Windcave

PxFusion – Technical Specification

Version 5.0

Copyright

Proprietary Notice

Contents

Overview

PxFusion is a product that allows merchants to accept credit card details within a form on their own web page. The form posts sensitive data directly to Windcave. Windcave will process the transaction and cause the user's browser to return to the merchant website in a way that is totally transparent to the cardholder.

The process workflow can be broken down into three main steps:

- Obtaining a TransactionId (via a Web Service).
- Using a HTML form to POST sensitive data directly to Windcave using the TransactionId as the identifier.
- Obtaining the results of the transaction using the TransactionId (via a Web Service).

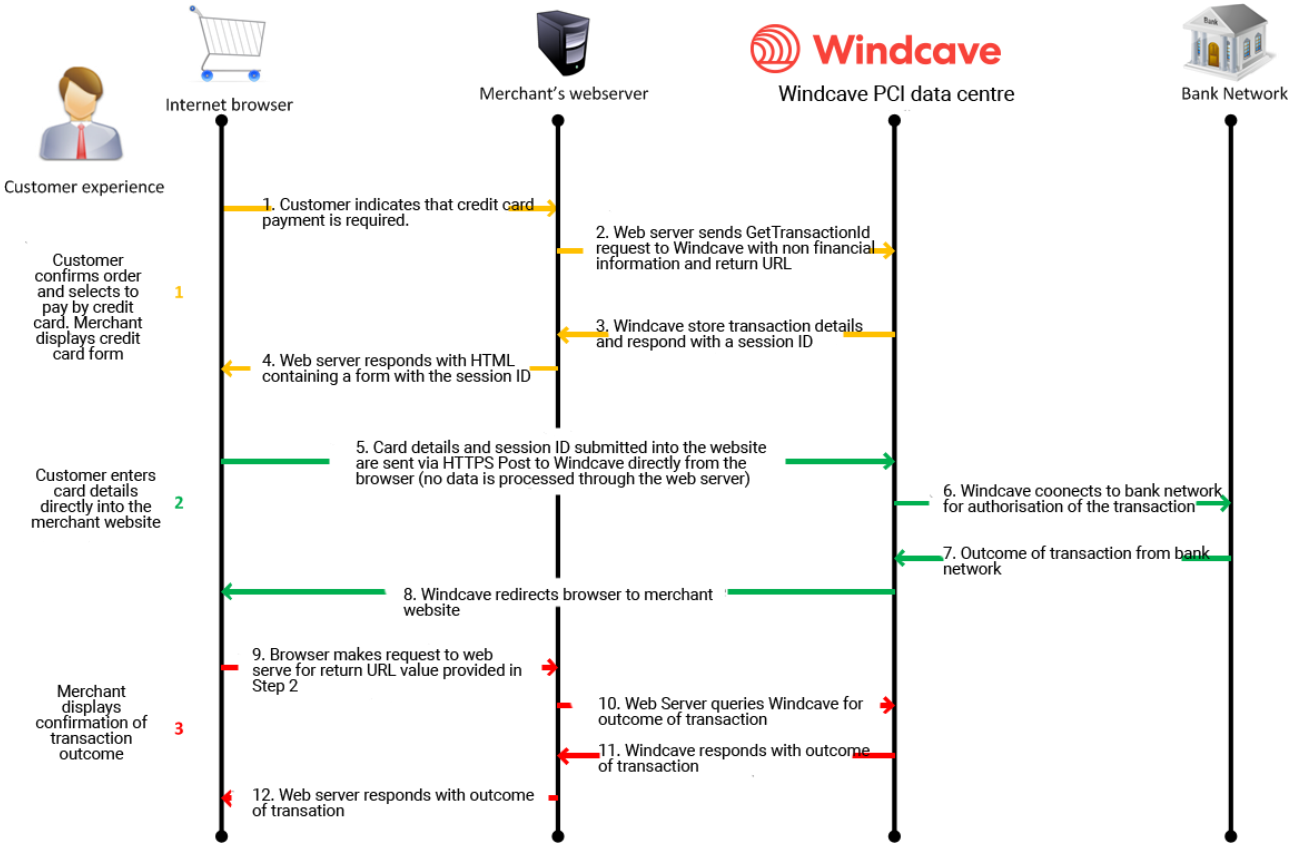
The following resources are exposed by Windcave® for testing and production;

A SOAP web service: https://sec.windcave.com/pxf/pxf.svc

WSDL is available here: https://sec.windcave.com/pxf/pxf.svc?wsdl

A page accepting client-side form POSTs: https://sec.windcave.com/pxmi3/pxfusionauth

When consuming the PxFusion web service please ensure that your web service stack supports lax versioning: that is, it does not throw exceptions for new unknown data members in received data. Also, ensure that your application does not perform schema validation against an expected schema before attempting to send or process messages.



GetTransactionId

The merchant's integrated application or server sends a server-side SOAP HTTP POST to the GetTransactionId web service call. The data submitted will not include sensitive card data such as card number, expiry date, CVC, card holder name and issue number. This data is not collected by, and is never available to, the merchant website.

Please review the GetTransactionId WSDL specifications. The sample call is provided below.

Please ensure the fields elements within the tranDetail element in the SOAP GetTransactionId have to be in alphabetical order as shown in the WSDL. Failure to ensure alphabetical ordering of fields can cause some fields to not be registered.

Input

The input properties for this GetTransactionId call are described below;

Property	Required	Description
amount	Yes	Amount of transaction in 1.23 format.
currency	Yes	Specify the currency here.
password	Yes	Password issued by Windcave
returnUrl	Yes	The URL (with protocol) to which the user's client side or browser process should be directed to once a transaction has been attempted. Max length 255 char. Note this URL (if externally accessible) will also receive the server side FPRN.
txnRef	Yes	Set by client to uniquely identify transaction. Used in order to process voids.
txnType	Yes	Purchase, Auth.
username	Yes	Username issued by Windcave.
enableAddBillCard	Optional	Needed for recurring billing transactions when adding a card to the Windcave system.
avsAction	No	Address Verification System property. Valid values are 0, 1, 2 & 3.
avsPostCode	No	Address Verification System property. Post Code that is listed on the customer's bank statement.
avsStreetAddress	No	Address Verification System property. Address that is listed on the customer's bank statement.
billingId	No	Specified for token billing transactions.

dateStart	No	The Issue date of the customer's credit card, if Issuer requires this field to be present.
enableAvsData	No	Address Verification System property. Values are 1 (Enable Verification), 0 (Disable Verification).
enablePaxInfo	No	Used for Airline Reservation Systems. Enable collection of extended booking data to go through to the acquirer if they support it.
merchantReference	No	64 character max free text field.
paxCarrier	No	Used for Airline Reservation Systems. 2 character airline identifier.
paxCarrier2	No	Used for Airline Reservation Systems. 2 character airline identifier.
paxCarrier3	No	Used for Airline Reservation Systems. 2 character airline identifier.
paxCarrier4	No	Used for Airline Reservation Systems. 2 character airline identifier.
paxClass1	No	Used for Airline Reservation Systems. Leg 1 flight class information.
paxClass2	No	Used for Airline Reservation Systems. Leg 2 flight class information.
paxClass3	No	Used for Airline Reservation Systems. Leg 3 flight class information.
paxClass4	No	Used for Airline Reservation Systems. Leg 4 flight class information.
paxDate2	No	Used for Airline Reservation Systems. Date departing in DD/MM/YY format.
paxDate3	No	Used for Airline Reservation Systems. Date departing in DD/MM/YY format.
paxDate4	No	Used for Airline Reservation Systems. Date departing in DD/MM/YY format.
paxDateDepart	No	Used for Airline Reservation Systems. Date departing in DD/MM/YY format.
paxFareBasis1	No	Used for Airline Reservation Systems.
paxFareBasis2	No	Used for Airline Reservation Systems.

paxFareBasis3	No	Used for Airline Reservation Systems.
paxFareBasis4	No	Used for Airline Reservation Systems.
paxFlightNumber1	No	Used for Airline Reservation Systems.
paxFlightNumber2	No	Used for Airline Reservation Systems.
paxFlightNumber3	No	Used for Airline Reservation Systems.
paxFlightNumber4	No	Used for Airline Reservation Systems.
paxLeg1	No	Used for Airline Reservation Systems. Leg 1 flight information.
paxLeg2	No	Used for Airline Reservation Systems. Leg 2 flight information.
paxLeg3	No	Used for Airline Reservation Systems. Leg 3 flight information.
paxLeg4	No	Used for Airline Reservation Systems. Leg 4 flight information.
paxName	No	Used for Airline Reservation Systems. Passenger Name.
paxOrigin	No	Used for Airline Reservation Systems. Passenger Origin.
paxStopOverCode1	No	Used for Airline Reservation Systems.
paxStopOverCode2	No	Used for Airline Reservation Systems.
paxStopOverCode3	No	Used for Airline Reservation Systems.
paxStopOverCode4	No	Used for Airline Reservation Systems.
paxTicketNumber	No	Used for Airline Reservation Systems. Passenger Ticket Number. Format: AAATTTTTTTTTTC.
paxTime1	No	Used for Airline Reservation Systems. Time departing.
paxTime2	No	Used for Airline Reservation Systems. Time departing.
paxTime3	No	Used for Airline Reservation Systems. Time departing.
paxTime4	No	Used for Airline Reservation Systems. Time departing.
paxTravelAgentInfo	No	Used for Airline Reservation Systems. Travel Agent description field.
timeout	No	A timestamp indicating a time after which, if attempted, the transaction will not be processed

		(http://www.w3.org/TR/xmlschema-2/#dateTime). Deprecated in favour of CancelTransaction.
transactionDetailsFields	No	The XML element to define the array of transaction details fields to enhance the usage of sessions and transactions depending on use cases.
transactionDetailsField	No	The XML element to define the array item that specifies the fieldName and fieldValue.
fieldName	No	Within the element transactionDetailsField, specifies the XML element to specify field's name. Possible string options: DebtRepaymentIndicator - Only send this field and set it to 1 to indicate that a debt repayment transaction is to be processed. Otherwise it will be set to 0 by default. EmailAddress – field to indicate setting the card holder's email address. InstallmentCount - Number value is used to indicate the total number of payments for an installment transaction. For example, if the consumer is making 12 installment payments for total payment, then this value should be set to 12. Only used for installment based payments. InstallmentNumber - Number value that is used to indicate the current payment number for an installment transaction. For example, if the consumer is making payment 1 of 12, then this value should be set to 1. Only used for installment based payments. RecurringMode – specifies the card storage reason on tokenisation (storing a card) and rebilling (rebilling card with a token) requests. Further details defined in the Token Billing section.
fieldValue	No	Within the element transactionDetailsField, the XML element to specify field's value after the fieldName XML element. Valid value options for fieldName: DebtRepaymentIndicator - set it to 1 to indicate that a debt repayment transaction is to be processed. Otherwise set to 0 or do not specify as it will be set to 0 by default. EmailAddress – set a valid email address of the cardholder to send the confirmation email about the payment transaction outcome.

		InstallmentCount – value to specify total the installment payments count. InstallmentNumber – value to specify current installment payment iteration number. RecurringMode – exact values defined in the Token Billing section.
txnData1	No	Optional Free Text. Max 255 char.
txnData2	No	Optional Free Text. Max 255 char.
txnData3	No	Optional Free Text. Max 255 char.

Basic Sample GetTransactionId web service call:

```
POST https://sec.windcave.com/PxF/pxf.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://paymentexpress.com/IPxFusion/GetTransactionId"
Host: sec.windcave.com
Content-Length: 2288
Expect: 100-continue
Connection: Keep-Alive
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransactionId xmlns="http://paymentexpress.com">
   <username>Username here</username>
   <password>Password here</password>
   <tranDetail xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <amount>1.00</amount>
    <currency>AUD</currency>
    <enableAddBillCard>false</enableAddBillCard>
    <merchantReference>Unique ID here</merchantReference>
    <returnUrl>https://www.mysite.com/returnpage.php</returnUrl>
    <txnRef>Unique ID here</txnRef>
    <txnType>Purchase</txnType>
   </tranDetail>
  </GetTransactionId>
 </s:Body>
</s:Envelope>
```

Output

The response to the GetTransactionId call will include an indication of the success of the operation and an ID relating to the session created. If the success element value is false then the merchant application need not look to the other properties, the call should be re-tried and brought to the attention of site administrators if attention is needed.

The SessionId property is a random identifier (which is not feasibly guessable) to be used for presentation within the form and URL presented on the client-side. The TransactionId property has been deprecated. It is expected that the merchant application stores the SessionId within their system.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 530
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
Date: Wed, 09 Feb 2011 04:16:48 GMT

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransactionIdResponse xmlns="http://paymentexpress.com">
   <GetTransactionIdResult xmlns:a="http://schemas.datacontract.org/2004/07/"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <a:sessionId>0000060001207174fc003a0a60b12345</a:sessionId>
    <a:success>true</a:success>
   <a:transactionId>0000060001207174fc003a0a60b12345</a:transactionId>
   </GetTransactionIdResult>
  </GetTransactionIdResponse>
 </s:Body>
</s:Envelope>
```

CancelTransaction

The merchant will make a server-side SOAP HTTP POST to the web service. The call will prevent a transaction taking place for a given SessionId.

Input

The input properties for this CancelTransaction call are described below;

Property	Required	Description
username	Yes	Username issued by Windcave.
password	Yes	Password issued by Windcave.
transactionId	Yes	The sessionId to be cancelled.

```
POST https://sec.windcave.com/PxF/pxf.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://paymentexpress.com/IPxFusion/CancelTransaction"
Host: sec.windcave.com
Content-Length: 348
Expect: 100-continue
Connection: Keep-Alive

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <CancelTransaction xmlns="http://paymentexpress.com">
   <username>Username here</username>
   <password>Password here</password>
   <transactionId>0000060001207174fc003a0a60b12345</transactionId>
  </CancelTransaction>
 </s:Body>
</s:Envelope>
```

Output

A single property named CancelTransactionResult is returned in response to the CancelTransaction call.

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 239
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
Date: Wed, 09 Feb 2011 04:35:04 GMT

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <CancelTransactionResponse xmlns="http://paymentexpress.com">
   <CancelTransactionResult>true</CancelTransactionResult>
  </CancelTransactionResponse>
 </s:Body>
</s:Envelope>
```

Rendering the Form

The merchant's website or application has to dynamically render a form on a web page which will POST sensitive card data directly to Windcave® securely using HTTPS (SSL Certificate required). Currently the AJAX based POST is not permitted. The following input elements should be included in the form:

Property	Mandatory	Type	Max Length
CardNumber	Yes	Numeric	19
ExpiryMonth	Yes	Numeric	2
ExpiryYear	Yes	Numeric	2
SessionId	Yes	Alphanumeric	N/A
CardHolderName	No	Alphanumeric	64
Cvc2	Yes	Numeric	4
Cvc2Presence	No	Numeric	1
IssueNumber	No	Numeric	3
UserTxnData1	No	Alphanumeric	255
UserTxnData2	No	Alphanumeric	255
UserTxnData3	No	Alphanumeric	255

The card detail input values are set by the user as they complete the fields. The SessionId returned by the GetTransactionId call should be included as the value of a hidden input named 'SessionId' within the form. Windcave will ignore any extraneous variables included within the data posted by the form. UserTxnData1 – 3 fields will be stored by Windcave only where no values were passed to Windcave within the TxnData1 – 3 fields within the GetTransactionId request.

Minimum required:

```
<form method="post" enctype="multipart/form-data"
action="https://sec.windcave.com/pxmi3/pxfusionauth">
<input type="text" name="CardNumber" />
<input type="text" name="ExpiryMonth" />
<input type="text" name="ExpiryYear" />
<input type="text" name="CardHolderName" />
<input type="text" name="Cvc2" />
<input type="hidden" name="SessionId" value="0000060001207174fc003a0a60b12345" />
<input type="submit" value="Submit" />
<input type="hidden" name="Action" value="Add" />
<input type="hidden" name="Object" value="DpsPxPay" />
<img src="https://www.windcave.com/Image/Windcave-Red-Logo-Horizontal.png" alt="Payment
Processor" width="236" height="42" />
</form>
```

```
POST https://sec.windcave.com/pxmi3/pxfusionauth HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-
xpsdocument, application/xaml+xml, application/x-ms-xbap, application/vnd.ms-excel,
application/vnd.ms-powerpoint, application/msword, application/x-shockwave-flash, */*
Accept-Language: en-us
Content-Type: multipart/form-data; boundary=----------7db1acd0c3c
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR
2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; MS-RTC LM 8; .NET CLR
3.0.30729; .NET4.0C; .NET4.0E)
Accept-Encoding: gzip, deflate
Host: sec.windcave.com
Content-Length: 794
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: PXID=; PXID=

----------7db1acd0c3c
Content-Disposition: form-data; name="CardNumber"

4111111111111111111
----------7db1acd0c3c
Content-Disposition: form-data; name="ExpiryMonth"

12
----------7db1acd0c3c
Content-Disposition: form-data; name="ExpiryYear"

12
----------7db1acd0c3c
Content-Disposition: form-data; name="CardHolderName"

C. Holder
----------7db1acd0c3c
Content-Disposition: form-data; name="Cvc2"

999
----------7db1acd0c3c
Content-Disposition: form-data; name="SessionId"

0000060001207174fc003a0a60b12345
----------7db1acd0c3c
Content-Disposition: form-data; name="Action"

Add
----------7db1acd0c3c
Content-Disposition: form-data; name="Object"

DpsPxPay
----------7db1acd0c3c--
```

For example;

```
<form method="post" enctype="multipart/form-data"
action="https://sec.windcave.com/pxmi3/pxfusionauth">
<input type="text" name="CardNum" />
<input type="text" name="ExMnth" />
<input type="text" name="ExYr" />
<input type="text" name="NmeCard" />
<input type="text" name="Cvc2" />
<input type="text" name="IssNum" />
<input type="text" name="UserTxnData1" />
<input type="text" name="UserTxnData2" />
<input type="text" name="UserTxnData3" />
<input type="hidden" name="SessionId" value="0000060001207174fc003a0a60b12345" />
<input type="submit" value="Submit" />
<input type="hidden" name="Action" value="Add" />
<input type="hidden" name="Object" value="DpsPxPay" />
<img src="https://www.windcave.com/Image/Windcave-Red-Logo-Horizontal.png" alt="Payment
Processor" width="236" height="42" />
</form>
```

In practice a merchant can implement the form in any way they choose including using different input types and JavaScript validation and CSS as long as they continue to pass through the correct name/value pairs. For example the following is valid;

```
   <form method="post" enctype="multipart/form-data"
action="https://sec.windcave.com/pxmi3/pxfusionauth" autocomplete="off">
   <table>
      <tr>
         <td>
            Card Number
         </td>
         <td>
            <input type="text" name="CardNumber" maxlength="19" />
         </td>
      </tr>
      <tr>
         <td>
            Expiry Month/Year
         </td>
```

```
          <td>
        <select name="ExpiryMonth">
          <option value="01">Jan</option>
          <option value="02">Feb</option>
          <option value="03">Mar</option>
          <option value="04">Apr</option>
          <option value="05">May</option>
          <option value="06">Jun</option>
          <option value="07">Jul</option>
          <option value="08">Aug</option>
          <option value="09">Sep</option>
          <option value="10">Oct</option>
          <option value="11">Nov</option>
          <option value="12">Dec</option>
        </select>
        <select name="ExpiryYear">
          <option value="09">2009</option>
          <option value="10">2010</option>
          <option value="11">2011</option>
          <option value="12">2012</option>
          <option value="13">2013</option>
          <option value="14">2014</option>
          <option value="15">2015</option>
        </select>
      </td>
    </tr>
    <tr>
      <td>
        Card holder name
      </td>
      <td>
        <input type="text" name="CardHolderName" />
      </td>
    </tr>
    <tr>
      <td>
        Security code
      </td>
      <td>
        <input type="text" name="Cvc2" maxlength="4" />
      </td>
    </tr>
    <tr>
      <td>
        Issue number
      </td>
      <td>
        <input type="text" name="IssueNumber" maxlength="3" />
      </td>
    </tr>
```

```
      <tr>
        <td>
          Transaction Data
        </td>            <td>
          <input type="text" name="UserTxnData1" />
        </td>
      </tr>
      <tr>
        <td>
          Transaction Data
        </td>
        <td>
          <input type="text" name="UserTxnData2" />
        </td>
      </tr>
      <tr>
        <td>
          Transaction Data
        </td>
        <td>
          <input type="text" name="UserTxnData3" />
        </td>
      </tr>
      <tr>
        <td>
                  <img src="https://www.windcave.com/Image/Windcave-Red-Logo-Horizontal.png"
alt="Payment Processor" width="236" height="42" />
        </td>
        <td>
          <input type="submit" value="Submit" />
        </td>
      </tr>
    </table>
    <input type="hidden" name="SessionId" value="00008400000001847b2bdcef432a4054" />
          <input type="hidden" name="Action" value="Add" />
          <input type="hidden" name="Object" value="DpsPxPay" />
    </form>
```

Transaction Result

Upon receiving the HTTP POST from the form at https://sec.windcave.com/pxmi3/pxfusionauth Windcave will process the transaction. The posted data is validated and combined with the other information supplied previously using the sessionId supplied in the hidden field.

Windcave will process the transaction to the acquirer in real-time. When the transaction is processed, typically in under 2 seconds, Windcave will direct the user's browser to the returnUrl supplied in the GetTransactionId call. Note this is an in-browser GET request to the returnUrl.

Also below is the section explained that helps ensure you receive a **server side** FPRN (Fail Proof Result Notification) in case the browser or client side request fails to reach the returnUrl due to the browser process being ended before it redirect to the returnUrl.

In order to obtain the full URL with which to redirect the user and notify the merchant of the transaction having taken place Windcave take the returnUrl supplied with the initial SOAP web service call and appends a 'sessionid' query string value. For example if the returnUrl originally supplied was:

https://www.mysite.com/returnpage.php?ses=1234

And the sessionId for the transaction is 0000060001207174fc003a0a60b12345; The URL to which the user will be returned is:

https://www.mysite.com/returnpage.php?ses=1234&sessionid=0000060001207174fc003a0a60b12345

```
HTTP/1.1 302 Redirect
Content-Type: text/html; charset=UTF-8
Location: https://www.mysite.com/returnpage.php?sessionid=0000060001207174fc003a0a60b12345
Server: Microsoft-IIS/7.5
Date: Wed, 09 Feb 2011 04:41:27 GMT
Content-Length: 214

<head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found <a HREF="
https://www.mysite.com/returnpage.php?sessionid=0000060001207174fc003a0a60b12345">here</a
></body>
```

Fail Proof Result Notification

Fail Proof Result Notification (FPRN) is a server side service that provides additional assurance that the merchant website will receive notification regarding the outcome of transactions completed via the client side merchant hosted payment page.

FPRN helps cater for the possibility that a user may not successfully navigate to the return URL within the browser enabling the merchant web application to acknowledge the outcome of the transaction.

The user could close their browser or otherwise navigate away from the client side merchant hosted payment page once they have been informed of the transaction outcome.

The merchant's web server may also be temporarily unavailable as the transaction is completed and therefore unable to recognise that a transaction has taken place.

Using the FPRN service the merchant website is virtually guaranteed to receive notification of the each and every transaction.

FPRN Process

FPRN is highly recommended by Windcave and is enabled on all new accounts by default.
The service ensures that the following processes occur for every transaction performed via the client side merchant hosted payment page.

1. Once the transaction outcome is determined for the session, a background process at Windcave makes an HTTP GET request to the merchant nominated URL set on the returnUrl field.
   Note: The URLs should resolve to a HTTP 200, 302 or 303 response externally to receive our server side FPRN.

2. If the merchant nominated URL is unreachable or returns any HTTP status code other than 200 or 404 the notification (HTTP GET) is retried up to a maximum of six times. For example, a 500 HTTP status code, indicating a temporary problem at the client site, will cause a retry.

3. Our FPRN service will stop sending any notifications immediately on receiving a 404 (Not Found) or 502 (Bad Gateway) HTTP status code.

We can configure the FPRN HTTP status code behaviour further per API user if required.

FPRN Guidelines

In order to ensure that the web application is in the best position to acknowledge the outcome of each and every transaction certain guidelines should be followed.

The merchant web application should not:

- Filter or base any conditional logic upon the originating IP address as this can vary.
- Depend upon receiving one and only one HTTP GET request for the returnUrl from the Windcave FPRN system (multiple requests may be sent).

The merchant web application should:

- Rely on the returnUrl field to receive the notification GET message.
  Note: the returnUrl will receive the server side notification (regardless of transaction outcome) separately and the same GET request redirect coming from a client side such as a web browser. The user agent for the server side notification is usually always one value fixed to our cloud but the client side redirect has a client side or web browser specific user agent.

- Your application should track and capture which ever GET request occurs first for any one sessionid. Any sessionid already processed should be disregarded to avoid any duplicate order payment fulfilment or creation.

- Get the 'sessionid' query string parameter name and value appended on the GET notification request to returnUrl.

- Use the GetTransaction request below to get the final decrypted transaction outcome details.

- Determine if a database operation or some form of communication such as generating an order record or sending an email is required.
  Generally this will mean that the application needs to be aware if these actions have been taken previously for the particular transaction or not. TxnRef should be used for this purpose.

Note: The URL at which the merchant website will process FPRN requests must be exposed via standard internet ports i.e. port 80 or port 443 for SSL/TLS traffic.

When specifying returnUrl field value please do not specify a non-standard port number within the URL.

Preconfigure Default FPRN and Custom Query Parameter

Also per specific API user account we can configure:

- The default preconfigured static URL to receive FPRN GET requests and/or,
- additional query string parameter key value pair for the FPRN.

This may help lock a static URL for the FPRN and custom query parameter for server side notifications. Please contact support@windcave.com to get assistance with this and specify the API username of concern. Otherwise we recommend to simply set your own externally accessible URL in the returnUrl field with your own custom query parameter in the returnUrl field for each GetTransactionId request.

GetTransaction

Upon receiving a request for the returnUrl the merchant is in a position to be able to update their records in recognition of the transaction and/or present the result to the user.

To do so the merchant must make a GetTransaction SOAP call. The merchant populates the transactionId parameter of the GetTransaction SOAP call with sessionid value contained within the query string.

Input

The following properties are required in the GetTransaction call input;

Property	Required	Description
username	Yes	Username issued by Windcave.
password	Yes	Password issued by Windcave.
transactionId	Yes	sessionId value supplied in the query string.

Example input request:

```
POST https://sec.windcave.com/PxF/pxf.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://paymentexpress.com/IPxFusion/GetTransaction"
Host: sec.windcave.com
Content-Length: 342
Expect: 100-continue

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransaction xmlns="http://paymentexpress.com">
   <username>Username here</username>
   <password>Password here</password>
   <transactionId>0000060001207174fc003a0a60b12345</transactionId>
  </GetTransaction>
 </s:Body>
</s:Envelope>
```

Output

The GetTransaction call response properties are as follows:

Property	Description
amount	Amount of transaction in 1.23 format.
authCode	Authorisation Code (up to 64 characters alphanumeric).
status	Status of transaction. See potential values below.
sessionId	Identifier returned by GetTransactionId call and submitted in GetTransaction request.
transactionId	The transactionId that is being queried.
billingId	Specified for token billing transactions.
cardHolderName	Card Holder Name as on Card.
cardName	Card used (Visa, MasterCard, Bankcard etc).
cardNumber	Truncated card number.
cardNumber2	A token which can be generated by Windcave when adding a card for recurring billing.
currencyId	Numeric currency identifier.
currencyName	Three character currency code.
currencyRate	Currency rate.
cvc2ResultCode	Contains information regarding verification of cvc2.
dateExpiry	Expiry date of card in 4 digit MMYY format. Note: do not include "/" or other delimiters.
dateSettlement	Date transaction will be settled to Bank Account in YYYYMMDD format. This is supported for most, but not all banks and card acquirers. If the DateSettlement is not available from the banking network, the DateSettlement will contain the current calendar date.
dpsBillingId	The Billing Id generated by Windcave when adding a card for recurring billing. Needed for rebilling transactions when you do not use your own BillingId.
dpsTxnRef	Unique identifier for the transaction. Required for programmatic completions and refunds.
merchantReference	64 character free text field.
reco	2 character response code.

responseText	Response Text associated with ReCo.
testMode	Mode indicator.
txnData1, 2 and 3	Value supplied in either form post or in GetTransactionId call.
txnType	Purchase, Auth.

Example output response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 1014
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
Date: Wed, 09 Feb 2011 04:41:53 GMT

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransactionResponse xmlns="http://paymentexpress.com">
   <GetTransactionResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <amount>1.00</amount>
    <authCode/>
    <billingId/>
    <cardHolderName>C. HOLDER</cardHolderName>
    <cardName>Visa</cardName>
    <cardNumber>411111......1111</cardNumber>
    <cardNumber2/>
    <currencyId>36</currencyId>
    <currencyName>AUD</currencyName>
    <currencyRate i:nil="true"/>
    <dateExpiry>1212</dateExpiry>
    <cvc2ResultCode>M</cvc2ResultCode>
    <dateSettlement>1980-01-01T00:00:00</dateSettlement>
    <dpsBillingId/>
    <dpsTxnRef>00000006067172bc</dpsTxnRef>
    <merchantReference>Unique ID here</merchantReference>
    <responseCode>BI</responseCode>
    <responseText>DECLINED</responseText>
    <sessionId>0000060001207174fc003a0a60b12345</sessionId>
    <status>1</status>
    <testMode>false</testMode>
    <transactionId>0000060001207174fc003a0a60b12345</transactionId>
    <txnData1/>
    <txnData2/>
    <txnData3/>
    <txnRef/>
    <txnType>Purchase</txnType>
   </GetTransactionResult>
  </GetTransactionResponse>
 </s:Body>
</s:Envelope>
```

Auth-Complete

Overview

Windcave supports Authorisation and Complete transaction types. An Auth transaction verifies that funds are available for the requested card and amount, and reserves the specified amount. A Complete transaction is sent at a later date to cause funds transfer for the previously authorised amount, or a smaller amount if the total original value is no longer required. This transaction set is useful when the merchant needs to ensure that funds up to a certain limit are available but the actual total amount is not yet known or goods or services have not yet been delivered.

Authorization

Use the GetTransactionId operation with TxnType set to "Auth" for the amount to be authorised. The Auth response contains a DpsTxnRef. The funds are not transferred from the cardholder account.

Complete

After a successful Auth transaction, but within 7 days maximum, a Complete transaction must be sent containing the DpsTxnRef returned by the Auth transaction. This can be done using either the PxPost, Web Service API or Payline/Payment manager (manual back office option).

Token Billing

Overview

Token Billing allows for regular billing of a cardholder card, under the control of the merchant, without requiring the merchant to either store sensitive card data securely or to obtain credit card details every time a new payment is requested. This functionality is implemented by providing the ability for a merchant to request Windcave to capture and store credit card number and expiry date and to link these stored details to a merchant supplied "BillingId".

The DpsBillingId is the default token type generated by Windcave not the merchant. A DpsBillingId will be generated for every transaction where the credit card information is to be stored (indicated by EnableAddBillCard field flag). The returned value will be 16 characters in length and is unique. The merchant can choose to use the DpsBillingId or their own BillingId.

The BillingId is a 32 character field that contains a reference that is unique to the merchant's customer that will be associated with the credit card information stored securely at Windcave. This is undertaken during the Setup Phase. For subsequent charges to the card (Rebill Phase), the merchant does not need to supply the card number or expiry date, only the BillingId originally associated during the Setup Phase.

CardNumber2 is a token generated by Windcave and associated with card details supplied. It is 16 numeric characters and conforms to a Luhn "mod 10" algorithm. This makes it ideal for storage within the database in place of a card number where the value is validated against checks which might normally be made against credit card numbers. A CardNumber2 value is always unique for a given card number. Should a card number be presented for tokenization multiple times the same CardNumber2 value will be returned.

CardNumber2 tokens are generated for all transactions once enabled by Windcave (please contact your Windcave account manager to discuss). The token number will be returned in the cardNumber2 property of the GetTransaction result.

Charging a CardNumber2 token involves a request from the merchant application or Batch processor including an appropriate cardNumber2, a TxnType (Purchase) and the amount to be charged (an optional MerchantReference can be added for reporting purposes). EnableAddBillCard value will need to be set to "False" (or 0) for the rebill phase. Windcave retrieves the credit card number and expiry date stored in the Setup Phase and a purchase transaction is formatted and processed to the card acquirer.

CardNumber2 transactions use the card expiry date stored with the token regardless of whether one is passed through in the transaction data. Once a successful transaction is processed using the real card number associated with a CardNumber2 token the expiry date stored with this token will be updated to that which was used to process the transaction. If your client application displays details of stored tokens to cardholders eg: masked number and expiry date, it is advisable upon a successful transaction for the merchant application to update the expiry date that is stored with the generated token.

Setup Phase

The setup phase consists of loading a card into Windcave with a transaction. The transaction can be a Purchase or Validate or Auth transaction type. The transaction can be an online amount 0.00 or 1.00 Validate which will only process a non-financial (zero dollar or no hold) transaction that is used check that the card and cardholder's account is valid. If the processing bank or acquirer does not support Validate then the Validate transaction request will be converted to an Auth transaction automatically.

Alternatively a $1.00 Auth transaction type request will determine that the card is valid and not on hot or stolen card lists but depending on the processing bank or acquirer the transaction may incur a temporary financial hold of the transaction amount.

The Purchase transaction type is used if the card is to be charged with an amount and tokenised at the same time.

To add a card for future rebilling, please include the following XML elements in the GetTransactionId web service call:

- enableAddBillCard - set to 1 when adding a card
- RecurringMode – required to be set within the transactionDetailsField as shown in the example request below
- billingId (optional – recommended to parse the dpsBillingId returned in the response instead)

In the RecurringMode request field, please set one of the card storage reason as the string listed below. When tokenising the card, please set one of the following:

RecurringMode	Usage explanation
credentialonfileinitial	Cardholder will save card and for future orders the cardholder selects to reuse the saved card for the one-off payment.
unscheduledcredentialonfileinitial	Cardholder will save their card and for future order based on an event (such as topup) the merchant will reuse the saved card on behalf of the cardholder for the one-off payment.

recurringinitial	Cardholder will save their card and merchant will reuse the saved card on behalf of cardholder for the subscribed recurring payments.
installmentinitial	Cardholder will save their card and merchant will reuse the saved card on behalf of cardholder for the installment payments.

Please discuss with our Implementation and Sales team about your tokenisation use cases if you are unsure. The RecurringMode string value should be set based on the merchant's business case for tokenising.

You can supply your own billing ID in the BillingId field or leave it blank and use the DpsBillingId returned in the response. We also recommend sending the unique txnRef and a merchantReference. This will make searching and identifying tokens/cards much easier.

Setup Token sample GetTransactionId web service call with RecurringMode in extra transaction details fields

```
POST https://sec.windcave.com/PxF/pxf.svc HTTP/1.1
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://paymentexpress.com/IPxFusion/GetTransactionId"
Host: sec.windcave.com
Content-Length: 2288
Expect: 100-continue
Connection: Keep-Alive

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransactionId xmlns="http://paymentexpress.com">
   <username>Username here</username>
   <password>Password here</password>
   <tranDetail xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <amount>1.00</amount>
    <currency>AUD</currency>
    <enableAddBillCard>true</enableAddBillCard>
    <merchantReference>Unique ID here</merchantReference>
    <returnUrl>https://www.mysite.com/returnpage.php</returnUrl>
    <transactionDetailsFields xmlns:ns="http://schemas.datacontract.org/2004/07/">
     <ns:transactionDetailsField>
      <ns:fieldName>RecurringMode</ns:fieldName>
      <ns:fieldValue>credentialonfileinitial</ns:fieldValue>
     </ns:transactionDetailsField>
    </transactionDetailsFields>
    <txnRef>Unique ID here</txnRef>
    <txnType>Purchase</txnType>
   </tranDetail>
  </GetTransactionId>
 </s:Body>
</s:Envelope>
```

Rebill Phase

The merchant application or Batch processor requests a new transaction and supplies the appropriate DpsBillingId or CardNumber2 or BillingId, RecurringMode (to specify the reason for rebilling with the token), MerchantReference (which appears on reports), and the amount to be charged using either PxPost or the Web Service.

EnableAddBillCard value will be set to "False" (or 0) for the rebill phase.

Windcave retrieves the credit card number and expiry date stored in the Setup Phase and a purchase transaction is formatted and processed to the card acquirer.

Recurring Transactions

If transactions are being processed to an acquirer that supports and is configured for recurring transactions, the Windcave account can also be setup to process recurring transactions only.

The main advantage of recurring transactions is that the ExpiryDate is not required. This further reduces the amount of data that needs to be stored for a merchant, and bypasses the issue of expired cards.

To setup your Windcave account for recurring transaction processing only, please contact support@windcave.com (note: please ensure that your merchant bank account has been setup for recurring transactions).

Extended Airline Booking Data

PxFusion is capable of taking extended booking information, which is used to display on cardholders statements.

If you would like to add booking information to your transaction details you will need to set the EnablePaxInfo input property to “True” (or 1) and you will be able to use the following properties -

PaxDateDepart, PaxName, PaxLeg1, PaxLeg2, PaxLeg3, PaxLeg4, PaxOrigin, PaxTicketNumber, PaxCarrier and PaxTravelAgentInfo.

Sample data:

PaxDateDepart = "14122008"

PaxName = "Brian Smith"

PaxOrigin = "AKL"

PaxLeg1 = "SYD"

PaxLeg2 = "LAX"

PaxLeg3 = "LHR"

PaxLeg4 = "AKL"

PaxTicketNumber = "0030458343"

PaxCarrier = "QB"

PaxTravelAgentInfo = "BA1234567890"

Test Cards

The following pre-approved 'test card' numbers can be used for testing, within test environments.

Visa - 4111111111111111

MasterCard - 5431111111111111

Amex - 371111111111114

Diners - 36000000000008

Note: These are only suitable for Windcave test accounts.

Visa Checkout

PxFusion supports Visa Checkout Payment as an alternative payment method to standard credit card. The following section documents the requirements and steps to accept payments using Visa Checkout.

Prerequisites

- Visa Checkout Sandbox account and credentials, consisting of an API key and shared secret, to communicate with Visa Checkout.
- Windcave PxFusion test account credentials.

1. Visa Checkout Button

   Merchant must code Visa Checkout Button on their webpage.

   A quick start tutorial can be found on Visa's website:
   https://developer.visa.com/capabilities/visa_checkout/docs-how-to

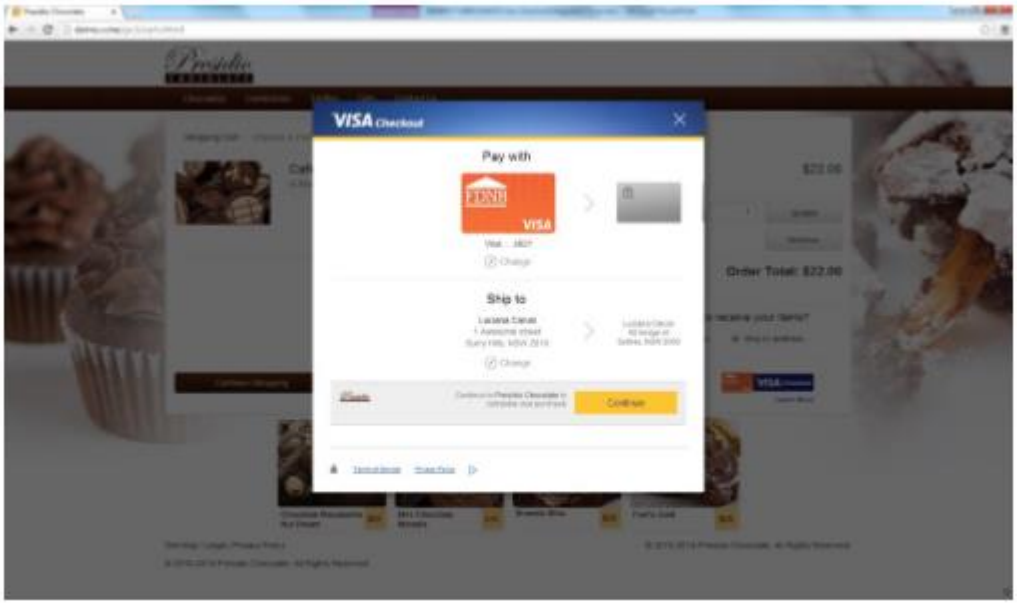
   When clicked, the Visa Checkout Button calls Visa Checkout directly and initiates a lightbox. The call contains the Merchant's API Key.

   Example:

```
function onVisaCheckoutReady(){
        V.init( {
                apikey: "PP83HJWDJZ9PZ8PFSDAC13C_3-1PARlGy0w5nivD8d1GY04bk",
                paymentRequest:{
                currencyCode: "NZD",
                total: "10.00",
                subtotal:"10.00",
                promoCode: "123456"
                },

                settings: {
         externalProfileId: "profileid",
          externalClientId: "clientid",
        locale: "en_NZ",
         countryCode: "NZ"
         }
```

```
        });
        V.on("payment.success", function(payment){
                console.log("Success " + JSON.stringify(payment));
        });

        V.on("payment.cancel", function(payment){
                console.log("cancel " + JSON.stringify(payment));
        });

        V.on("payment.error", function(payment, error){
                console.log("Error " + JSON.stringify(payment));
                  } ) ;
}
```

2. Process Payment
   Visa Checkout Customer completes the payment process within the lightbox.



3. Payload Returned
   Merchant is returned the payload which contains the Call ID.

4. Submit Transaction using PxFusion
   Merchant submits the Call ID to Windcave using PxFusion. This is done using GetTransactionId.

   Request:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <GetTransactionId xmlns="http://paymentexpress.com">
   <username>Sample_User</username>
   <password>**********************************</password>
   <tranDetail>
    <amount>1.00</amount>
```

```
      <avsAction />
      <avsPostCode />
      <avsStreetAddress />
      <billingId />
      <currency>NZD</currency>
      <dateStart />
      <merchantReference>PX Test</merchantReference>
      <returnUrl>https://www.mysite.com/return.html</returnUrl>
      <txnData1 />
      <txnData2 />
      <txnData3 />
      <txnRef />
      <txnType>Purchase</txnType>
      <callId>INSERT CALL ID</callId>
    </tranDetail>
  </GetTransactionId>
 </soap:Body>
</soap:Envelope>
```

Response:

```
GetTransactionId
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
 <s:Body>
  <GetTransactionIdResponse xmlns="http://paymentexpress.com">
   <GetTransactionIdResult xmlns:a="http://schemas.datacontract.org/2004/07/"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <a:sessionId>000001000079155000584ceba047da3d</a:sessionId>
    <a:success>true</a:success>
    <a:transactionId>000001000079155000584ceba047da3d</a:transactionId>
   </GetTransactionIdResult>
  </GetTransactionIdResponse>
 </s:Body>
</s:Envelope>
```

5. Post SessionID and CallId to Windcave

```
<form action="https://sec.windcave.com/pxmi3/pxfusionvisacheckout">
 SessionId: <input type="text" id="SessionId" name="SessionId" value="" size="60">
 <br>
 CallId: <input type="text" id="CallId" name="CallId" value="" size="60">
 <br><br>
 <input type="submit" value="Submit">
</form>
```

6. Windcave Call Visa Checkout

   Windcave will call Visa Checkout to retrieve consumer's payment data.

7. Process Transaction

   Windcave Processes transaction with the merchant's acquirer.

8. Transaction Result

   Windcave returns transaction result information to Merchant in PxFusion Response.

   Request:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetTransaction xmlns="http://paymentexpress.com">
      <username>Sample_User</username>
      <password>Sample_Password</password>
      <transactionId>00000100079155400f5fc9b111b8937</transactionId>
    </GetTransaction>
  </soap:Body>
</soap:Envelope>
```

Response:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <GetTransactionResponse xmlns="http://paymentexpress.com">
      <GetTransactionResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <amount>1.00</amount>
        <authCode>012529</authCode>
        <billingId />
        <cardHolderName>PX TEST</cardHolderName>
        <cardName>Visa</cardName>
        <cardNumber>411111.....11</cardNumber>
        <cardNumber2 />
        <currencyId>554</currencyId>
        <currencyName>NZD</currencyName>
        <currencyRate i:nil="true" />
        <cvc2ResultCode>U</cvc2ResultCode>
        <dateExpiry>0120</dateExpiry>
        <dateSettlement>2016-05-02T00:00:00</dateSettlement>
        <dpsBillingId />
        <dpsTxnRef>0000000100312345</dpsTxnRef>
        <merchantReference>Merchant ref</merchantReference>
        <responseCode>00</responseCode>
        <responseText>APPROVED</responseText>
        <riskRuleMatches xmlns:a="http://schemas.datacontract.org/2004/07/" />
        <sessionId>000001000079155400f5fc9b11112345</sessionId>
        <status>0</status>
        <testMode>false</testMode>
        <transactionId>00000100079155400f5fc9b111b8937</transactionId>
        <transactionResultFields xmlns:a="http://schemas.datacontract.org/2004/07/" />
        <txnData1>My TxnData 1</txnData1>
        <txnData2>My TxnData 2</txnData2>
        <txnData3>My TxnData 3</txnData3>
        <txnMac>CCC20210</txnMac>
        <txnRef />
        <txnType>Purchase</txnType>
        <callId>CALL ID</callId>
      </GetTransactionResult>
    </GetTransactionResponse>
  </s:Body>
</s:Envelope>
```

Properties Description

The following section provides a detailed description of each PxFusion element and indicates if it used as an input or output. If a property is marked as input, it is not included in the response to GetTransaction.

Amount (input/output) Datatype: String Max 12 characters

Format is d.cc where d is dollar amount (no currency indicator) and cc is cents amount. For example, $1.80 (one dollar and eighty cents) is represented as "1.80", not "1.8". A string value is used rather than the conventional Currency Datatype to allow for easy integration with Web applications. Maximum value allowable is $99,999.99. Note that acquirer or card limits may be lower than this amount. When submitting transactions for currencies with no decimal division of units such as JPY the Amount input must be in an appropriate format e.g. "10".

AuthCode (output) Datatype: String Max 22 characters
Authorization code returned for approved transactions.

AvsAction (input) 1 digit
Address Verification System property. Valid values are 0, 1, 2 & 3.

0 - do not check AVS details with acquirer, but pass them through to Windcave only.
1 - Attempt AVS check. If the acquirer doesn't support AVS or AVS is unavailable, then the transaction will proceed as normal. If AVS is supported and the AVS check fails, then the transaction will be declined.
2 - The same as 1 except the transaction must be checked by AVS. If AVS isn't available, the transaction will be declined.
3 - AVS check will be attempted and any outcome will be recorded, but ignored i.e. transaction will not be declined if AVS fails or unavailable.

AvsPostCode (input) Datatype: String Max 20 characters
Address Verification System property. Post Code that is listed on the customer's bank statement.

AvsStreetAddress (input) Datatype: String Max 60 characters
Address Verification System property. Address that is listed on the customer's bank statement.

BillingId (input/output) Datatype: String Max 32 characters
If a token based billing transaction is to be created, a BillingId may be supplied. This is an identifier supplied by the merchant application that is used to identify a customer or billing entry and can be used as input instead of card number and date expiry for subsequent billing transactions.

CardHolderName (input) Datatype: String Max 64 characters
The cardholder name as it appears on customer card; Optional and may be left blank.

CardName (output) Datatype: String Max 16 characters
The card type used for the transaction. Note that the list may be expanded as support for new cards is added. The CardName format is to capitalize the first letter with remaining letters in lowercase.

CardName Value	Description
Amex	American Express
Bankcard	Bank Card
Diners	Diners Card
Jcb	JCB
Mastercard	Mastercard

Visa	Visa

CardNumber2 (output) Datatype: String Max 19 characters
A token generated by Windcave when adding a card for recurring billing. CardNumber2 is a 16 digit number which conforms to a Luhn 'mod 10' algorithm and has a 1-to-1 relationship with the actual card number used. To use CardNumber2 tokens your account must be configured to generate them. Please contact Windcave support if you would like to use this value.

Currency (input/output) Datatype: String Max 4 characters
Indicates currency used for this transaction. If blank, currency will be determined by the bank account used which is selected using the Username/Password details. Not all acquirers can support multiple currencies. Valid values for Currency are:

CAD	Canadian Dollar
CHF	Swiss Franc
DKK	Danish Krone
EUR	Euro
FRF	French Franc
GBP	United Kingdom Pound
HKD	Hong Kong Dollar
JPY	Japanese Yen
NZD	New Zealand Dollar
SGD	Singapore Dollar
THB	Thai Baht
USD	United States Dollar
ZAR	Rand
AUD	Australian Dollar
WST	Samoan Tala
VUV	Vanuatu Vatu
TOP	Tongan Pa'anga
SBD	Solomon Islands Dollar
PGK	Papua New Guinea Kina
MYR	Malaysian Ringgit
KWD	Kuwaiti Dinar

FJD	Fiji Dollar

CurrencyId (output) Datatype: String Max 4 characters
Indicates the input currency's numeric ISO currency code.

CurrencyRate (Output) Datatype: String Max 3 characters
Charge rate, if applicable.

Cvc2 (Input) Datatype: String Max 4 characters
Card Verification Code 2 number. Some payment cards are issued with additional identifying information. These cards will have the account number printed on the signature panel of the card followed by a three or four digit value. This value is generated by the issuing bank and can be verified by the bank. Payment card brands have varying names for the value:

- American Express: Four-digit batch code (4DBC)
- MasterCard: Card Verification Code 2 (CVC2)
- Visa: Card Verification Value 2 (CVV2)

Supplying this value provides an indication of that the person participating in a transaction had physical possession of the card at some point in time. This is not currently implemented by all acquirer and may not necessarily be checked

Cvc2Presence (Input) String 1 characters
Merchant to send Windcave a presence indicator within "**Cvc2Presence**" field in the transaction request to one of the below:
0 - You (MERCHANT) have chosen not to submit CVC
1 - You (MERCHANT) have included CVC in the Auth / Purchase
2 - Card holder has stated CVC is illegible.
9 - Card holder has stated CVC is not on the card.

Cvc2ResultCode (Output) String 1 characters
The CVC result code indicates the following:

Response Code	Definition	Interpreting Response Codes
M	CVC matched.	You will want to proceed with transactions for which you have received an authorisation approval. A CVC match indicates the values provided matches the Issuing Banks details
N	CVC did not match.	You may want to follow up with the cardholder to verify the CVC value before completing the transaction, even if you have received an authorisation approval. The CVC details provided by the Cardholder do not match their Issuing Banks details
P	CVC request not processed.	Issuing Bank is unable to process CVC at this time
S	CVC should be on the card, but merchant has sent code indicating there was no CVC.	You may want to follow up with the cardholder to verify that the customer checked the correct location for the CVC. If the transaction is Approved you may also wish to consider not fulfilling the transaction
U	Issuer does not support CVC.	The card issuing bank does not support CVC process

DateExpiry (output) Datatype: String Max 4 characters
Indicates card expiry date. Format is MMYY where MM is month 01-12 and Year 00-99. Do not insert "/" or other delimiter. Not required for Complete or Refund transactions.

DateSettlement (output) Datatype: String Max 8 characters
Indicates Date of settlement (when money will be deposited in Merchant bank account) if this is supported by the Acquirer, otherwise contains the date the transaction was processed in YYYYMMDD format.

DateStart (input) Datatype: String Max 4 characters
The Issue date of the customer's credit card, if Issuer requires this field to be present.
Format is MMYY where MM is month 01-12 and Year 00-99. Do not insert "/" or other delimiter.
Used for Maestro/Solo cards.

EnableAddBillCard (input) Datatype: Boolean True/False
If set to "True" (or 1) the details necessary to charge the same customer in the future are securely stored. A BillingId may optionally be attached on input. A DpsBillingId is returned.

EnableAvsData (input) Datatype: Boolean True/False
Address Verification System property. Values are 1 (Enable Verification), 0 (Disable Verification). Your bank may require that you use AVS, in which case you will need to set to 1.

EnableMandatoryCVC2 (input) Datatype: Boolean True/False
When set to true, mandates a CVC2 value in the form post, the transaction will return a declined response if a CVC2 value is not submitted.

EnablePaxInfo (input) Data type: Boolean True/False
Used for Airline Reservation Systems. Enable collection of extended booking data to go through to the acquirer. Value will need to be true (1) if ticket information is included with the transaction.

DpsBillingId (input/output) Datatype: String Max 16 characters
Returned for a successful billing transaction if EnableAddBillCard is set. Supplied as input to rebill a transaction if BillingId is not used. It is not allowed to specify both a BillingId and a DpsBillingId when rebilling a transaction.

DpsTxnRef (output) Datatype: String Max 16 characters
Unique transaction identifier that is returned for every transaction. If the transaction was approved, DpsTxnRef can be used as an input to a Refund transaction. Used to specify a transaction for refund without supplying the original card number and expiry date (can only be done using PxPost or Web Service).

MerchantReference (input/output) Datatype: String Max 64 characters
Free Text Field for use by merchant (could be order number, customer number etc.).

Password (input) Data type: String Max 64 characters
Used with Username to determine account for settlement. Windcave clients can be set up with more than one bank account. Each transaction may be designated for a specific account if required.

PaxCarrier (input) Data type: String Max 2 characters
Used for Airline Reservation Systems. Carrier flight information. Alphanumeric.

PaxCarrier2 - 4 (input) Data type: String Max 2 characters
Used for Airline Reservation Systems. Carrier flight information. Alphanumeric.

PaxClass1 - 4 (input) Data type: String Max 1 characters
Used for Airline Reservation Systems. Class flight information. Alphanumeric.

PaxDate2 - 4 (input) Data type: String Max 20 characters
Used for Airline Reservation Systems. Leg depart date flight information. Alphanumeric.

PaxDateDepart (input) Data type: String Max 8 characters
Used for Airline Reservation Systems. Date departing in YYYYMMDD format. Numeric.

paxFareBasis1 - 4 (input) Data type: String Max 6 characters
Used for Airline Reservation Systems. Fare basis flight information. Alphanumeric.

paxFlightNumber1 - 4 (input) Data type: String Max 6 characters
Used for Airline Reservation Systems. Flight number information. Alphanumeric.

PaxLeg1 - 4 (input) Data type: String Max 3 characters
Used for Airline Reservation Systems. Flight number information. Alphanumeric.

PaxName (input) Data type: String Max 20 characters
Used for Airline Reservation Systems. Passenger Name. Alphanumeric.

PaxOrigin (input) Data type: String Max 3 characters
Used for Airline Reservation Systems. Passenger Origin of departure. Alphanumeric.

PaxStopoverCode1 - 4 (input) Data type: String Max 1 characters
Used for Airline Reservation Systems. Stop over code flight information. Alphanumeric.

PaxTicketNumber (input) Data type: String Max 10 characters
Used for Airline Reservation Systems. Passenger Ticket Number. Format: AAATTTTTTTTTTC. AAA is airline code, TTTTTTTTTT (10 chars) is actual ticket number and C is check digit. Numeric.

PaxTime1 - 4 (input) Data type: String Max 4 characters
Used for Airline Reservation Systems. Leg depart time flight information. Alphanumeric.

PaxTravelAgentInfo (input) Data type: String Max 25 characters
Used for Airline Reservation Systems. Travel Agent description field. Also known as the Booking Reference on some of Windcave screens. Alphanumeric free text field.

ResponseCode (output) Datatype: String Max 2 characters
Response Code generated by Windcave Server (for locally declined transactions) or by the Card Acquirer (for host originated responses). The ResponseCode should not be checked by the client application, as these values differ according to acquirer.

ResponseText (output) Datatype: String Max 32 characters
Response Text associated with the response code of the transaction

ReturnUrl (input) Datatype: String Max 255 characters
URL (with protocol) that the user will be forwarded to once the card details are submitted (with the session ID appended).

SessionId (output) Datatype: String Max 32 characters
Unique transaction identifier that is returned after a successful GetTransactionId call and used to obtain Transaction results using GetTransaction.

Status (output) Datatype: Integer Max 1 character
The status property contains an integer indicating the status of the GetTransactionId call and any transaction associated with it. Possible values for the status property are as follows.

Value	Detail
0	Transaction approved.
1	Transaction declined.
2	Transaction declined due to transient error (retry advised).
3	Invalid data submitted in form post (alert site admin).
4	Transaction result cannot be determined at this time (re-run GetTransaction).
5	Transaction did not proceed due to being attempted after timeout timestamp or having been cancelled by a CancelTransaction call.

6	No transaction found (SessionId query failed to return a transaction record – transaction not yet attempted).

TransactionId (output) Datatype: String Max 32 characters
Unique transaction identifier that is returned after a successful GetTransactionId call and used to obtain Transaction results using GetTransaction.

TransactionDetailsFields (input) Datatype: String
The XML element to define the array of transaction details fields to enhance the usage of sessions and transactions depending on use cases.

TransactionDetailsField (input) Datatype: String
The XML element to define the array item that specifies the fieldName and fieldValue XML elements.

FieldName (input) Datatype: String
Within the element transactionDetailsField, the XML element to specify field's name.

FieldValue (input) Datatype: String
Within the element transactionDetailsField, the XML element to specify field's value after the fieldName XML element.

TxnData1, TxnData2, TxnData3 (input/output) String Max 255 characters
Optional free text fields.

TxnRef (input/output) Datatype: String Max 16 characters
A unique identifier provided by your application to uniquely identify the transaction. This is the TxnRef supplied by the client to initiate the transaction.

TxnType (input/output) Datatype: String Max 8 character

Value	Meaning
Validate	Validates the card and account with no financial transaction or hold of funds. Only used to tokenise the card in Token Billing setup phase.
Auth	Authorise - amount is authorised and held temporarily, no funds transferred.
Purchase	Purchase - Funds are transferred immediately.

Username (input) Data type: String Max 32 characters
Used with Password to determine account for settlement. Windcave clients can be set up with more than one bank account. Each transaction may be designated for a specific account if required.