SOAP Integration Guide

Document Version 1.4
December 2010

For further information please contact Beanstream customer support at (250) 472-2326 or support@beanstream.com

Overview

The Simple Object Access Protocol (SOAP) is a lightweight XML-based protocol that can be used in combination with a variety of existing Internet protocols and formats including HTTP, SMTP, and MIME and can support a wide range of applications from messaging systems to RPC.

This document contains an overview of the SOAP protocol for processing transactions through the Beanstream Payment Gateway. This guide is intended for users that are already familiar with SOAP, Web Services and Web Services Description Language (WSDL).

Additional Reference Material

This guide is a supplement to the Beanstream Process Transaction API Guide. Consult the Process Transaction API Guide for a complete description of input/response variables and other essential information for connecting to the Beanstream gateway.

Recent SOAP API Updates

In the fall of 2010 Beanstream introduced an updated SOAP API to extend the full Beanstream payment gateways feature-set to SOAP developers. With the new SOAP API, the following system changes took effect.

- A new SOAP API service version 3.1 has been introduced
- All standard Beanstream gateway API input and response variables are now accessible via SOAP
- The MS Soap Toolkit is no longer required for integration
- New Web Services and WSDL file packages have been created

1 System Requirements

In order to leverage the SOAP interface, you must have a merchant account with Beanstream and be able to communicate with our web server via 40-bit or 128-bit SSL.

2 SOAP Message Format

Transaction requests are made with a single XML message passed to the TransactionProcess method. A second TransactionProcessAuth method is used by merchants processing INTERAC Online transactions or for those using the Verified by Visa service to validate the results of a credit card password authentication

Before submitting your transaction to the Beanstream Transaction Server, all XML special characters must be converted to their associated Entity Reference. The Beanstream Transaction Server uses a fully compliant XML parser to retrieve your transaction details. If XML special characters are detected, the transaction request will be declined.

An Entity Reference is a named or numeric code that allows the XML parser to interpret the special character. There are five named Entity References that the XML parser will handle, and are listed in the table below. Any additional Entity References you wish to use must be declared within your message.

Special Character	Entity Reference
&	&amp;
<	&lt;
>	&gt;
"	&amp;quote;
‘	&apos;

3 Web Services and WSDL Files

To integrate via SOAP, you will need to reference the following essential files.

https://www.beanstream.com/WebService/ProcessTransaction.asmx

https://www.beanstream.com/WebService/ProcessTransaction.asmx?WSDL

4 Input and Response Variables

The Beanstream SOAP API supports all of the standard Transaction Processing API variables for processing credit card transactions (with or without Verified by Visa or SecureCode) and INTERAC Online® transactions. In addition to these standard input variables, you must specify a SOAP service version.

- **serviceVersion=1.3**

The current SOAP service version is 1.3. If you fail to pass this variable, or specify another service version, the transaction will fail to process.

All standard response variables will be returned in XML format as per any standard Process Transaction API integration.

5 Converting to the New SOAP API (Existing Clients)

If you have an existing SOAP integration, you can easily convert to the new API. You will need to:

- Update you SOAP service version to 1.3 by specifying <serviceVersion>1.3</serviceVersion> in all requests.
- Refresh any references to the old web services and WSDL files. Update these to the current locations specified in section 3 of this guide.

To test your changes in a safe environment, contact support@beanstream.com to request a sandbox account. You will be issued a new merchant ID number and login information for the Beanstream member area. Your sandbox account will allow you to test your changes free of charge before migrating your new API integration to a live processing account.

6 Samples

6.1 Sample Standard Transaction Request and Response

Request Scenario

In the following scenario, a standard credit card transaction is submitted to the Beanstream gateway for customer Samp Shopper for a transaction value of $11.99. This request uses API username and passcode validation.

```
<transaction>
          <serviceVersion>1.3</serviceVersion>
          <merchant_id>123456789</merchant_id>

          <trnType>P</trnType>
          <trnOrderNumber> SO5446</trnOrderNumber>
          <trnAmount>19.99</trnAmount>

          <trnCardOwner>Sam Shopper</trnCardOwner>
          <trnCardNumber>4030000010001234</trnCardNumber>
          <trnExpMonth>05</trnExpMonth>
          <trnExpYear>12</trnExpYear>

          <ordEmailAddress>user@domain.com</ordEmailAddress>
          <ordName>Sam Shopper</ordName>
          <ordPhoneNumber>999-999-9999</ordPhoneNumber>
          <ordAddress1>123 Main Street</ordAddress1>
          <ordAddress2></ordAddress2>
          <ordCity>Toronto</ordCity>
          <ordProvince>ON</ordProvince>
          <ordPostalCode>M5W 1E6</ordPostalCode>
          <ordCountry>CA</ordCountry>

          <ref1>Customer information 1</ref1>
          <ref2> Customer information 2</ref2>
          <ref3> Customer information 3</ref3>
          <ref4> Customer information 4</ref4>
          <ref5> Customer information 5</ref5>

          <trnCardCvd>123</trnCardCvd>
          <username>APIusername</username>
          <password>APIpassword</password>

</transaction>
```

Transaction Response

In this basic response, an approved message is returned and a transaction ID is assigned. Additional AVS and CVD response messaging is also included. Order number and reference fields are returned untouched.

```
<response>
    <trnApproved>1</trnApproved>
    <trnId>11223344</trnId>
    <messageId>1</messageId>
    <messageText>Approved</messageText>
    <trnOrderNumber>SO5446</trnOrderNumber>
    <authCode>123</authCode>
    <errorType>N</errorType>
    <errorFields></errorFields>
    <responseType>T</responseType>
    <trnAmount>19.99</trnAmount>
    <trnDate>09/2/2010 4:15:55 PM</trnDate>
    <avsId>Z</avsId>
    <avsResult>0</avsResult>
    <avsAddrMatch>0</avsAddrMatch>
    <avsPostalMatch>1</avsPostalMatch>
    <avsMessage>Postal/ZIP matches, street address does not match.</avsMessage>
    <cvdId>2</cvdId>
    <cardType>MC</cardType>
    <trnType>P</trnType>
    <paymentMethod>CC</paymentMethod>
    <riskScore>5</riskScore>
    <ref1>Customer information 1</ref1>
    <ref2> Customer information 2</ref2>
    <ref3> Customer information 3</ref3>
    <ref4> Customer information 4</ref4>
    <ref5> Customer information 5</ref5>
</response>
```

6.2 Sample Verified by Visa (VBV) Request and Response

Request Scenario

In this basic VBV scenario, standard transaction details are submitted with an additional termURL value.

```
<transaction>
        <serviceVersion>1.3</serviceVersion>
        <merchant_id>123456789</merchant_id>
<username>user1234</username>
<password>password1234</password>

        <trnType>P</trnType>
        <trnOrderNumber>SO5544</trnOrderNumber>
        <trnAmount>11.50</trnAmount>
        <termUrl>https://www.mydomain.com/auth_redirect.asp</termUrl>

        <trnCardOwner>VBV Shopper</trnCardOwner>
        <trnCardNumber>4123450131003312</trnCardNumber>
        <trnExpMonth>05</trnExpMonth>
        <trnExpYear>12</trnExpYear>

        <ordEmailAddress>user@domain.com</ordEmailAddress>
        <ordName>VBV Shopper</ordName>
        <ordPhoneNumber>999-999-9999</ordPhoneNumber>
        <ordAddress1>123 Main Street</ordAddress1>
        <ordAddress2>Suite 331</ordAddress2>
        <ordCity>Victoria</ordCity>
        <ordProvince>BC</ordProvince>
        <ordPostalCode>V8T4R5</ordPostalCode>
        <ordCountry>CA</ordCountry>

        <ref1>Customer information 1</ref1>
        <ref2> Customer information 2</ref2>
        <ref3> Customer information 3</ref3>
        <ref4> Customer information 4</ref4>
        <ref5> Customer information 5</ref5>

        <vbvEnabled>1</vbvEnabled>
        <scEnabled>1</scEnabled>
        <trnCardCvd>123</trnCardCvd>

</transaction>
```

VBV Redirection Response Message

The gateway responds with the appropriate redirection method. Refer to the Process Transaction API guide for a complete description of the VBV redirection process.

```
<?xml version="1.0" standalone="yes"?>
<response>
<responseType>R</responseType>
<pageContents>
<HTML>
```

```
<HEAD></HEAD>
<BODY>

<FORM action="https://www.domain.com/gateway.asp" method=POST id=form1 name=form1>

<INPUT type=hidden name=PaReq value="TEST_paRaq">
<input type="hidden" name="merchant_name" value="MyCompanyName">
<input type="hidden" name="trnDatetime" value="11/21/2010 2:37:28 PM">
<input type="hidden" name="trnAmount" value="11.50">
<input type="hidden" name="trnEncCardNumber" value="XXXX XXXX XXXX 3312">

<INPUT type=hidden name=MD value="C1350A85-E6D9-43F6-BBFEB5B87800751F">
<INPUT type=hidden name=TermUrl value="https://www.domain.com/auth_redirect.asp">
</FORM>

<SCRIPT language="JavaScript">document.form1.submit();</SCRIPT>
</BODY>
</HTML>
</pageContents>
</response>
```

Parsing Response Variables into XML Format

Response redirection messages(where responseType=R) must be URL decoded.

```
if responseType = "R" then
'Redirect response
beanstreamResponse = xmlDecode(GetTagValue(trnResponse, "redirectionPage"))
response.write beanstreamResponse


Function xmlDecode(text)
 text = Replace(text, "&lt;", "<")
 text = Replace(text, "&gt;", ">")
 text = Replace(text, "&quot;", """")
 text = Replace(text, "&apos;", "'")
 text = Replace(text, "&amp;", "&")

 xmlDecode = text
End Function
```

VBV Authentication Request Message

When the customer completes VBV password authentication, the results of the authentication are sent back to the merchant via the merchant's Terminal URL page.

The merchant must formulate the following XML message and send it to Beanstream via the SOAP Transaction Process Auth method. Refer to the Process Transaction API guide for a complete description of this process.

```
<transaction>
<md>requestType=SOAP&amp;trnCardNumber=4123450131003312&amp;trnExpMonth=08&amp;trnE
xpYear=04&amp;trnType=P&amp;trnAmount=11.50&amp;merchant_id=123456789&amp;errorPage=ht
tps%3A%2F%2Fwww%2Ebeanstream%2Ecom%2Fsamples%2Forder_form.asp&amp;trnCardOwner=VBV
l+Shopper&amp;trnOrderNumber=SO5544&amp;ordEmailAddress=user@domain.com&amp;ordName=
VBV+Shopper&amp;ordPhoneNumber=9999999&amp;ordAddress1=123+Main+Street&amp;ordAddres
s2=Suite+331&amp;ordCity=Victoria&amp;ordProvince=BC&amp;ordPostalCode=V8R+1J6&amp;ordCou
ntry=CA&amp;TermUrl=https%3A%2F%2Fwww.beanstream.com%2Fsamples%2Fsample_s2s_vbv_auth.
asp&amp;xid=310</md>
<paRes>eJxlUdtuwjAMfd9XVH1f46R3ZIJgaBoPTLuwD8haCyrRFtJ2ZX+/BFrYtCiRfOwcX45xdir3zhfppqirqcs
9cB2qsjovqu3U/dg83ifuTN7hZqeJlu+UdZokrqlp1JacIjcUCIUf8DQUoeDmgIAE/CDy0yROeRS6El/mb3SUOF
SRpognkI3QZNPZTlWtRJUdF6tnGYQiBUA2QCxJr5ZSiAiCKDTMC8ZKlSQXpKqm1aRKZGcHZnVXtfpbchEgGw
F2ei93bXuYMNb3vfd5pXlZbag2jOzWyUtnrcakOxW5XC/n/f+3gvXmdYrM/sBctSQFgA8hxI7gEy7MRXb2oy
ptH3ZyO9YF4MHWmA8RG/jtQKO0NpsY5xgR0ulQV2R+mORXG9mt4YcnK2TWWsliDhH4wk9EGnCIraTng
M1SGFkgsM4BILNUNmyLDUs21p/l/wBZzLOy</paRes>
</transaction>
```

Sample Approved Transaction Response

The gateway receives the authorization request, processes the transaction and responds with a final response message.

```
<response>
   <trnApproved>1</trnApproved>
   <trnId>22446688</trnId>
   <messageId>1</messageId>
   <messageText>Approved</messageText>
   <trnOrderNumber>SO5446</trnOrderNumber>
   <authCode>123</authCode>
   <errorType>N</errorType>
   <errorFields></errorFields>
   <responseType>T</responseType>
   <trnAmount>11.50</trnAmount>
   <trnDate>11/21/2010 2:37:48 PM </trnDate>
   <avsId>Z</avsId>
   <avsResult>0</avsResult>
   <avsAddrMatch>0</avsAddrMatch>
```

```
        <avsPostalMatch>1</avsPostalMatch>
        <avsMessage>Postal/ZIP matches, street address does not match.</avsMessage>
        <cvdId>2</cvdId>
        <cardType>MC</cardType>
        <trnType>P</trnType>
        <paymentMethod>CC</paymentMethod>
        <riskScore>5</riskScore>
        <ref1>Customer information 1</ref1>
        <ref2> Customer information 2</ref2>
        <ref3> Customer information 3</ref3>
        <ref4> Customer information 4</ref4>
        <ref5> Customer information 5</ref5>
    </response>
```