AVS/CV2 rules (<http://www.sagepay.com/help/userguides/573>).

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

ADDITIONAL NOTE: 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: Direct requests card authorisation.



The Sage Pay services 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.

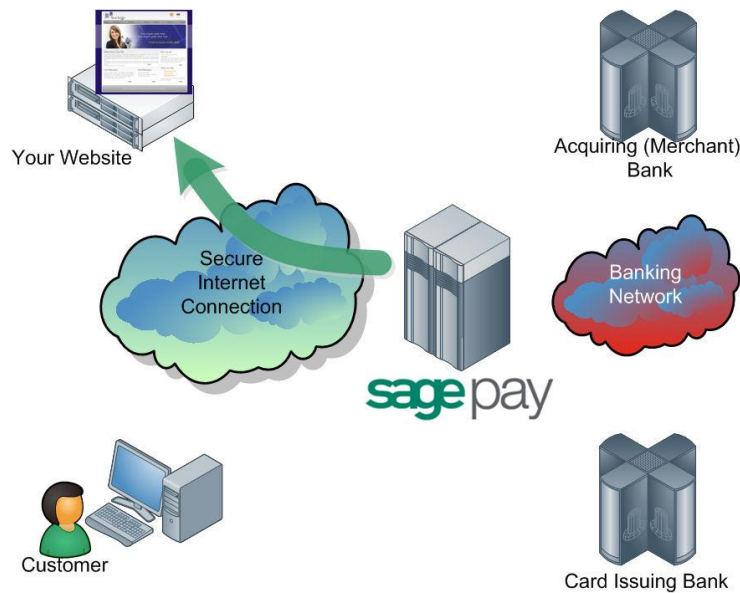
The request is normally answered within a second or so with either an authorisation code, or a failure message. This is obtained directly from the issuing bank by the acquiring bank in real time.

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 and CV2 checks are being performed, the results are compared to any rule bases you have set up (see the Sage Pay Rulebase Guide for more information). If the bank has authorised the transaction but the card has failed the AVS and/or CV2 rules you have established, Sage Pay immediately reverse the authorisation on the card and prepares a **REJECTED** response, returning the reason for the failure in the AVSCV2 field.

Please note: Some card issuing banks may decline the online 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.

Step 9: Sage Pay reply to your server's 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.

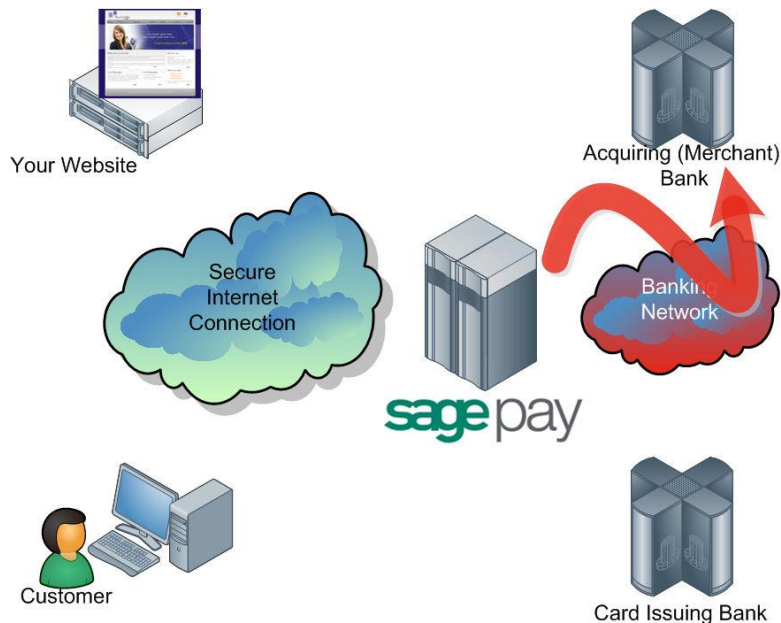
If the transaction was authorised and the Status field contains **OK**, you will also receive a field called **TxAuthNo**. The TxAuthNo field DOES NOT contain the actual Authorisation Code sent by the bank (this is returned in the **BankAuthCode** field) but contains instead a unique reference number to that authorisation that we call the **VPSTxAuthCode**. 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 by the Direct gateway) 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 to confirm payments.



Once per day, from 12:01am, the Sage Pay system batches all authorised card payment transactions for each acquirer and creates a bank specific settlement file.

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

If the file does not transmit correctly, the system tries a further nine times at 10-minute intervals. If all 10 attempts fail the transactions for that bank are rescheduled for inclusion in the following day's batch instead. Sage Pay monitor this process each day to ensure the files have been sent, and if not, the support department will correct the problem during the day to ensure the file is sent correctly that evening (or normally resubmit the file manually the same day to ensure 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 correct any errors and resubmit them for you.

***Important note for PayPal transactions:** PayPal transactions are settled immediately by PayPal. The funds from your customers' PayPal payments are deposited into your PayPal Business account immediately. 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, 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.

Applying Surcharges to Transactions

The ability to apply surcharges based on payment type selected will provide a financial benefit to you by transferring the cost of these transactions to the customer. This is a facility which is only available through using Protocol v3.00 or a later version.

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

How does it work

1. You set up default surcharges for the payment types/currencies you wish to apply them to in MySagePay.
2. Customers select the goods they wish to purchase from your website.
3. They then select the payment type to complete the transaction.
4. Alternatively you can use the SurchargeXML(see Appendix A1) 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: support@sagepay.com.

Direct Payments using PayPal

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

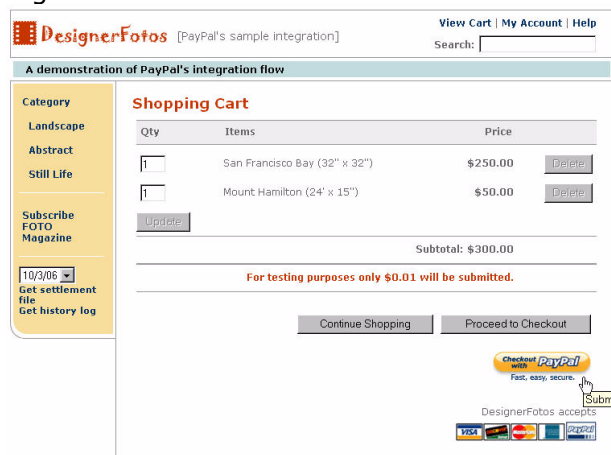
1. The customer shops at your site and fills up a shopping basket with items.
2. At the point the customer wishes to check-out, BEFORE they enter any address or customer details, your site can *optionally* allow the customer to select to pay either with PayPal, or another payment process. This is the “**Express Checkout**” option and should be presented using the orange logo:



To download the PayPal images to use on your own site visit:

<https://www.paypal.com/uk/webapps/mpp/logos>

e.g.



This is optional because you may wish to use PayPal as a different “card type” rather than an alternative payment method, (for example, if you wish to collect customer details on your own pages first). This latter option is described later.

If the customer selects this Express Checkout option, the process jumps ahead to step 6.

3. Since the customer has not selected Express Checkout (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.



(or “PayPal” as a drop down item in a card-type list)

Select Payment Method

Please click below to select the type of card you wish to use.



Credit

2.5 % surcharge for Visa

Please only click the cancel button below if you intend to abort this payment process.

This is called a “**Mark**” integration (as opposed to “Express Checkout” integration above). 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.

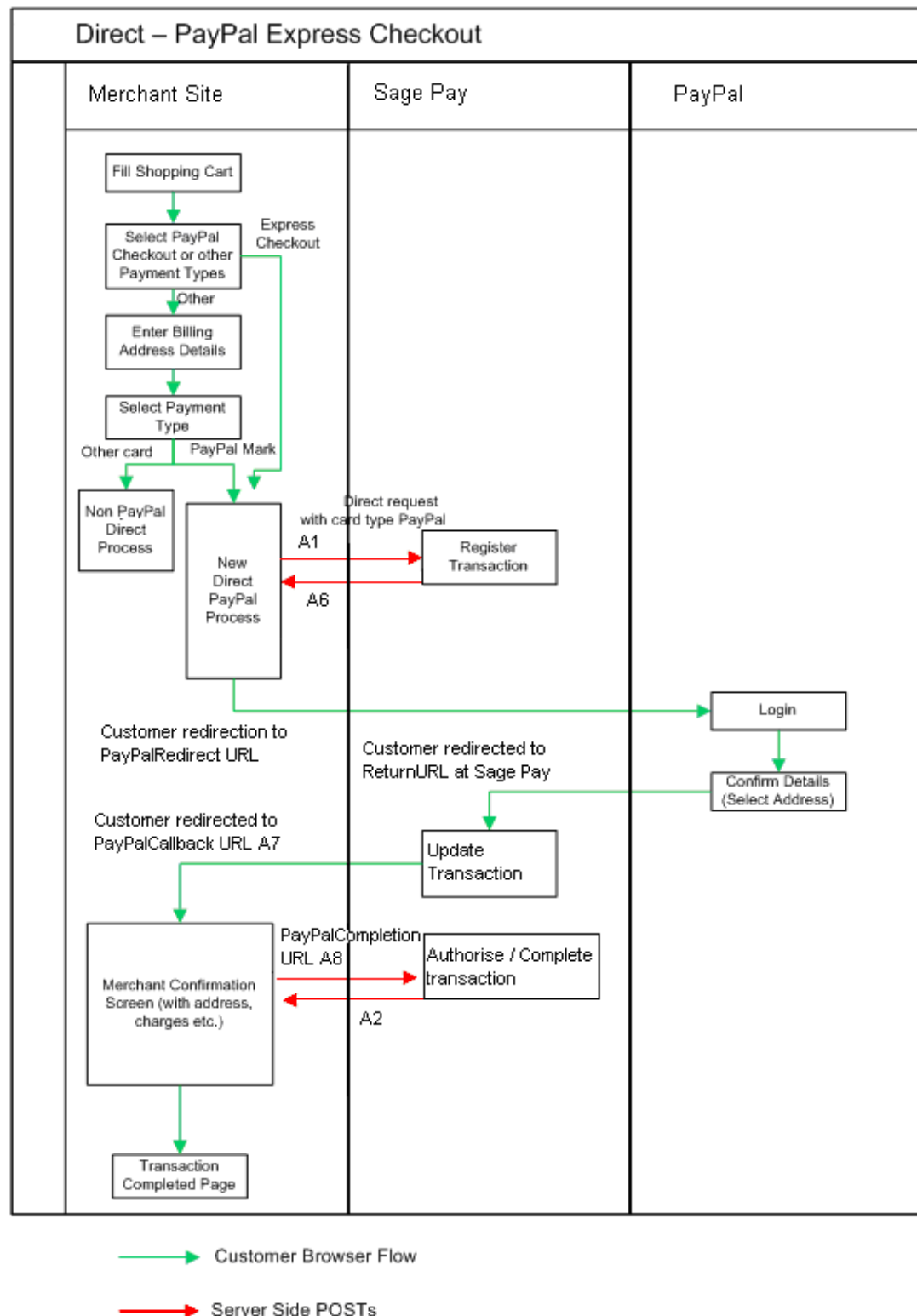
If the customer has selected PayPal, either Mark or Express Checkout, the new process begins at this step.

6. 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 **billingnnnnn** AND **deliverynnnn** 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 [step 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 A6).
9. Your site redirects the customer's browser to the **PayPalRedirectURL** value returned in the Direct response (Appendix A6).
 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 A7. 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 A8) 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 A8) 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.

Direct PayPal Message Flow

The diagram below shows the message and customer flow for a Direct PayPal payment (described above).



Integrating with 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.

N.B: The Sage Pay Simulator cannot be used to test Protocol v3 and you will therefore need to test your integration directly with the Test server.

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 page for viewing your Test transactions can be found here:
<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 Live MySagePay can be accessed from here :
<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>

Stage 1: Testing on the Test Server

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 Admin screens 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.

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 as part of the application process or obtained from the downloads area (www.sagepay.com/help/downloads).

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 A section A1 in the attached protocol) and how to read the response that comes back (section 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 **NOTAUTHED** 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.

3D-Authenticated Transactions

If you plan to support Verified by Visa and MasterCard SecureCode, collectively the 3D-Secure authentication system, you should now go on to test that your scripts can handle these messages. You should ONLY do this once your transaction registration script can successfully process non-authenticated transactions as described in section 3 above.

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 (as described in the step by step transaction process earlier in this document) 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 section 3 above), 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.

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 a PayPal integration with Sage Pay, you will need to create an account at:

<https://developer.paypal.com/>

This account will need 2 test accounts:

- one for a business user
- one for a test customer

The business account will need to be configured to grant API permission to the following API account:

ppdev_1256915571_biz_api1.sagepay.com

(Please note that this is different to the live API account)

NB: To test your PayPal integration you will need to log into your Test MySagePay area and add the userID which corresponds to the business user created above. The support email address provided in your account settings MUST be the same as your PayPal primary email address. The values need to match for registration to complete successfully. You can amend the support email address after PayPal has been set up if required. During this process, you must remain logged into your developer.PayPal.com account. When prompted for customer details, the test customer user credentials created above must be used.

Browse to your Direct integration kit and proceed to the checkout page. The kits present both Express Checkout and Mark options:





ASP.Net Integration Kit

Transaction Registration Page

Enter Card Details


Card Type:	<input type="text" value="PayPal"/>	(Edit to those card you can accept)
Card Number:	<input type="text"/>	(With no spaces or separators)
Card Holder Name:	<input type="text"/>	
Start Date:	<input type="text"/>	(Where available. Use MMY format e.g. 0207)
Expiry Date:	<input type="text"/>	(Use MMY format with no / or - separators e.g. 1109)
Issue Number:	<input type="text"/>	(Older Switch cards only. 1 or 2 digits as printed on the card)
Card Verification Value:	<input type="text"/>	(Additional 3 digits on card signature strip, 4 on Amex cards)
Store For Future:	<input type="checkbox"/>	(Use Sage Pay Token system)

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 (see below):

Your code should store the VPSTxId and redirect the customer's browser to the PayPal Sign In page and ensure that your script can handle a PAYPALOK response (see Appendix A7 below).

The default PayPalCallbackURL code in the kits shows the results of the PayPal Callback Post from Direct after the customer has completed PayPal Authentication, and gives you the option of rejecting the transaction based on the results returned.



Resources

-  read and review the Direct ASP .Net Integration Kit readme document
-  visit the Sage Pay support website and view the online Form Integration guide
-  visit the Sage Pay support website and download the Form protocol and integration guide
-  if you're having trouble, e-mail the technical support team. Please include your Vendor Name in the mail.

Direct ASP.Net Integration Kit

PayPal Callback Page


This page shows the results of the PayPal callback POST from Direct after the customer has completed PayPal Authentication. Because you are in SIMULATOR mode, you are seeing this information and having to click Proceed to complete the transaction. In LIVE mode, this would happen invisibly without user involvement, and the customer would simply be informed about the outcome of the transaction completion.

Direct returned a Status of PAYPALOK

☒ Reply from Direct

Status:	PAYPALOK
StatusDetail:	Confirmed Details at PayPal
VPSTxId:	(P889624C-128E-4B27-8EC7-D782E30CCB87)
AddressStatus:	CONFIRMED
PayerStatus:	UNVERIFIED
DeliverySurname:	Proth
DeliveryFirstnames:	Jimmy
DeliveryAddress1:	88
DeliveryAddress2:	High Street
DeliveryCity:	London
DeliveryPostCode:	W4 12A
DeliveryCountry:	GB
DeliveryState:	
DeliveryPhone:	0123456789
CustomerEmail:	
PayerID:	paypaluser

☒ Order Details stored in your Database

Quantity	Image	Description	Price
1		IronMan	8.75 GBP

[< Back](#)
[Proceed >](#)

This page does not necessarily have to be displayed to your customer; you can choose to either handle the response behind the scenes, or display a page to the shopper confirming the details of the transaction. If required, this page also allows you to change the amount sent in the original POST in Appendix A1 by +/- 15% of the original value (for example, if the delivery price changes as a result of the address selected).

In the example above, the AddressStatus returned was CONFIRMED and the PayerStatus returned was VERIFIED. As the customer has successfully completed the PayPal verification process to help establish the shopper's identity, and their address has been reviewed by PayPal and found highly likely to be that of the User to which it is associated, this is a strong indication that this is a genuine shopper. 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 A8).

If the address was not confirmed, and the payer not verified, for example, you may not wish continue without **PayPal Seller Protection***. 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 A8).

* **PayPal's Seller Protection Policy** can protect sellers from claims, chargebacks and reversals. PayPal will reimburse you for a specified amount if you meet the requirements set out in their Terms and Conditions of the User Agreement. Please visit the link below for further information about Seller Protection:

Please note that you will need to select the agreements for your country if you are not UK based.

https://cms.paypal.com/uk/cgi-bin/?cmd=_render-content&content_ID=ua/Legal_Hub_full&locale.x=en_GB

NB: Sage Pay provides example code in several scripting languages to aid developers integrate our Direct product with PayPal. Within the kits there is a file called paypalCallback. This file provides a basic working example of completing a successful PayPal transaction. By default, the Amount of the transaction is unchanged from the original value, and the value returned in the Accept field is always "YES", regardless of the results returned to you in Appendix A7. You can amend your own code and use the results returned to determine whether you wish to proceed with the PayPal transaction.

You should test PayPal Completion messages that will generate the Direct result of your choosing. By sending the Completion POST with Accept=YES, a Status of OK will be returned in the final POST to your servers. A value of Accept=NO will return a Status of NOTAUTHED.

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.

You will always receive an OK message 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 correct Address, Postcode and CV2 digits, the transaction will still authorise, but you will receive NOTMATCHED messages in the AVS/CV2 checks, allowing you to test your rule-bases and fraud specific code.

Card Type	Card Number	Issue	CV2	Address	Post Code
Visa (VISA)	49290000000006		123	88	412
MasterCard (MC)	5404000000000001		123	88	412
Visa Debit / Delta (DELTA)	4462000000000003		123	88	412
UK Maestro (MAESTRO)	5641820000000005	01	123	88	412
International Maestro (MAESTRO)	30000000000000004	N/A	123	88	412
American Express (AMEX)	3742000000000004		123	88	412
Visa Electron (UKE)	4917300000000008		123	88	412
JCB (JCB)	3569990000000009		123	88	412
Diner's Club (DINERS)	3600000000000008		123	88	412
Laser (LASER)	630499000000000044		123	88	412
Debit MasterCard (MCDEBIT)	5573470000000001		123	88	412

If you have 3D-Secure set up on your test account, you can use the MySagePay interface to switch on the checks at this stage to test your 3D-Secure Terminal URL script against a simulation of the 3D-Secure environment.

This simulation creates PAREq and PAREs messages. It does not talk to the real Visa and MasterCard systems though, so no live authentications can occur.

At the Simulated Authentication screens, to successfully authenticate the transaction, enter "**password**" (without the quotes) into the password box. Any other phrase will fail the authentication.

At the 3D-secure callback stage you'll need to change your POST to go to:

<https://test.sagepay.com/gateway/service/direct3dcallback.vsp>

Once your site can handle all Direct status types, on both your transaction registration and 3D-Secure Terminal URL pages, then you've completed your basic Direct integration and can move on to testing your site against the real Direct Server. If, however, you wish to link in additional processes, such as Refunds or Repeats, or the ability to Release or Abort Deferred transactions, you should continue with testing the additional transaction types.

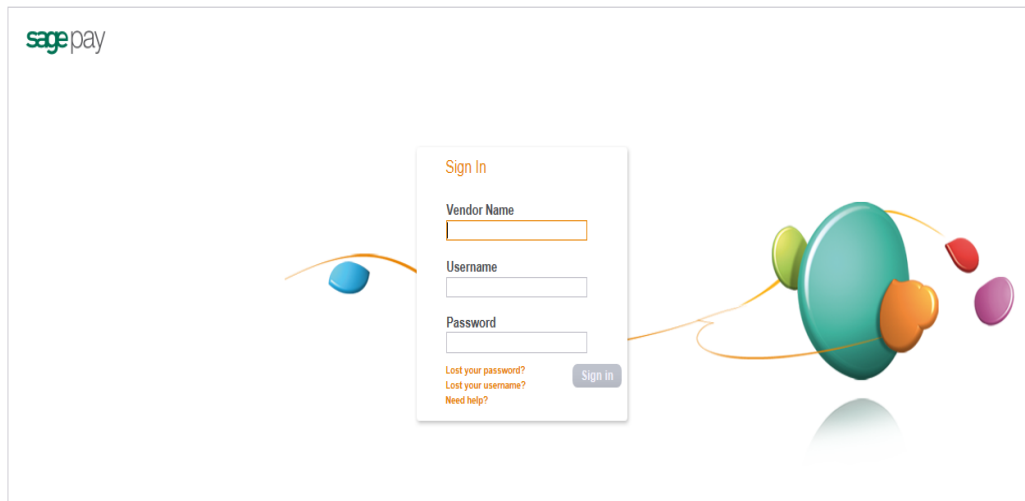
Once you've checked you can process an end-to-end transaction and, where appropriate, can successfully process 3D-Authentication, and tested any additional transaction types you have set up (such as Refunds and Releases) then you are almost ready to go live. Before doing so, however, you should log into your unique MySagePay account on the Test servers to view your transactions and familiarise yourself with the interface.

The Test Server MySagePay interface

A Test Server version of the MySagePay interface 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 when you begin to take real payments.

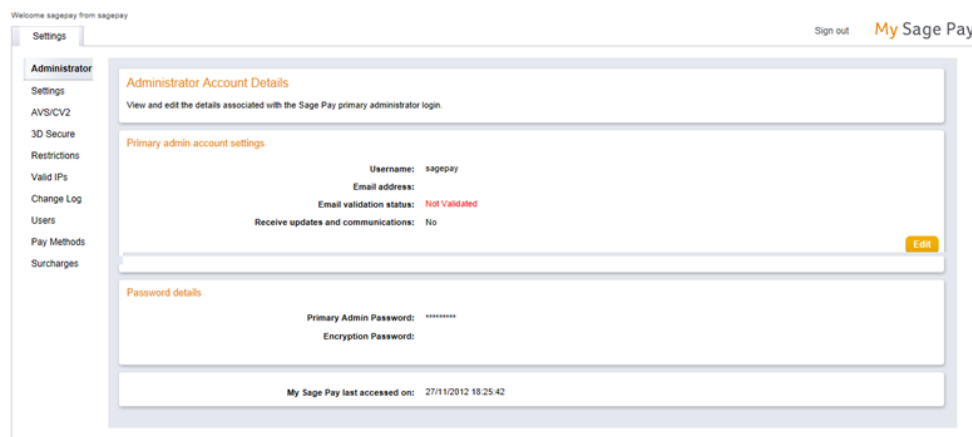
The Test Server MySagePay can be found at:

<https://test.sagepay.com/mysagepay>



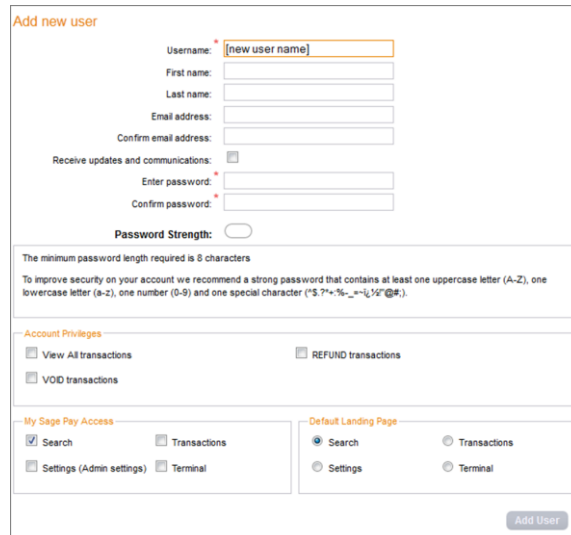
When you log in to the MySagePay screens you will be asked for a **Vendor Name**, a **User Name** and a **Password**. The first time you log in you will need to do so as your system Administrator:

- In the **Vendor Name** box, enter your Vendor Name, as selected in your Online Registration screens and used throughout the development as your unique merchant identifier.
- In the **User Name** box, enter the Vendor Name **again**.
- In the **Password** box, enter the Admin password as supplied to you by Sage Pay when your test account was set up.
- Click **Login** and you will see the settings section (below).



The administrator can ONLY access the Settings Tab. You cannot, whilst logged in as administrator, view your transactions or take 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. Click on the user tab on the left to create a new user and you will be presented the screen below:



Enter a username for yourself and a password you'll remember, and then ensure all the check boxes are enabled for your account. Click the Add User button and your new User account will appear in the list.

Now click the sign out button in the top right hand corner and click to Log back in, this time entering:

- Your Vendor name in the **Vendor Name** box.
- The User Name of the account you just created in the **User Name** box.
- The password for the 'user' account you just created in the **Password** box.

...and click **Login**.

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 log 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 the MySagePay admin area can be found in the online help centre (www.sagepay.com/help) or you can watch a video demo available in the demo area (www.sagepay.com/help/demos). Play with the system until you are comfortable with it; you cannot inadvertently charge anyone or damage anything whilst on the Test server.

Additional Transaction Types

DEFERRED transactions.

By default a PAYMENT transaction type is used in your scripts to gain an authorisation from the bank, 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, but merely place a "shadow" on the customer's card to ensure they cannot subsequently spend those funds elsewhere, and then only 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 normal PAYMENT. You just need to change your script to send a TxType of DEFERRED when you register the transaction (Appendix A1) 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** message to our servers (see the "Server and Direct Shared Protocols" document for more detail) or by logging into the MySagePay interface, finding the transaction and clicking the Release button.

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 (according to card scheme rules). After that the shadow may disappear from the card 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 (see below).

DEFERRED transactions remain available for RELEASE for up to 30 days. After that time they are automatically ABORTed by the Sage Pay systems.

Additional notes for using Deferred/Release with PayPal transactions

Unlike a normal Sage Pay DEFERRED transaction, no shadow is placed on the customer's card 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 after obtaining a successful release.

REPEAT payments

If you have already successfully authorised a customer's card using a PAYMENT, a released DEFERRED or an AUTHORISE (see below) 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 "Continuous Authority" merchant number from your bank (please contact your acquiring bank for further details), but ad-hoc REPEATs do not require a Continuous Authority merchant number. REPEAT payments cannot be 3D-Secured, or have CV2 checks performed on them (unless you supply this number again. Sage Pay are not allowed to store CV2 numbers) so you are better to make use of Authenticate and Authorise if you need to vary the transaction amount on a regular basis.

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

Additional notes for using Repeat Payments with PayPal transactions

You can only REPEAT a PayPal transaction if the initial transaction was set up as a PayPal Reference transaction (with BillingAgreement set to 1. See the Appendix for details).

AUTHENTICATE and AUTHORISE

The AUTHENTICATE and AUTHORISE methods are specifically for use by merchants who are either (i) unable to fulfil the majority of orders in less than 6 days (or sometimes need to fulfil them after 30 days) or (ii) 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 authorisation.

Your site will register your transaction with a TxType of AUTHENTICATE. Direct will contact the 3D-Secure directories to check if the card is part of the scheme. If it is not, then the card details are simply held safely at Sage Pay and Direct will reply with 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 for that transaction).

Additional notes for using Authenticate and Authorise with PayPal transactions

You can use the Authenticate and Authorise Payment Type but the transaction will only ever be **REGISTERED** (because the transaction will never be 3D Secured). Similarly to Releasing a Deferred transaction, we recommend you to Authorise the transaction via the MySagePay area when you are ready to ship the goods and take the funds.

If, however, the card *is* part of the 3D-Secure scheme, Direct will reply with a Status of **3DAUTH** along with the MD, PAREq and ACSURL. You must then redirect

the customer to their card issuing bank for authentication (just like a normal 3D-Secure payment, see steps 5 and 6 in the Payment Process above). Here they will authenticate themselves and be returned to your TermUrl along with the PAREs and MD. These you'll forward to the Direct 3D Callback page and Sage Pay will decode the response for you.

If the customer has not passed authentication, your rule base is consulted to check if they can proceed for authorisation anyway. If not, Direct replies with a Status of **REJECTED**. If the customer failed authentication but can proceed, Direct replies with a **REGISTERED** Status. If the user passed authentication with their bank and a CAVV/UCAF value is returned, Direct sends a Status of **AUTENTICATED** and a **CAVV** value 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 an **AUTHORISE** or **CANCEL** request. This can be sent as a message to our servers (see the "Server and Direct Shared Protocols" document for more detail) or by logging into MySagePay, finding the transaction and performing the required action.

To charge the customer when you are ready to fulfil the order, your site will need to send an **AUTHORISE** request. You can Authorise any amount up to 115% of the value of the original Authentication, and use any number of Authorise requests against an original Authentication so long as the total value of those authorisations does not exceed the 115% limit, and the requests are inside the 90 days limit. This is the stage at which your acquiring bank is contacted for an auth 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, so will still receive the fraud liability shift.

When you have completed all your Authorisations, or if you do not wish to take any, you can send a **CANCEL** message to our Server to archive away the Authentication and prevent any further Authorisations being made against the card. This happens automatically after 90 days.

REFUNDS and VOIDS

Once a PAYMENT, AUTHORISE or REPEAT transaction has been authorised, or a DEFERRED transaction has been RELEASED, it will be settled with the acquiring bank early the next morning and the funds will be moved from the customer's card account, across 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 send a **VOID** message to our servers to prevent the transaction ever being settled (see the "Server and Direct Shared Protocols" document for more detail), thus saving you your transaction charges and the customer from ever being charged. You can also VOID transactions through the MySagePay Admin interface. VOIDed transactions can NEVER be reactivated though, so use this functionality carefully.

Once a transaction has been settled, however, you can no longer VOID it. If you wish to return funds to the customer you need to send a **REFUND** message to our servers, or use the MySagePay Admin screens to do the same.

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. Again, the REFUND protocol can be found in the "Server and Direct Shared Protocols" document.

Please note that the Sage Pay gateway now archives all transactions that are more than 2 years old and we therefore recommend that you check the date of the original transaction which you wish to refund before processing

Additional notes for using Refunds and Voids with PayPal transactions

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

For information regarding registering additional transaction types using HTTPS POSTS, please refer to the Server and Direct Shared Protocols Guide, which can be obtained from the developer's area on our website:

www.sagepay.com/help/downloads

Stage 2: Going Live

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 testing your account before you are granted access to your Live account. Details can be found below:

www.sagepay.com/help/faq/processes_to_go_live_how_to_start_accepting_payments_from_your_customers

NB – Without confirmation from the bank and without 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 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 your own valid credit or debit card. If you receive an authorisation code, then everything is working correctly.

Finally should then log into the Live Server MySagePay screens at <https://live.sagepay.com/mysagepay> and in a similar manner to the Test Server, first log in as the Administrator, then create a Live System User account for yourself, log in as that user, locate your test transaction and VOID it, so you are not charged for the transaction. At this stage the process is complete.

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.

Congratulations, you are Live with the Sage Pay.

Well done! Hopefully the process of getting here was as painless and hassle free as possible. You'll be pleased to know that now you are live we don't cut the strings and run away. You should contact us with any transaction queries that arise or for any help you need with the MySagePay system.

Here are the best ways to reach us and the best people to reach:

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

Appendix A - The Sage Pay Direct 3.00 Protocol

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

A1: Transaction Registration

This is performed via a **HTTPS POST** request, sent to the initial Direct Payment URL. The details should be URL encoded Name=Value fields separated by '&' characters.






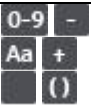


Request format

Name	Mandatory	Format	Max Length	Allowed Values	Description
VPSProtocol	Yes	0-9 .	4 Chars	Must be 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 and Direct Shared Protocols" for other transaction types (such as Refund, Releases, Aborts and Repeats). TxType should be in capital Letters.
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. Vendor Login Name
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. NB : 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. NB : This amount should be the transaction amount excluding any surcharge applied. This will be added to the total value by Sage Pay
Currency	Yes	ISO4217	3 Chars		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		The description of goods purchased for your information in the MySagePay screens
	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 MUST be in MMY format i.e. 1215 for December 2015. No / or - characters should be included. Not required if CardType=PAYPAL
	CV2	No	0-9	3 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 NB: 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	MC is MasterCard, UKE is Visa Electron. MAESTRO is both UK and International Maestro. AMEX and DC (DINERS) may only be accepted if you have additional merchant accounts with those acquirers.

	BillingSurname	Yes	<div> <div>Aa á</div> <div>/\</div> <div>& -</div> <div>'</div> </div>	20 chars		<p>N.B: All mandatory fields must contain a value, apart from the postcode. The postcode 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.</p>
	BillingFirstnames	Yes	<div> <div>Aa á</div> <div>/\</div> <div>& -</div> <div>'</div> </div>	20 chars		
	BillingAddress1	Yes	<div> <div>Aa á</div> <div>0-9</div> <div>+ ' / \ & : , -</div> <div>CR / LF</div> <div>()</div> </div>	100 Chars		
	BillingAddress2	No	<div> <div>Aa á</div> <div>0-9</div> <div>+ ' / \ & : , -</div> <div>CR / LF</div> <div>()</div> </div>	100 Chars		
	BillingCity	Yes	<div> <div>Aa á</div> <div>0-9</div> <div>+ ' / \ & : , -</div> <div>CR / LF</div> <div>()</div> </div>	40 Chars		

	BillingPostCode	Yes	Aa 0-9 -	10 Chars		N.B: All mandatory fields must contain a value, apart from the postcode. The postcode 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.
	BillingCountry	Yes	ISO3166	2 Chars	ISO 3166-1 country code	
	BillingState	No	US	2 Chars	State code for US customers only*	
	BillingPhone	No	0-9 - Aa + ()	20 Chars		
	DeliverySurname	Yes	Aa á /\ & - '	20 Chars		
	DeliveryFirstnames	Yes	Aa á /\ & - '	20 Chars		
	DeliveryAddress1	Yes	Aa á 0-9 + ' /\ & : , - ' CR / LF ()	100 Chars		
	DeliveryAddress2	No	Aa á 0-9 + ' /\ & : , - '	100 Chars		

						
	DeliveryCity	Yes		40 Chars		
	DeliveryPostCode	Yes		10 Chars		
	DeliveryCountry	Yes		2 Chars	ISO 3166-1 country code of the cardholder's delivery address	
	DeliveryState	No		2 Chars	State code for US customers only*	
	DeliveryPhone	No		20 chars		
	PayPalCallbackURL	No		255 Chars		<p>Fully qualified domain name of the URL to which customers are redirected upon completion of a PayPal transaction.</p> <p>Must begin http:// or https:// Only required if the CardType=PAYPAL.</p>
	CustomerEMail	No		255 Chars		<p>The customer's email address.</p> <p>NOTE: If you wish to use multiple email addresses, you should add them using the : (colon) character as a separator. e.g. me@mail1.com:me@mail2.com</p> <p>The current version of the Direct integration method</p>

						does not send confirmation emails to the customer. This field is provided for your records only.
	Basket	No	<HTML>	7500 Chars	See A1.2	You can use this field to supply details of the customer's order. This information will be displayed to you in MySagePay. If this field is supplied then the BasketXML field should not be supplied.
	AllowGiftAid	No	0-9		0 1	0 = No Gift Aid Box displayed (default) 1 = Display Gift Aid Box This flag allows the gift aid acceptance box to appear for this transaction on the payment page. This only applies if your vendor account is Gift Aid enabled.
	ApplyAVSCV2	No	0-9		0 1 2 3	Using this flag you can fine tune the AVS/CV2 checks and rule set you've defined at a transaction level. This is useful in circumstances where direct and trusted customer contact has been established and you wish to override the default security checks. 0 = If AVS/CV2 enabled then check them. If rules apply, use rules. (default) 1 = Force AVS/CV2 checks even if not enabled for the account. If rules apply, use rules. 2 = Force NO AVS/CV2 checks even if enabled on account. 3 = Force AVS/CV2 checks even if not enabled for the account but DON'T apply any rules. This field is ignored for PAYPAL and LASER 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	1 Char	0 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

						<p>and you wish to override the default security checks.</p> <p>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.</p> <p>This field is ignored for PAYPAL and LASER transactions</p>
	AccountType	No	Aa	1 Char	E M C	<p>This optional flag is used to tell the Sage Pay System which merchant account to use. If omitted, the system will use E, then M, then C by default.</p> <p>This field is ignored for PAYPAL transactions 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).</p>
	BillingAgreement	No	0-9	1 Char	0 1	<p>This field must be set for PAYPAL REFERENCE 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 Billing Agreement field and a value of 1 for successful repeat payments. All non-PayPal transactions can be repeated without this flag. 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 REPEAT.</p> <p>If you wish to register this transaction as the first in a series of regular payments, this field should be set</p>

						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.
	CreateToken	No	0-9	1 Char	0 1	<p>0 = This will not create a token from the payment. 1 = This will create a token from the payment if successful and return a token ID.</p> <p>This can be used on a PAYMENT, AUTHENTICATE or DEFERRED txtype.</p>
	StoreToken	Yes	0-9	1 Char	0 1	<p>0 = Do not store token (this is the default)</p> <p>1 = Store a token after three failed attempts or after a successful payment .</p> <p>To store a token repeatedly a value of 1 must be passed with every use of the token.</p>
	BasketXML	No		20000 chars	See A1.3	A more flexible version of the current basket field which can be used instead of. 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 default surcharge settings in MySagePay for the current transaction. Percentage and fixed amount surcharges can be set for different payment types.
	VendorData	No	0-9 Aa	200 Chars		Use this to pass data you wish to be displayed against the transaction in MySagePay.
	ReferrerID	No	Aa a 0-9 + /\ & : - CR / LF	40 Chars		This can be used to send the unique reference for the Partner that referred the Vendor to Sage Pay.

			()			
	Language	No	ISO639	2 Chars		The language the customer sees the payment pages in is determined by the code sent here. If this is NULL 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.

A1.1 SurchargeXML field

Use this field to override the default surcharge set in MySagePay for the current transaction. You can set a different surcharge value for each payment type (except PayPal).
NB : 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 value can either be a percentage or fixed amount.

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

Summary of Surcharge XML elements

		Node/Element	Mandatory	Format	Max Length	Allowed Values	Description
		<surcharges>	No	Node			The root element for all other surcharge elements.
		L<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.
		L<paymentType>	Yes	Aa	15 Chars	VISA AMEX DELTA JCB DC MC UKE MAESTRO LASER MCDEBIT	The payment type this surcharge element applies to.
		L<percentage>	Yes unless a fixed element is 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.
		L<fixed>	Yes unless a percentage element is supplied	0-9 , -	X digits to decimal places where appropriate		Minor digits should be formatted to two decimal places. 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. NB : 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.

See Appendix A10 - Example Surcharge XML - Example 1

A1.2 Basket Contents

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 n including tax:  
Total cost of item n
```

IMPORTANT NOTES:

- 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).

So, for example, the following shopping cart...

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

Would be represented thus:

```
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:::199.99
```


A1.3 BasketXML field

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:

$\text{unitGrossAmount} = \text{unitNetAmount} + \text{unitTaxAmount}$

$\text{totalGrossAmount} = \text{unitGrossAmount} * \text{quantity}$

$\text{amount (sent in transaction Registration)} = \text{Sum of totalGrossAmount} + \text{deliveryGrossAmount} - \text{Sum of fixed (discounts)}$

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

Summary of Basket XML elements

	Node/Element	Mandatory	Format	Max Length	Allowed Values	Description
	<basket>	No	Node			The root element for all other basket elements.
	L<agentId>	No	Aa 0-9	1-16 Chars		The ID of the seller if using a phone payment.
	L<item>		Xml container element			There can be as many Items as you like in the Basketxml, each holding a different item and recipient. The sum of all totalGrossAmounts in all item elements and the deliveryGross amount must match the amount field sent with the transaction
	L<description>	Yes	Aa 0-9 á + / \ & : - CR / LF { }	100 Chars		Description of the item
	L<productSku>	No	Aa 0-9 á + / \ & : - CR / LF { }	1-12 Chars		Item SKU. This is your unique product identifier code.
	L<productCode>	No	Aa 0-9 á + / \ & : - CR / LF { }	1-12 Chars		Item product code.

			CR / LF { }			
	L<quantity>	Yes	0-9	12 Chars		Quantity of the item ordered
	L<unitNetAmount>	Yes	0-9 .	14 digits to 2 decimal places		<p>The cost of the item before tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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.</p> <p>NB : 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.</p>
	L<unitTaxAmount>	Yes	0-9 .	14 digits to 2 decimal places		<p>The amount of tax on the item. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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.</p> <p>NB : 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.</p>
	L<unitGrossAmount>	Yes	0-9 .	14 digits to 2 decimal places		<p>The total cost of the item with tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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.</p> <p>NB : 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.</p>
	L<totalGrossAmount>	Yes	0-9 .	14 digits to 2 decimal places		<p>The total cost of the line including quantity and tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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.</p> <p>NB : Amounts must be in the UK currency format. The period must be used to indicate the decimal</p>

						place. The comma must only be used to separate groups of thousands.
	L<recipientFName>	No	<div> <div>Aa</div> <div>0-9</div> <div>á</div> <div>+</div> <div>'</div> <div>/\</div> <div>&</div> <div>:</div> <div>,</div> <div>-</div> <div>-</div> <div>CR / LF</div> <div>()</div> </div>	1-20 Chars		The first name of the recipient of this item.
	L<recipientLName>	No	<div> <div>Aa</div> <div>0-9</div> <div>á</div> <div>+</div> <div>'</div> <div>/\</div> <div>&</div> <div>:</div> <div>,</div> <div>-</div> <div>-</div> <div>CR / LF</div> <div>()</div> </div>	1-20 Chars		The last name of the recipient of this item.
	L<recipientMName>	No	<div> <div>Aa</div> <div>0-9</div> <div>á</div> <div>+</div> <div>'</div> <div>/\</div> <div>&</div> <div>:</div> <div>,</div> <div>-</div> <div>-</div> <div>CR / LF</div> <div>()</div> </div>	1 Char		The middle initial of the recipient of this item.
	L<recipientSal>	No	<div> <div>Aa</div> <div>0-9</div> <div>á</div> <div>+</div> <div>'</div> <div>/\</div> <div>&</div> <div>:</div> <div>,</div> <div>-</div> <div>-</div> <div>CR / LF</div> <div>()</div> </div>	2-4 Chars		The salutation of the recipient of this item.

			{ }			
	L<recipientEmail>	No	RFC532N	1-45 Chars		The email of the recipient of this item.
	L<recipientPhone>	No	Aa 0-9 + - { }	1-20 Chars		The phone number of the recipient of this item.
	L<recipientAdd1>	No	Aa 0-9 ä + * / \ & : , - - CR / LF { }	1-100 Chars		The first address line of the recipient of this item.
	L<recipientAdd2>	No	Aa 0-9 ä + * / \ & : , - - CR / LF { }	1-100 Chars		The second address line of the recipient of this item.
	L<recipientCity>	No	Aa 0-9 ä + * / \ & : , - - CR / LF { }	1-40 Chars		The city of the recipient of this item.
	L<recipientState>	No	US	Exactly 2 chars		If in the US, the 2 letter code for the state of the recipient of this item.

	L<recipientCountry>	No	ISO3166	Exactly 2 chars		The 2 letter country code (ISO 3166) of the recipient of this item.
	L<recipientPostCode>	No	Aa 0-9 -	1-9 Chars		The postcode of the recipient of this item.
	L<itemShipNo>	No	Aa 0-9	1-19 Chars		The shipping item number.
	L<itemGiftMsg>	No	Aa	1-160 Chars		Gift message associated with this item.
	L<deliveryNetAmount>	No	0-9 .	14 digits to 2 decimal places		The total cost of the line including quantity and tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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. NB : 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.
	L<deliveryTaxAmount>	No	0-9 .	14 digits to 2 decimal places		The total cost of the line including quantity and tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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. NB : 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.
	L<deliveryGrossAmount>	No	0-9 .	14 digits to 2 decimal places		The total cost of the line including quantity and tax. Must be positive and numeric, and may include a decimal place where appropriate. Minor digits should be formatted to two decimal places. 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. NB : 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.
	L<discounts>	No	XML Container			The root element for all other discount elements.

			element			
	L<discount>	Yes	XML Container element			There can be multiple discount elements.
	L<fixed>	Yes	0-9 -	14 digits to 2 decimal places	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
	L<description>	No	Aa 0-9 + ' /\n : - { } _ @ () " ~ [] \$ = ! # ?	100 Chars		This is the description of the discount. This will appear on the payment pages, MySagePay and the PayPal checkout pages if appropriate.
	L<shipId>	No	Aa 0-9	1-16 Chars		The ship customer ID.
	L<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.
	L<shippingFaxNo>	No	Aa 0-9 - () +	1-20 Chars		The Fax Number

	L<hotel>	No	Xml container element			Used to provide hotel information for settlement. There can be only one hotel element.
	L<checkIn>	Yes	DATE			Check in date for hotel.
	L<checkout>	Yes	DATE			Check out date for hotel.
	L<numberInParty>	Yes	0-9	3 Chars		Number of people in the hotel booking.
	L<folioRefNumber>	No	Aa á 0-9 + /\ & : - ()	10 Chars		Folio reference number for hotel.
	L<confirmedReservation>	No	Aa		Y N	Flag to indicate whether a guest has confirmed their reservation Y= Confirmed Reservation N = Unconfirmed Reservation
	L<dailyRoomRate>	Yes	Aa á 0-9 + /\ & : - () CR / LF	15 Chars		Daily room rate for the hotel.
	L<guestName>	Yes	Aa á 0-9 + /\ & : - ()	1-20 Chars		Name of guest

			() CR / LF			
	L<cruise>	No	Xml container element			Used to provide cruise information for settlement. There can be only one cruise element.
	L<checkIn>	Yes	DATE			start date for cruise.
	L<checkout>	Yes	DATE			End date for cruise.
	L<carRental>	No	Xml container element			Used to provide car rental information for settlement. There can be only one car rental element.
	L<checkIn>	Yes	DATE			Check in date for car rental.
	L<checkout>	Yes	DATE			Check out date for car rental.
	L<tourOperator>	No	Xml container element			Used to provide tour operator information for settlement. There can be only one tour operator element.
	L<checkIn>	Yes	DATE			Check in date for tour operator.
	L<checkout>	Yes	DATE			Check out date for tour operator.
	L<airline>	No	Xml container element			Used to provide airline information for settlement. There can be only one airline element
	L<ticketNumber>	Yes	Aa 0-9	1-11 Chars		The airline ticket number
	L<airlineCode>	Yes	0-9	3 Chars		IATA airline code
	L<agentCode>	Yes	0-9	8 Chars		IATA agent code
	L<agentName>	Yes	Aa 0-9	1-26 Chars		Agency name
	L<flightNumber>	No	Aa 0-9	6 Chars		Flight number
	L<restrictedTicket>	Yes	BOOLEAN			Can be 0,1, true or false.
	L<passengerName>	Yes	Aa 0-9	1-29 Chars		Name of passenger
	L<orginatingAirport>	Yes	Aa	3 Chars		IATA airport code
	L<segment>	Yes	Xml container element			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.

L<carrierCode>	Yes	Aa	3 Chars	IATA carrier code
L<class>	Yes	Aa 0-9	3 Chars	Class of service
L<stopover>	Yes	BOOLEAN		Can be 0,1, true or false to indicate a stopover
L<legDepartureDate>	Yes	DATE		Departure date of the segment.
L<destination>	Yes	Aa	3 Chars	IATA airport code of destination
L<fareBasis>	No	Aa 0-9	1-6 Chars	Fare basis code
L<customerCode>	No	Aa 0-9	1-20 Chars	Airline customer code
L<invoiceNumber>	No	Aa 0-9	1-15 Chars	Airline Invoice Number
L<dinerCustomerRef>	No	Aa 0-9	1-15 Chars	Diners customer reference Can include up to 5 elements

See Appendix A10- Example Basket XML - Example 2

A1.4 CustomerXML field

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.

Summary of Customer XML elements

Node/Element	Mandator y	Format	Max Length	Allowed Values	Description
customer	No	Node	1 Char		The root element for all other customer elements.
L<customerMiddleInitial>	No	Aa 0-9 + /\ & : - CR / LF			The middle initial of the customer.

			()			
	L<customerBirth>	No	DATE	11-19 Chars		The date of birth of the customer.
	L<customerWorkPhone>	No	0-9 - Aa + ()	11-19 Chars		The work phone number of the customer.
	L<customerMobilePhone>	No	0-9 - Aa + ()			The mobile number of the customer.
	L<previousCust>	No	BOOLEAN			Whether the customer is a previous customer or new.
	L<timeOnFile>	No	0-9	1-16 Chars	Min Value 0	The number of days since the card was first seen.
	L<customerId>	No	Aa 0-9	1 Char		The ID of the customer

See Appendix A10 - Customer XML - Example 3





A2: Sage Pay Response to the Registration or Callback POSTs

This is the plain text response part of the POST originated by your servers in A1 (or A4 for 3D-Secure transactions). 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	This will match the protocol version supplied in A1.
	Status	Yes	Aa 0-9	15 Chars	OK AUTHENTICATED REGISTERED NOTAUTHED REJECTED 3DAUTH PPREDIRECT MALFORMED INVALID ERROR	<p>OK – Transaction completed successfully with authorisation.</p> <p>AUTHENTICATED – The 3D-Secure checks were performed successfully and the card details secured at Sage Pay.</p> <p>REGISTERED – 3D-Secure checks failed or were not performed, but the card details are still secured at Sage Pay.</p> <p>NOTAUTHED – The transaction was not authorised by the acquiring bank. No funds could be taken from the card. A NOTAUTHED status is also returned for PayPal transactions in response to the PayPal Completion Post (if Accept=NO was sent to complete PayPal transaction, see Appendix A8).</p> <p>REJECTED – The SAGE PAY System rejected the transaction because of the rules you have set on your account.</p> <p>3DAUTH – The customer needs to be directed to their card issuer for 3D-Authentication. GO TO APPENDIX A3.</p> <p>PPREDIRECT – Customer needs to be redirected to PayPal GO TO APPENDIX A6</p> <p>MALFORMED – Input message was missing fields or badly formatted – normally will only occur during development and vendor integration.</p> <p>INVALID – Transaction was not registered because although the</p>

					<p>POST format was valid, some information supplied was invalid. e.g. incorrect vendor name or currency.</p> <p>ERROR – A problem occurred at Sage Pay which prevented transaction completion. Please notify Sage Pay if a Status report of ERROR is seen, together with your VendorTxCode and the StatusDetail text.</p> <p>If the status is not OK, the StatusDetail field will give more information about the problem.</p> <p>AUTHENTICATED and REGISTERED statuses are only returned if the TxType is AUTHENTICATE.</p> <p>3DAUTH is only returned if 3D-Authentication is available on your account AND the directory services have issued a URL to which you can progress. A Status of 3DAUTH only returns the StatusDetail, 3DSecureStatus, MD, ACSURL and PAREq fields. The other fields are returned with other Status codes only. See Appendix 3.</p>
	StatusDetail	Yes	Aa 0-9	255 Chars	Human-readable text providing extra detail for the Status message. You should always check this value 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. Not present if Status is 3DAUTH
	SecurityKey	Yes	0-9 Aa	10 Chars	A Security key which Sage Pay uses to generate a MD5 Hash to sign the transaction. This value is used to allow detection of tampering with notifications from the Sage Pay Server. It must be kept secret from the Customer and held in your database. Not present if Status is 3DAUTH or PPREDIRECT .
	TxAuthNo	Yes	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	<p>ALL MATCH SECURITY CODE MATCH ONLY, ADDRESS MATCH ONLY, NO DATA</p> <p>This is the response from AVS and CV2 checks. Provided for Vendor info and backward compatibility with the banks. Rules set up at the Sage Pay server will accept or reject the transaction based on these values.</p> <p>More detailed results are split out in the next three fields. Not</p>

					MATCHES or DATA NOT CHECKED.	present if the Status is 3DAUTH, AUTHENTICATED, PPREDIRECT or REGISTERED.
	AddressResult	Yes		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		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		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		50 Chars	OK NOTCHECKED NOTAUTHED INCOMPLETE ERROR ATTEMPTONLY NOAUTH CANTAUTH MALFORMED INVALID	<p>This field details the results of the 3D-Secure checks (where appropriate)</p> <p>OK – The 3D-Authentication step completed successfully. If the Status field is OK too, then this indicates that the authorized transaction was also 3D-authenticated and a CAVV will be returned. Liability shift occurs.</p> <p>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.</p> <p>NOAUTH – This means the card is not in the 3D-Secure scheme.</p> <p>CANTAUTH - This normally means the card Issuer is not part of the scheme.</p> <p>NOTAUTHED – The cardholder failed to authenticate themselves with their Issuing Bank.</p> <p>NOTCHECKED - No 3D Authentication was attempted for this transaction. Always returned if 3D-Secure is not active on your account.</p> <p>INCOMPLETE – 3D-Secure authentication was unable to complete (normally at the card issuer site). No authentication occurred.</p>

						<p>MALFORMED,INVALID,ERROR – These statuses indicate a problem with creating or receiving the 3D-Secure data. These should not occur on the live environment.</p> <p>If the status is not OK, the StatusDetail field will give more information about the status.</p> <p>AUTHENTICATED and REGISTERED statuses are only returned if the TxType is AUTHENTICATE.</p> <p>Please notify Sage Pay if a Status report of ERROR is seen, together with your VendorTxCode and the StatusDetail text.</p> <p>3DAUTH is only returned if 3D-Authentication is available on your account AND the directory services have issued a URL to which you can progress. A Status of 3DAUTH only returns the StatusDetail, 3DSecureStatus, MD, ACSURL and PAREq fields. The other fields are returned with other Status codes only. See Appendix 3.</p>
	CAVV	No	Aa 0-9	32 Chars		<p>The encoded result code from the 3D-Secure checks (CAVV or UCAF).</p> <p>Only present if the 3DSecureStatus field is OK or ATTEMPTONLY</p>
	Token	No	0-9 Aa	38 Chars		<p>The token generated by Sage Pay and provided in response to the registration phase.</p> <p>A GUID will be returned</p>
	FraudResponse	No	Aa	10 Chars	<p>ACCEPT CHALLENGE DENY NOTCHECKED</p>	<p>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</p>
	Surcharge	No	0-9 , -		0.01 to 100,000.00	<p>Returns the surcharge amount charged and is only present if a surcharge was applied to the transaction.</p>
*	ExpiryDate	Yes	0-9 Aa	4 chars		<p>Expiry date of credit card used, in the format MMY. NB For PayPal "XXXX" is returned</p>
	BankAuthCode	No	0-9 Aa	6 Chars		<p>The authorisation code returned from the bank. e.g T99777</p>
	DeclineCode	No	0-9	2 Chars		<p>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</p>

A3: Sage Pay Response to the 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. A Status of 3DAUTH only returns the StatusDetail, 3DSecureStatus, MD, ACSURL and PAREq fields.
	StatusDetail	Yes	Aa 0-9	255 Chars		Human-readable text providing extra detail for the Status message. You should always check this value if the Status is not OK .
	3DSecureStatus	Yes	Aa 0-9	20 Chars	OK	OK – If the Status field is 3DAUTH, this means the card is part of the scheme. 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. Only present if the Status field is 3DAUTH .
	ACSURL	Yes	RFC1738	7500 Chars		A fully qualified URL that points to the 3D-Authentication system at the Cardholder's Issuing Bank. Only present if the Status field is 3DAUTH .
	PAREq	Yes	BASE64	7500 Chars		A Base64 encoded, encrypted message to be passed to the Issuing Bank as part of the 3D-Authentication. Only present if the Status field is 3DAUTH . NOTE: 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.

A4: 3D-Authentication Results POST from your Terminal URL to Sage Pay.

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.

A5: Response to 3D Callback POSTs

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

A6: Sage Pay Response to the Registration POST (for PayPal transactions)

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
	VPSPProtocol	Yes	0-9 .	4 Chars	3. 00	This will match the protocol version supplied in A1.
	Status	Yes	Aa 0-9	15 Chars	PPREDIRECT	
	StatusDetail	Yes	Aa 0-9	255 Chars	Please redirect the customer to the PayPal URL	Will contain the text in the Allowed Values column
	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.

A7: Sage Pay Callback after PayPal Authentication

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	This will match the protocol version supplied in A1.
	Status	Yes	Aa 0-9	15 Chars	PAYPALOK MALFORMED INVALID ERROR	<p>PAYPALOK – The customer has selected a payment type and the transaction is ready to be taken</p> <p>MALFORMED – Input message was missing fields or badly formatted – normally will only occur during development and vendor integration.</p> <p>INVALID – Transaction was not registered because although the POST format was valid, some information supplied was invalid. e.g. incorrect vendor name or currency.</p> <p>ERROR – A problem occurred at Sage Pay which prevented transaction completion.</p> <p>Please notify Sage Pay if a Status report of ERROR is seen, together with your VendorTxCode and the StatusDetail text.</p>
	StatusDetail	Yes	Aa 0-9	255 Chars		Human readable status or error information
	VPSTxId	Yes	Aa 0-9 { } -	38 Chars		The Sage Pay ID to uniquely identify the Transaction on our system. You will need to use this when confirming the PayPal transaction.

	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.
	AddressStatus	Yes	Aa 0-9	20 Chars	NONE CONFIRMED UNCONFIRMED	If AddressStatus is confirmed and PayerStatus is verified, the transaction may be eligible for PayPal Seller Protection. To learn more about PayPal Seller Protection, please contact PayPal directly or visit https://cms.paypal.com/uk/cgi-bin/?cmd=_render-content&content_ID=ua/Legal_Hub_full&locale.x=en_GB and view the relevant user agreements for further information.
	PayerStatus	Yes	Aa 0-9	20 Chars	VERIFIED UNVERIFIED	
	DeliverySurname	Yes	Aa á / \ & - '	20 Chars		N.B: All mandatory fields must contain a value, apart from the postcode. The postcode 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.
	DeliveryFirstnames	Yes	Aa á / \ & - '	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 3166-1 country code of the cardholder's delivery address	
	DeliveryState	No	US	2 Chars	State code for US customers only*	
	DeliveryPhone	No	0-9 - Aa + ()	20 chars		
	CustomerEMail	Yes	RFC532N	255 Chars		The customer's email address registered at PayPal
	PayerID	Yes	Aa 0-9	15 Chars		Unique PayPal User Reference ID

A8: Complete PayPal transaction

If you wish to complete a PayPal transaction (whose details are returned in A7 above), 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.

The response to this POST is the normal Direct transaction response detailed in A2.

Request format

	Name	Mandatory	Format	Max Length	Allowed Values	Description
	VPSProtocol	Yes	0-9 .	4 Chars	3. 00	This will match the protocol version supplied in A1.
	TxType	Yes	Aa	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 10,000.00	<p>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.</p> <p>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).</p> <p>NB : 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.</p>
	Accept	Yes	Aa	3 Chars	YES NO	<p>YES - if you wish to proceed with the PayPal transaction</p> <p>NO - if you wish to cancel based on the information returned.</p>

A9: Direct Full URL Summary

The table below shows the complete web addresses to which you send the messages detailed above.

Transaction Registration (PAYMENT, DEFERRED, AUTHENTICATE)	
TEST System:	https://test.sagepay.com/gateway/service/vspdirect-register.vsp
Live System:	https://live.sagepay.com/gateway/service/vspdirect-register.vsp

3D-Secure Callback	
TEST System:	https://test.sagepay.com/gateway/service/direct3dcallback.vsp
Live System:	https://live.sagepay.com/gateway/service/direct3dcallback.vsp

PayPal Completion	
TEST System:	https://test.sagepay.com/gateway/service/complete.vsp
Live System:	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).

A10: Sample XML

Surcharge XML - Example 1

IMPORTANT NOTE: The line breaks in the below example are included for readability only. No line breaks are needed.

```
<surcharges>
  <surcharge>
    <paymentType>VISA</paymentType>
    <percentage>4.25</percentage>
```

```

    </surcharge>
  <surcharge>
    <paymentType>MC</paymentType>
    <fixed>2.50</fixed>
  </surcharge>
</surcharges>

```

The above surchargeXML specifies that for this request all Visa payments will include a 4.25% surcharge and MasterCard payment will include a fixed surcharge of 2.50 in the currency being used.

Basket XML - Example 2

IMPORTANT NOTE: The line breaks in the below examples are included for readability only. No line breaks are needed.

An example of a basketXML with just products and no Trips data :

```

<basket>
  <agentId>johnsmith</agentId> -
  <item>
    <description>DVD 1</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
    <quantity>2</quantity>
    <unitNetAmount>24.50</unitNetAmount>
    <unitTaxAmount>00.50</unitTaxAmount>
    <unitGrossAmount>25.00</unitGrossAmount>
    <totalGrossAmount>50.00</totalGrossAmount>
    <recipientFName>firstname</recipientFName>
    <recipientLName>lastname</recipientLName>
    <recipientMName>M</recipientMName>
    <recipientSal>MR</recipientSal>
    <recipientEmail>firstname.lastname@test.com</recipientEmail>
    <recipientPhone>1234567890</recipientPhone>
    <recipientAdd1>add1</recipientAdd1>
    <recipientAdd2>add2</recipientAdd2>
  </item>
</basket>

```

```
<recipientCity>city</recipientCity>
<recipientState>CA</recipientState>
<recipientCountry>GB</recipientCountry>
<recipientPostCode>ha412t</recipientPostCode>
<itemShipNo>1123</itemShipNo>
<itemGiftMsg>Happy Birthday</itemGiftMsg>
</item>
<item>
  <description>DVD 2</description>
  <productSku>TIMESKU</productSku>
  <productCode>1234567</productCode>
  <quantity>1</quantity>
  <unitNetAmount>24.99</unitNetAmount>
  <unitTaxAmount>00.99</unitTaxAmount>
  <unitGrossAmount>25.98</unitGrossAmount>
  <totalGrossAmount>25.98</totalGrossAmount>
  <recipientFName>firstname</recipientFName>
  <recipientLName>lastname</recipientLName>
  <recipientMName>M</recipientMName>
  <recipientSal>MR</recipientSal>
  <recipientEmail>firstname.lastname@test.com</recipientEmail>
  <recipientPhone>1234567890</recipientPhone>
  <recipientAdd1>add1</recipientAdd1>
  <recipientAdd2>add2</recipientAdd2>
  <recipientCity>city</recipientCity>
  <recipientState>CA</recipientState>
  <recipientCountry>GB</recipientCountry>
  <recipientPostCode>ha412t</recipientPostCode>
  <itemShipNo>1123</itemShipNo>
  <itemGiftMsg>Congrats</itemGiftMsg>
</item>
<deliveryNetAmount>4.02</deliveryNetAmount>
<deliveryTaxAmount>20.00</deliveryTaxAmount>
<deliveryGrossAmount>24.02</deliveryGrossAmount>
<discounts>
  <discount>
    <fixed>5</fixed>
```

```
        <description>Save 5 pounds </description>
    </discount>
    <discount>
        <fixed>1</fixed>
        <description>Spend 5 pounds and save 1 pound</description>
    </discount>
</discounts>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
</basket>
```

An example of a basketXML field for a tour operator:

```
<basket>
  <agentId>johnsmith</agentId>
  <item>
    <description>Tour</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
    <quantity>1</quantity>
    <unitNetAmount>90.00</unitNetAmount>
    <unitTaxAmount>5.00</unitTaxAmount>
    <unitGrossAmount>95.00</unitGrossAmount>
    <totalGrossAmount>95.00</totalGrossAmount>
    <recipientFName>firstname</recipientFName>
    <recipientLName>lastname</recipientLName>
    <recipientMName>M</recipientMName>
    <recipientSal>MR</recipientSal>
    <recipientEmail>firstname.lastname@test.com</recipientEmail>
    <recipientPhone>1234567890</recipientPhone>
    <recipientAdd1>add1</recipientAdd1>
    <recipientAdd2>add2</recipientAdd2>
    <recipientCity>city</recipientCity>
    <recipientState>CA</recipientState>
    <recipientCountry>GB</recipientCountry>
  </item>
</basket>
```

```
<recipientPostCode>ha412t</recipientPostCode>
<itemShipNo>1123</itemShipNo>
<itemGiftMsg>Happy Birthday</itemGiftMsg>
</item>
<deliveryNetAmount>5.00</deliveryNetAmount>
<deliveryTaxAmount>0.00</deliveryTaxAmount>
<deliveryGrossAmount>5.00</deliveryGrossAmount>
<discounts>
  <discount>
    <fixed>5</fixed>
    <description>Save 5 pounds </description>
  </discount>
  <discount>
    <fixed>1</fixed>
    <description>Spend 5 pounds and save 1 pound</description>
  </discount>
</discounts>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<tourOperator>
  <checkIn>2012-10-12</checkIn>
  <checkOut>2012-10-29</checkOut>
</tourOperator>
</basket>
```

An example of a basketXML field for Car Rental:

```
<basket>
  <agentId>johnsmith</agentId>
  <item>
    <description>Tour</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
```

```
<quantity>1</quantity>
<unitNetAmount>90.00</unitNetAmount>
<unitTaxAmount>5.00</unitTaxAmount>
<unitGrossAmount>95.00</unitGrossAmount>
<totalGrossAmount>95.00</totalGrossAmount>
<recipientFName>firstname</recipientFName>
<recipientLName>lastname</recipientLName>
<recipientMName>M</recipientMName>
<recipientSal>MR</recipientSal>
<recipientEmail>firstname.lastname@test.com</recipientEmail>
<recipientPhone>1234567890</recipientPhone>
<recipientAdd1>add1</recipientAdd1>
<recipientAdd2>add2</recipientAdd2>
<recipientCity>city</recipientCity>
<recipientState>CA</recipientState>
<recipientCountry>GB</recipientCountry>
<recipientPostCode>ha412t</recipientPostCode>
<itemShipNo>1123</itemShipNo>
<itemGiftMsg>Happy Birthday</itemGiftMsg>
</item>
<deliveryNetAmount>5.00</deliveryNetAmount>
<deliveryTaxAmount>0.00</deliveryTaxAmount>
<deliveryGrossAmount>5.00</deliveryGrossAmount>
<discounts>
  <discount>
    <fixed>5</fixed>
    <description>Save 5 pounds </description>
  </discount>
  <discount>
    <fixed>1</fixed>
    <description>Spend 5 pounds and save 1 pound</description>
  </discount>
</discounts>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<carRental>
```

```
        <checkIn>2012-10-12</checkIn>
        <checkOut>2012-10-29</checkOut>
    </carRental>
</basket>
```

An example of a basketXML field for a Hotel Reservation:

```
<basket>
  <agentId>johnsmith</agentId>
  <item>
    <description>Tour</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
    <quantity>1</quantity>
    <unitNetAmount>90.00</unitNetAmount>
    <unitTaxAmount>5.00</unitTaxAmount>
    <unitGrossAmount>95.00</unitGrossAmount>
    <totalGrossAmount>95.00</totalGrossAmount>
    <recipientFName>firstname</recipientFName>
    <recipientLName>lastname</recipientLName>
    <recipientMName>M</recipientMName>
    <recipientSal>MR</recipientSal>
    <recipientEmail>firstname.lastname@test.com</recipientEmail>
    <recipientPhone>1234567890</recipientPhone>
    <recipientAdd1>add1</recipientAdd1>
    <recipientAdd2>add2</recipientAdd2>
    <recipientCity>city</recipientCity>
    <recipientState>CA</recipientState>
    <recipientCountry>GB</recipientCountry>
    <recipientPostCode>ha412t</recipientPostCode>
    <itemShipNo>1123</itemShipNo>
    <itemGiftMsg>Happy Birthday</itemGiftMsg>
  </item>
  <deliveryNetAmount>5.00</deliveryNetAmount>
  <deliveryTaxAmount>0.00</deliveryTaxAmount>
  <deliveryGrossAmount>5.00</deliveryGrossAmount>
  <discounts>
```



```

<discount>
  <fixed>5</fixed>
  <description>Save 5 pounds </description>
</discount>
<discount>
  <fixed>1</fixed>
  <description>Spend 5 pounds and save 1 pound</description>
</discount>
</discounts>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<hotel>
  <checkIn>2012-10-12</checkIn>
  <checkOut>2012-10-13</checkOut>
  <numberInParty>1</numberInParty>
  <guestName>Mr Smith</guestName>
  <folioRefNumber>A1000</folioRefNumber>
  <confirmedReservation>Y</confirmedReservation>
  <dailyRoomRate>150.00</dailyRoomRate>
</hotel>
</basket>

```

An example of a basketXML field for a Cruise:

```

<basket>
  <agentId>johnsmith</agentId>
  <item>
    <description>Tour</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
    <quantity>1</quantity>
    <unitNetAmount>90.00</unitNetAmount>
    <unitTaxAmount>5.00</unitTaxAmount>
    <unitGrossAmount>95.00</unitGrossAmount>
    <totalGrossAmount>95.00</totalGrossAmount>
    <recipientFName>firstname</recipientFName>
  </item>
</basket>

```

```
<recipientLName>lastname</recipientLName>
<recipientMName>M</recipientMName>
<recipientSal>MR</recipientSal>
<recipientEmail>firstname.lastname@test.com</recipientEmail>
<recipientPhone>1234567890</recipientPhone>
<recipientAdd1>add1</recipientAdd1>
<recipientAdd2>add2</recipientAdd2>
<recipientCity>city</recipientCity>
<recipientState>CA</recipientState>
<recipientCountry>GB</recipientCountry>
<recipientPostCode>ha412t</recipientPostCode>
<itemShipNo>1123</itemShipNo>
<itemGiftMsg>Happy Birthday</itemGiftMsg>
</item>
<deliveryNetAmount>5.00</deliveryNetAmount>
<deliveryTaxAmount>0.00</deliveryTaxAmount>
<deliveryGrossAmount>5.00</deliveryGrossAmount>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<cruise>
  <checkIn>2012-10-12</checkIn>
  <checkOut>2012-10-29</checkOut>
</cruise>
</basket>
```

An example of a basketXML field for a Airline:

```
<basket>
  <agentId>johnsmith</agentId>
  <item>
    <description>Tour</description>
    <productSku>TIMESKU</productSku>
    <productCode>1234567</productCode>
    <quantity>1</quantity>
    <unitNetAmount>90.00</unitNetAmount>
  </item>
</basket>
```

```
<unitTaxAmount>5.00</unitTaxAmount>
<unitGrossAmount>95.00</unitGrossAmount>
<totalGrossAmount>95.00</totalGrossAmount>
<recipientFName>firstname</recipientFName>
<recipientLName>lastname</recipientLName>
<recipientMName>M</recipientMName>
<recipientSal>MR</recipientSal>
<recipientEmail>firstname.lastname@test.com</recipientEmail>
<recipientPhone>1234567890</recipientPhone>
<recipientAdd1>add1</recipientAdd1>
<recipientAdd2>add2</recipientAdd2>
<recipientCity>city</recipientCity>
<recipientState>CA</recipientState>
<recipientCountry>GB</recipientCountry>
<recipientPostCode>ha412t</recipientPostCode>
<itemShipNo>1123</itemShipNo>
<itemGiftMsg>Happy Birthday</itemGiftMsg>
</item>
<deliveryNetAmount>5.00</deliveryNetAmount>
<deliveryTaxAmount>0.00</deliveryTaxAmount>
<deliveryGrossAmount>5.00</deliveryGrossAmount>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<airline>
  <ticketNumber>12345678901</ticketNumber>
  <airlineCode>123</airlineCode>
  <agentCode>12345678</agentCode>
  <agentName>26characterslong</agentName>
  <restrictedTicket>0</restrictedTicket>
  <passengerName>29characterslong</passengerName>
  <originatingAirport>BLR</originatingAirport>
  <segment>
    <carrierCode>ABC</carrierCode>
    <class>A01</class>
    <stopOver>1</stopOver>
    <legDepartureDate>2012-03-20</legDepartureDate>
```

```

                <destination>LHR</destination>
                <fareBasis>FARE12</fareBasis>
            </segment>
            <customerCode>20characterslong</customerCode>
            <flightNumber>BA0118</flightNumber>
            <invoiceNumber>123123123123123</invoiceNumber>
        </airline>
    </basket>

```

An example of a basketXML field for a Diners :

```

<basket>
    <agentId>johnsmith</agentId>
    <item>
        <description>Tour</description>
        <productSku>TIMESKU</productSku>
        <productCode>1234567</productCode>
        <quantity>1</quantity>
        <unitNetAmount>90.00</unitNetAmount>
        <unitTaxAmount>5.00</unitTaxAmount>
        <unitGrossAmount>95.00</unitGrossAmount>
        <totalGrossAmount>95.00</totalGrossAmount>
        <recipientFName>firstname</recipientFName>
        <recipientLName>lastname</recipientLName>
        <recipientMName>M</recipientMName>
        <recipientSal>MR</recipientSal>
        <recipientEmail>firstname.lastname@test.com</recipientEmail>
        <recipientPhone>1234567890</recipientPhone>
        <recipientAdd1>add1</recipientAdd1>
        <recipientAdd2>add2</recipientAdd2>
        <recipientCity>city</recipientCity>
        <recipientState>CA</recipientState>
        <recipientCountry>GB</recipientCountry>
        <recipientPostCode>ha412t</recipientPostCode>
        <itemShipNo>1123</itemShipNo>
        <itemGiftMsg>Happy Birthday</itemGiftMsg>
    </item>

```

```
<deliveryNetAmount>5.00</deliveryNetAmount>
<deliveryTaxAmount>0.00</deliveryTaxAmount>
<deliveryGrossAmount>5.00</deliveryGrossAmount>
<shipId>SHIP00002</shipId>
<shippingMethod>N</shippingMethod>
<shippingFaxNo>1234567890</shippingFaxNo>
<dinerCustomerRef>123123123</dinerCustomerRef>
</basket>
```

Example of a Customer XML field:

Customer XML - Example 3

IMPORTANT NOTE: The line breaks in the below example are included for readability only. No line breaks are needed.

```
<customer>
  <customerMiddleInitial>W</customerMiddleInitial>
  <customerBirth>1983-01-01</customerBirth>
  <customerWorkPhone>020 1234567</customerWorkPhone>
  <customerMobilePhone>0799 1234567</customerMobilePhone>
  <previousCust>0</previousCust>
  <timeOnFile>10</timeOnFile>
  <customerId>CUST123</customerId>
</customer>
```