
# Process Transaction API

Document Version 6.2
September, 2012

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

# Overview

Systems integrators and developers may connect the Beanstream gateway to custom payment pages and e-commerce processing systems using our XML-based API. This guide includes information on the API transaction protocol, input variables and response messages for processing purchases, returns and other transactions using the Beanstream gateway processing service. Reference this guide for information on implementing a custom connection to the Beanstream gateway.

# 1 Using this document

The Beanstream gateway includes multiple services and transaction processing options. Review the sections that are pertinent to the services you will be implementing for process flows, sample request strings and input variables. Refer to the table of Response Variables at the end of this document for a description of the parameters returned in Beanstream response strings.

Also, be aware that some advanced options must be activated by Beanstream before they will be available to merchants and their developers. Contact support@beanstream.com if you wish to activate a service or confirm availability.

| | Service | Description | Requires Activation |
|---|---|---|---|
| **Payment Methods** | Credit Card Processing | Accept popular credit cards online. | No |
| | INTERAC Online Processing | Accept real time bank payments. | Yes |
| | Direct Payments & ACH | Process bank to bank credits and debits. | Yes |
| **Billing Methods** | Payment Profiles | Store customer data on Beanstream's servers. | Yes |
| | Recurring Billing | Create automated billing schedules. | Yes |
| | Batch Processing | Upload files to process multiple transactions. | Yes |
| **Security and Authentication** | CVD &AVS | Use common card company security programs. | No |
| | VBV & Secure Code | Accept Visa and MasterCard secure pin numbers. | Yes |
| | Hash Validation | Submit Hash encrypted transaction requests. | No |
| | Username/Password Validation | Protect transaction requests with secure username and password parameters. | No |
| | Inventory Validation | Validate orders against product data stored on Beanstream's servers. | Yes |
| | Canadian Address Verification | Compare customer submitted information with data on file at the Equifax consumer bureau. | Yes |

# 2 System Requirements

The Beanstream API does not require the installation of a software development kit. System integrators should ensure that they have the following items in place for a successful implementation:

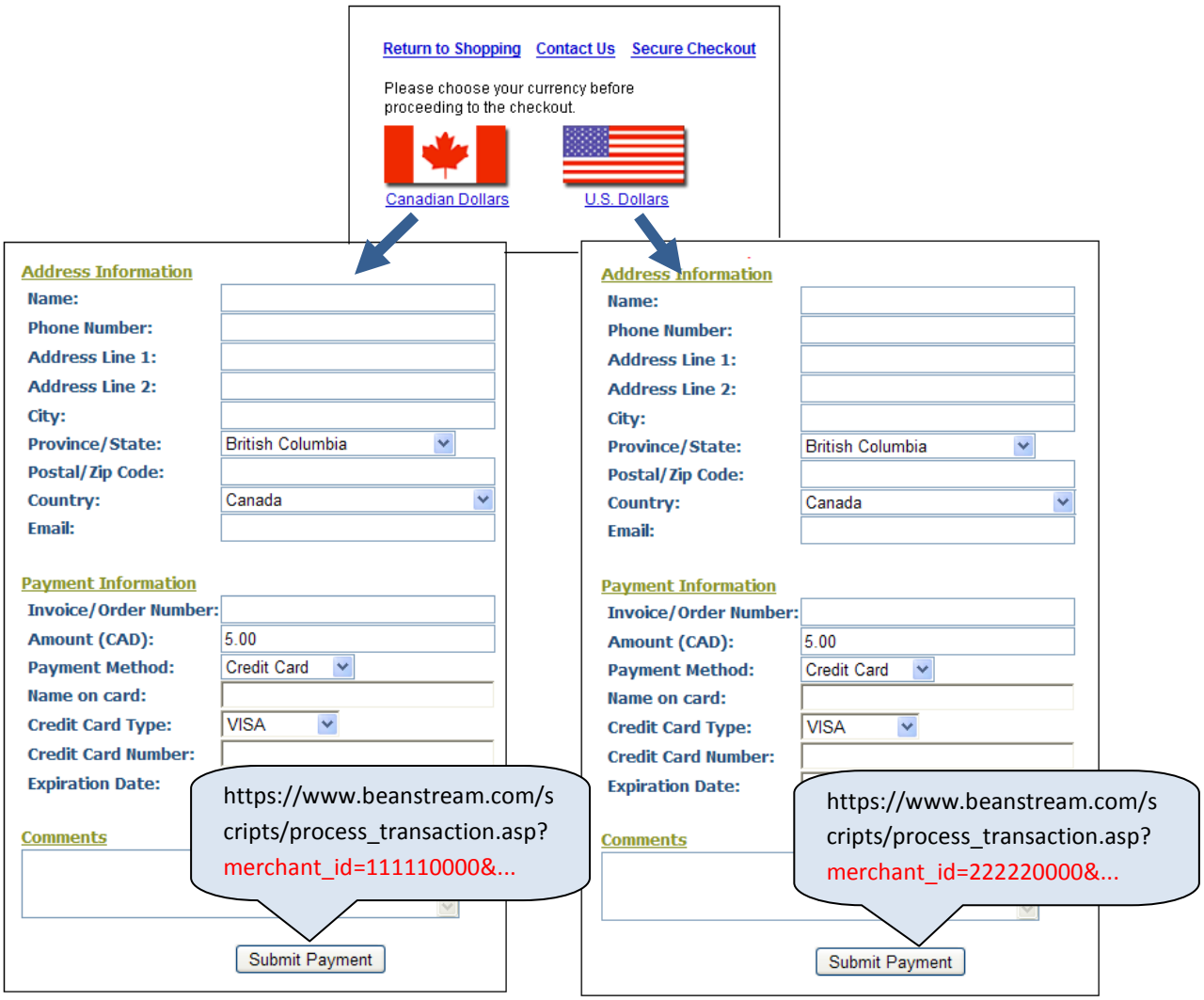
- The merchant must have (or be in the process of acquiring) active, compatible ecommerce merchant accounts for each of the card types in each of the currencies that they wish to process.
- The merchant must have one Beanstream gateway merchant ID and administrator account for each processing currency.
- The merchant's site must be able to communicate with the Beanstream web server via 40-bit or 128-bit SSL.
- Optional or value-added gateway service options must be activated by Beanstream.

# 3 Dual Currency Processing

The Beanstream system works by assigning unique identifiers to each merchant. These merchant IDs are key to correctly processing transactions in the right currency and with the correct services. Merchants are issued a 9-digit merchant ID for each processing currency. You must complete integration for each merchant ID that has been issued. Be sure to reference the correct number or your set up may not be successful. Contact support@beanstream.com if you would like to confirm your merchant ID numbers before you get started.

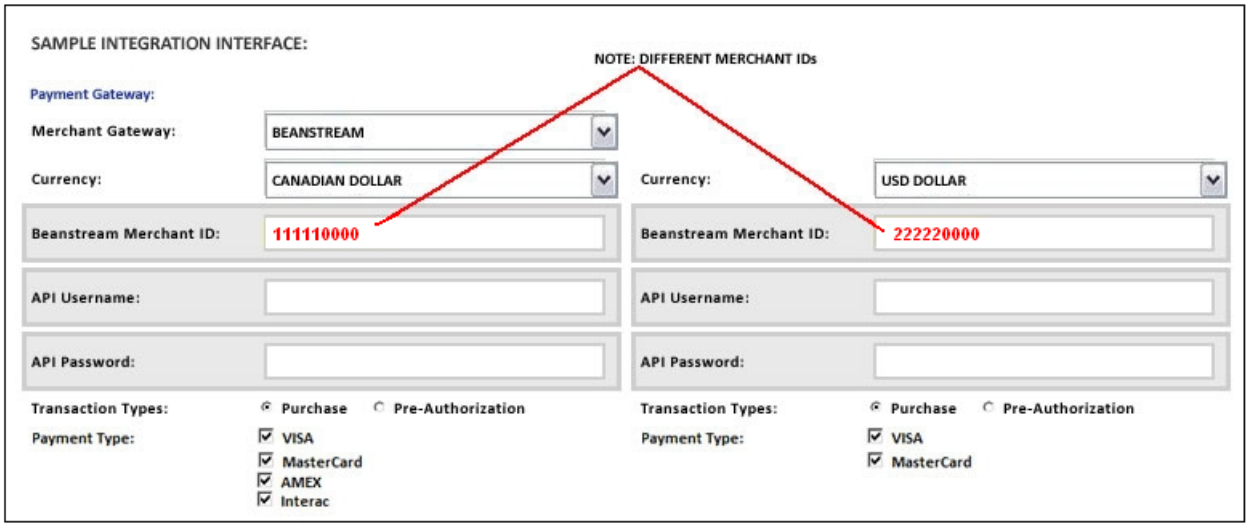
## Sample Dual Currency Site

In the simple setup shown below, the merchant offers clients the option of paying on a Canadian dollar or US dollar payment page. The code behind the submit buttons on the payment page directs the merchant to the correct Beanstream merchant ID.



## Sample Shopping Cart Integration

In the sample cart integration shown below, the shopping cart provider has created a simple interface to allow merchants to specify their unique merchant IDs and card types before customizing their shopping cart software. Remember that merchants must have active e-commerce merchant accounts issued by a compatible service provider in order to be able to process all payment types and all currencies shown in this image.



# 4 Test vs. Live Processing Environments

New merchants receive login information and merchant ID numbers for accounts that are in "test" mode. Transactions processed through the test environment are free of charge. Before processing live transactions, you must complete integration for each of your merchant IDs and respond to the following important communications from Beanstream.

### Authorized for Live Email

The Authorized for Live notification lets the merchant know they are ready to start processing transactions. We're waiting to hear back that the merchant's integration is complete. You'll still be able to process test transactions for free until the merchant confirms that they are "Ready for Testing". Monthly fees will be charged at this point.

### Site Review Emails

When integration is complete, notify us that you are "Ready for Testing." We'll do a quick review of the merchant's website to make sure that you've got everything running smoothly. Once we're satisfied, we'll let you know.

Remember to respond to our final email. We wait for the merchant's final authorization before turning an account Live.

In most cases, there will be no need to change merchant IDs between the test and live processing environments. However, developers may request a sandbox account if they wish to maintain a permanent testing environment. Those using a sandbox account will have separate sandbox and live ID numbers.

## 4.1 Test Card Numbers

Use the following card numbers to emulate the full transaction process in our test environment. These card numbers are not valid for live accounts. Use an expiry date that is equal to or later than the current month and year.

### Visa - Use CVD/CVV code “123

| | |
|---|---|
| Approved | 4030000010001234 |
| Approved $100 Limit | 4504481742333 |
| Approved VBV | 4123450131003312 with VBV passcode 12345 |
| Declined | 4003050500040005 |

### MasterCard - Use CVD/CVV code “123

| | |
|---|---|
| Approved | 5100000010001004 |
| Approved | 5194930004875020 |
| Approved | 5123450000002889 |
| Approved 3D Secure | 5123450000000000 passcode 12345 |
| Declined | 5100000020002000 |

### American Express - Use CVD/CVV code “1234”

| | |
|---|---|
| Approved | 371100001000131 |
| Declined | 342400001000180 |

### Discover - Use CVD/CVV code “123”

| | |
|---|---|
| Approved | 6011500080009080 |
| Declined | 6011000900901111 |

# 5 The Standard Transaction Process

With the Beanstream gateway, the basic transaction process occurs over three stages:

- The transaction is submitted to the API
- Automated error checks validate the information submitted in the request string
- The data is submitted to the bank and a response is returned to the merchant's server

The following sections describe in detail the process for handling this standard transaction flow.

## 5.1 Submitting the Transaction Request

**API Service URL** - https://www.beanstream.com/scripts/process_transaction.asp

Transaction details are sent to the Process Transaction API Service URL https://www.beanstream.com/scripts/process_transaction.asp as set of field name/value pairs and submitted through either a form post or a query string. Merchants may integrate using a Server to Server method or a basic HTTP POST. We do not recommend connecting to our processing server using the GET method. Data passed using GET will be visible in the browser's address bar meaning requests may be viewed at the time of submission. GET requests are also limited by the browser to an average of 1 k of data meaning large transactions may be truncated causing failure.

### Basic HTTP Post

A basic HTTP POST integration is the simplest way of integrating Beanstream's processing system. With this technique the customer's browser will be pointed to the Beanstream server at the time of processing. For this reason the basic HTTP POST option is sometimes referred to as a "redirection method." This option is particularly useful for merchants that wish to host payment pages on Beanstream's secure server. For other setups, we highly recommend using the more advanced Server to Server method for optimal security, and to achieve the full functionality of the Beanstream system.

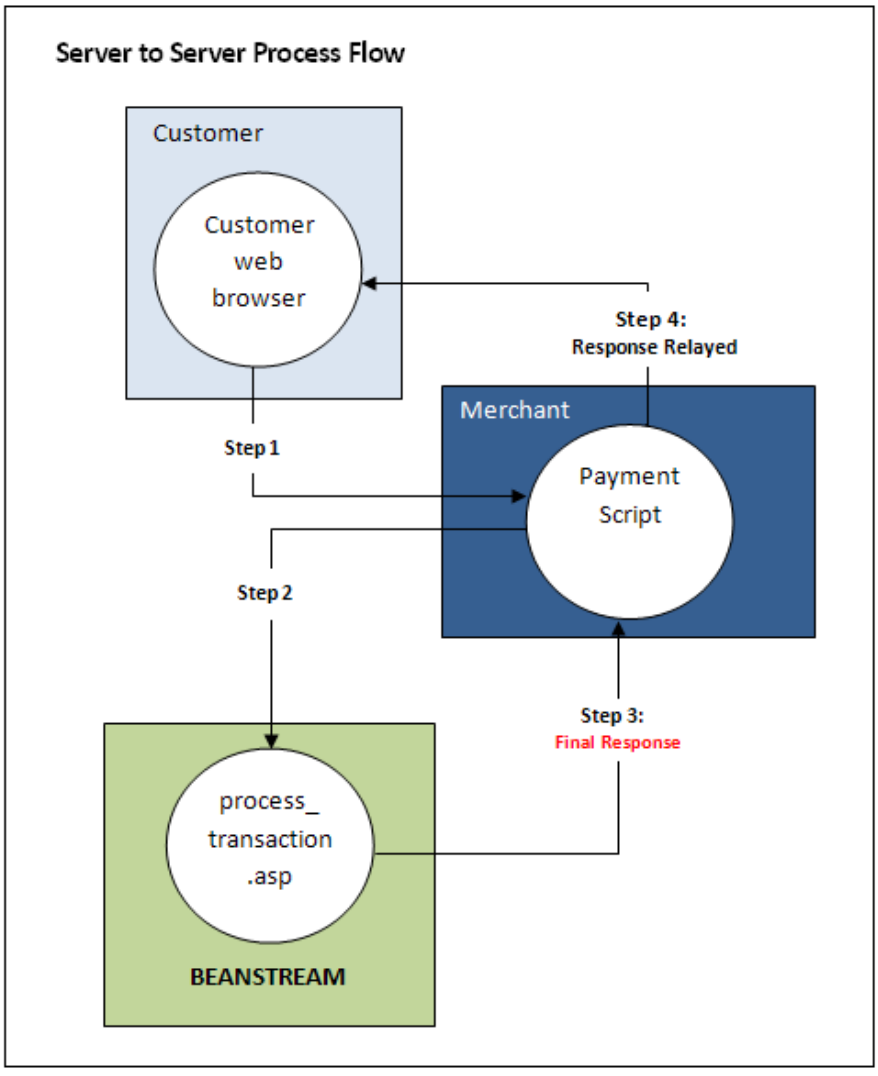
### Server to Server Protocol

Server to Server protocol opens a separate, secure session when sending customer transaction details to the Beanstream gateway. The merchant's processing script creates a browser object to POST the transaction request to the Beanstream API. After processing the request, Beanstream sends the transaction details and returns response messages back via the secure session. The customer is

informed of the transaction results through the secure session rather than being redirected to separate approved/declined response pages. There are many methods of construction and developers are encouraged to use their preferred platform and programming language.

**Advantages of Server-to-Server integration**

- Optimal transaction security
- Prevents browser redirects from occurring during the transaction process.
- Allows for querying results of transactions that have already been processed.




## SOAP Method

Beanstream also offers a SOAP interface for those that prefer to use this XML-based protocol. If you wish to use a SOAP integration, please consult our supplemental SOAP Guide for additional information.

## 5.2 Validation and Error Handling

### Form Field Validation (User Generated Errors)

Beanstream handles field validation in different ways depending on the integration method chosen.

In a basic HTTP Post, transaction requests must contain an **errorPage** variable. This variable specifies the URL where customers will be directed in the case of a form field entry error. If a customer attempts to submit a transaction with missing or invalid billing information, the full transaction request string is returned to the errorPage along with two additional error response variables. An **errorFields** variable will contain a list of all fields that failed. An **errorMessage** field provides descriptive text to indicate the reasons why a submission failed field validation. This descriptive text may be displayed to customers if desired.

**Sample HTTP Post Error Response Notification:**

**errorMessage**=%3CLI%3ECard+owner+name+is+missing%3Cbr%3E%3CLI%3EInvalid+Card+Number%3Cbr%3E%3CLI%3EEnter+your+email+address%3Cbr%3E%3CLI%3EPhone+number+must+be+between+7+and+32+characters+long%3Cbr%3E%3CLI%3EInvalid+expiry+date%3Cbr%3E&**errorFields**=trnCardOwner%2CtrnCardNumber%2CordEmailAddress%2CordPhoneNumber%2CtrnExpMonth&merchant_id=123450000&trnType=P&errorPage=https%3A%2F%2Fwww%2Ebeanstream%2Ecom%2Fsecure%2FABCEnterprises%2Fselect%2Easp&approvedPage=https%3A%2F%2Fwww%2Ebeanstream%2Ecom%2Fsecure%2FABCEnterprises%2Fpost%5Fproc%2Easp&declinedPage=https%3A%2F%2Fwww%2Ebeanstream%2Ecom%2Fsecure%2FABCEnterprises%2Fpost%5Fproc%2Easp&ref1=&ref2=&ref3=&ref4=&ref5

In Server to Server integrations, error messages are returned as part of the standard URL encoded transaction response string. The **errorType** response variable will indicate "U" if a form field error occurs. The errorFields variable will contain a list of fields that failed validation. **errorMessage** will contain descriptive text that may be displayed to customers if desired.

**Sample Server to Server Error Response**

trnApproved=0&trnId=0&messageId=0&**messageText**=%3CLI%3ECard+owner+name+is+missing%3Cbr%3E%3CLI%3EInvalid+Card+Number%3Cbr%3E%3CLI%3EEnter+your+email+address%3Cbr%3E%3CLI%3EPhone+number+must+be+between+7+and+32+characters+long%3Cbr%3E%3CLI%3EInvalid+expiry+date%3Cbr%3E&&trnOrderNumber=E40089&authCode=TEST&**errorType=**U&**errorFields**=trnCardOwner%2CtrnCardNumber%2CordEmailAddress%2CordPhoneNumber%2CtrnExpMonth&responseType=T&trnAmount=10%2E00&trnDate=1%2F17%2F2008+11%3A36%3A34+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&rspCodeCav=0&rspCavResult=0&rspCodeCredit1=0&rspCodeCredit2=0&rspCodeCredit3=0&rspCodeCredit4=0&rspCodeAddr1=0&rspCodeAddr2=0&rspCodeAddr3=0&rspCodeAddr4=0&rspCodeDob=0&rspCustomerDec=&trnType=P&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

## System Generated Errors

System generated errors provide messaging to notify developers of poorly formatted request strings. These errors are designed to assist with troubleshooting during the initial development stages. If integration has been properly completed, they should not appear once an account is Live. System generated errors are displayed directly on a Beanstream error page. They are not returned in a server to server transaction response or displayed on the dedicated error page for HTTP Post integrations. Messages include:

- Connection is not secure
- Invalid merchant ID
- Authorization failed
- Missing transaction data
- Missing errorPage address (basic HTTP Post integrations only)

System generated errors can be identified in a Server to Server integration by a response message "errorType=S" in the Beanstream response string. If a system generated error occurs, validate your integration and website setup.

## Duplicate Transactions

Beanstream will automatically check and block duplicate transactions. In order for a transaction to qualify as a duplicate, the following fields must contain identical information to another transaction processed within the same hour:

- Transaction Amount
- Transaction Type
- Credit Card Number
- Order Number (if passed with the transaction request)

Duplicate transactions are returned with the response messageId =16.

**Sample Duplicate Transaction Response**

trnApproved=0&trnId=10000075&**messageId**=16&**messageText**=Duplicate+Transaction+%2D+This+transaction+has+already+been+approved …

## 5.3 Transaction Completion

After order information has been validated, the transaction is passed to the bank for authorization. A dedicated response message and code is assigned to indicate if the transaction has been approved or declined.

In a basic HTTP Post, Beanstream will automatically direct the customer to a transaction approved or declined page. These pages may be Beanstream's default hosted approved and declined pages or they may be custom pages if approvedPage and declinedPage variables were sent with the transaction request.

In a Server to Server integration, Beanstream Posts a response message to the merchant's server including full transaction confirmation details. The merchant integration must parse out the messaging and display responses to the customer in the desired format.

### Sample Approved Transaction Response (HTTP Post)

https://www.mydomain.com/approved_page.asp?**trnApproved**=1&trnId=10000083&messageId=1&**messageText**=Approved&authCode=TEST&**responseType**=T&trnAmount=5.50&trnDate=8%2F24%2F2009+11%3A31%3A56+AM&trnOrderNumber=10000083&trnLanguage=eng&trnCustomerName=Mary+Smith&trnEmailAddress=msmith%40mydomain%2Ecom&trnPhoneNumber=250%2D123%2D0001&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=P&ref1=&ref2=&ref3=&ref4=&ref5=

### Sample Approved Transaction Response (Server to Server)

**trnApproved**=1&trnId=10003067&messageId=1&**messageText**=Approved&trnOrderNumber=E40089&authCode=TEST&**errorType**=N&errorFields=&**responseType**=T&trnAmount=10%2E00&trnDate=1%2F17%2F2008+11%3A36%3A34+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&rspCodeCav=0&rspCavResult=0&rspCodeCredit1=0&rspCodeCredit2=0&rspCodeCredit3=0&rspCodeCredit4=0&rspCodeAddr1=0&rspCodeAddr2=0&rspCodeAddr3=0&rspCodeAddr4=0&rspCodeDob=0&rspCustomerDec=&trnType=P&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

### Response Notification Pages

Integrators may set up the system to send automated responses to a dedicated notification page. This feature is designed for merchants that wish to receive an HTTP POST transaction response notification at a specified URL. Response notification pages for Recurring and Payment Profiles transactions must be set separately.

- Go to *Administration* → *Account Settings* → *Order Settings*.
- On the *Order Settings* page, use the fields provided to enter a URL for your notification page(s).
- Click on *Update* to save your changes.

# 6 Credit Card Purchases

The Beanstream gateway supports Canadian and US dollar processing for Visa, MasterCard, American Express, Diners, Discover, JCB and Sears. While the Beanstream system can handle all of these cards, merchants must acquire merchant accounts for each card type they wish to accept on their website.

## 6.1 Standard Purchase Flow

A standard credit card purchase is the simplest type of transaction to be processed through the Beanstream system. These purchases will follow the basic transaction flow exactly as outlined in section 5. The following sample request string shows the information that must be submitted to the Process Transaction API to perform a basic credit card purchase using Server to Server integration.

### Sample Transaction Request

https://www.beanstream.com/scripts/process_transaction.asp?merchant_id=123456789&requestType=BACKEND&trnType=P&trnOrderNumber=1234TEST&trnAmount=5.00&trnCardOwner=Joe+Test&trnCardNumber=4030000010001234&trnExpMonth=10&trnExpYear=10&ordName=Joe+Test&ordAddress1=123+Test+Street&ordCity=Victoria&ordProvince=BC&ordCountry=CA&ordPostalCode=V8T2E7&ordPhoneNumber=5555555555&ordEmailAddress=joe%40testemail.com

On transaction completion, Beanstream will return a transaction response message. In the following sample response string, blue text indicates the fields that must be displayed to the customer. Other fields are for your reference purposes and include information on errors, AVS validation and other services if applicable. The “ref” variables in blue would include custom order information if this data was included in the transaction request.

### Sample Transaction Response

trnApproved=1&trnId=10001364&messageId=1&messageText=Approved&trnOrderNumber=1234TEST&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=5%2E00&trnDate=7%2F31%2F2009+11%3A57%3A12+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=P&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

On receipt of the transaction response, the merchant must display order amount, transaction ID number, bank authorization code (authCode), currency, date and “messageText” to the customer on a confirmation page.

## 6.2 VBV and SecureCode Purchase Flow

Verified by Visa (VBV) and Secure Code (SC) are security features that prompt customers to enter a passcode when they pay by Visa or MasterCard. Merchants that wish to integrate VBV or Secure Code must have signed up for the service through their bank merchant account issuer. This service must also be enabled by the Beanstream support team. Contact support@beanstream.com to confirm availability before integrating.

In a VBV or SC transaction, the customer is redirected to a bank portal to enter their secure pin number before a transaction is processed. The bank returns an authentication response which must be forwarded to Beanstream in order for a transaction to complete. This process may be implemented in one of two ways.

### 4.2.1 VBV/SC Certified Merchants

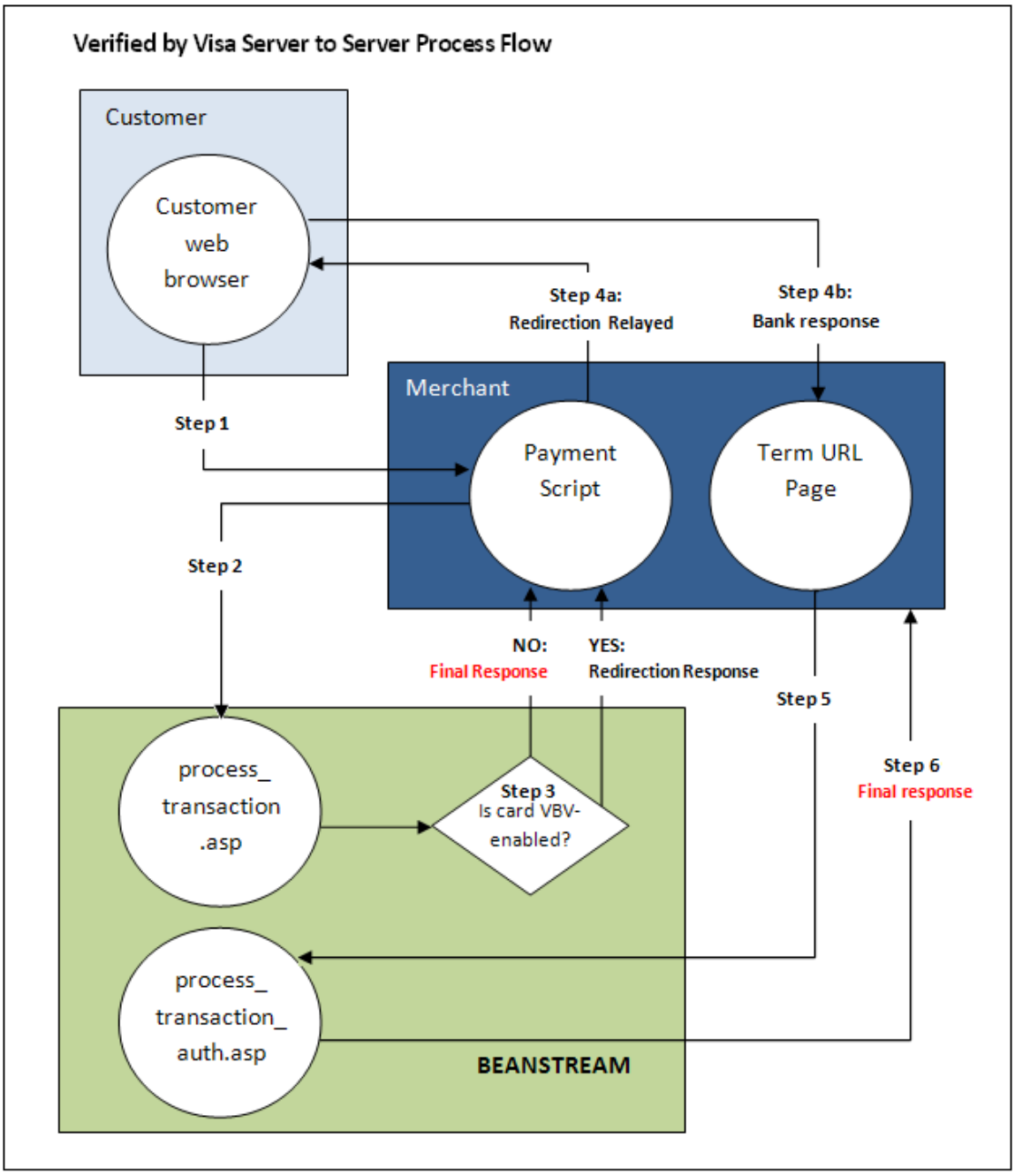
Some large merchants may have completed VBV/SC certification to handle VBV/SC authentication on their own side. These merchants may use their existing VBV/SC authentication process and send the results of the bank authentication to Beanstream with their standard transaction request. To do this, the merchant must integrate using a Server to Server type connection. The VBV/SC bank authentication results must be sent with the transaction request using the following three system variables:

| | |
|---|---|
| **SecureXID** | Include the 20 digit 3D secure transaction identifier |
| **SecureECI** | Provide a 1 digit ECI status code.<br>5=authenticated, 6=attempted but not completed |
| **SecureCAVV** | Include the 40 character Cardholder Authentication Verification Value |

This option must be enabled by Beanstream. Please notify our support team if you wish to use this method.

### 4.2.2 All Other Merchants

The majority of merchants must go through Beanstream to both initiate the VBV/SC process and complete the transaction request. In this standard integration, the VBV and SC process will require two transaction requests as described below.




## Step 1: Submitting the Transaction

The customer browses the merchant's website and navigates to an order payment page where they choose to make a purchase from the merchant's website using a credit card. They complete their order information and submit the transaction to the merchant processing script.

## Step 2: Beanstream Process Transaction Request

The merchant's processing script forwards the transaction details to Beanstream. The request includes a special termURL variable. This termURL variable allows the merchant to specify the URL where the bank VBV or SC response will be returned after the customer PIN number has been entered and verified on the bank portal.

**Sample Request String (Server to Server)**

requestType=BACKEND&merchant_id=109040000&trnCardOwner=Paul+Randal&trnCardNumber=4030000010001234&trnExpMonth=01&trnExpYear=05&trnOrderNumber=1234&trnAmount=10.00&ordEmailAddress=prandal@mydomain.net&ordName=Paul+Randal&ordPhoneNumber=6042229999&ordAddress1=1045+Main+Street&ordAddress2=&ordCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCountry=CA&**termUrl**=https%3A%2F%2Fwww%2Emerchantserver%2Ecom%2Fauth_script.asp

## Step 3: Beanstream Reviews and Responds

Beanstream verifies that the card is VBV or SecureCode enabled. If the customer has not signed up for VBV or SecureCode service (and does not have the feature enabled on their card), the transaction proceeds as normal. If the card is VBV or SC enabled, Beanstream responds with a JavaScript redirection response message. This response string includes the variable trnResponseType=R and a URL encoded pageContents variable.

**Sample Response Redirect**

**responseType**=R%26**pageContents**=%3CHTML%3E%3CHEAD%3E%3C%2FHEAD%3E%3CBODY%3E%3CFORM%20action%3D%22https%3A%2F%2Fwww.vbvgateway.asp%22%20method%3DPOST%20id%3Dform1%20name%3Dform1%3E%3CINPUT%20type%3Dhidden%20name%3DPaReq%20value%3D%22TEST_paRaq%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22merchant_name%22%20value%3D%22TEST%20Company%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22trnDatetime%22%20value%3D%223%2F3%2F2008%202%3A15%3A38%20PM%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22trnAmount%22%20value%3D%22100.00%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22trnEncCardNumber%22%20value%3D%22XXXX%20XXXX%20XXXX%203312%22%3E%3CINPUT%20type%3Dhidden%20name%3DMD%20value%3D%2265523BC5-5551-4CAF-AE7727CAA393B0F9%22%3E%3CINPUT%20type%3Dhidden%20name%3DTermUrl%20value%3D%22http%3A%2F%2Fwww.myCompanyTermUrl.asp%22%3E%3C%2FFORM%3E%3CSCRIPT%20language%3D%22JavaScript%22%3Edocument.form1.submit()%3B%3C%2FSCRIPT%3E%3C%2FBODY%3E%3C%2FHTML%3E

## Step 4a: Forward to the Bank Portal

The merchant's processing script URL decodes the response redirect and displays the information to the customer's web browser. This forwards the client to the VBV or SC banking portal. On the bank portal, the customer enters their secure credit card pin number in the fields provided on the standard banking interface.

**Sample URL decoded response**

responseType=R&pageContents=<HTML><HEAD></HEAD><BODY><FORM action="https://www.vbvgateway.asp" method=POST id=form1 name=form1><INPUT type=hidden name=PaReq value="TEST_paRaq"><input type="hidden" name="merchant_name" value="TEST Company"><input type="hidden" name="trnDatetime" value="3/3/2008 2:15:38 PM"><input type="hidden" name="trnAmount" value="100.00"><input type="hidden" name="trnEncCardNumber" value="XXXX XXXX XXXX 3312"><INPUT type=hidden name=MD value="65523BC5-5551-4CAF-AE7727CAA393B0F9"><INPUT type=hidden name=TermUrl value="http://www.myCompanyTerm_Url.asp"></FORM><SCRIPT language="JavaScript">document.form1.submit();</SCRIPT></BODY></HTML>

## Step 4b: Bank Response

The bank forwards a response to the merchant's TERM URL including the following variables:

| |
|---|
| **PaRes (VBV Authentication Code)** |
| **MD (Unique Payment ID)** |

## Step 5: Process Transaction Auth Request

The merchant takes the data posted to the TERM URL and posts the PaRes and MC variables to www.beanstream.com/scripts/process_transaction_auth.asp .

## Step 6: Approval/Decline Response

If the transaction fails VBV or SC it is declined immediately with messageId=311 (3d Secure Failed). If the transaction passes, it is forwarded to the banks for processing. On completion, an approved or declined message is sent to the merchant processing script.

**Sample Approved Transaction Response**

trnApproved=1&trnId=10003067&messageId=1&messageText=Approved&trnOrderNumber=E40089&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=10%2E00&trnDate=1%2F17%2F2008+11%3A36%3A34+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&rspCodeCav=0&rspCavResult=0&rspCodeCredit1=0&rspCodeCredit2=0&rspCodeCredit3=0&rspCodeCredit4=0&rspCodeAddr1=0&rspCodeAddr2=0&rspCodeAddr3=0&rspCodeAddr4=0&rspCodeDob=0&rspCustomerDec=&trnType=P&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

On receipt of the transaction response, the merchant must display order amount, transaction ID number, bank authorization code (authCode), currency, date and "messageText" to the customer on a confirmation page.

## 6.3 Credit Card Purchase Variables

Server to Server  Basic HTTP Post

| | Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|---|
| **Basic API Call** | requestType | R | BACKEND | Enter requestType=BACKEND for the recommended server to server integration method. Note that server to server typically cannot be used when hosting forms in the Beanstream Secure Webspace. |
| | merchant_id | R | 9-digits | Beanstream assigns one merchant ID number for each processing currency. Include the 9-digit Beanstream ID number here. Additional accounts may also have been issued for special services. Complete one full integration for each of the merchant IDs issued. |
| | trnOrderNumber | Optional but Recommended | 30 alphanumeric (a/n) characters | Include a unique order reference number if desired. If no number is passed, Beanstream will place the default transaction identification number (trnId) in this field. Custom order numbers will be used in duplicate transaction error checking. Order numbers are also required for Server to Server transaction queries. Integrators that wish to use the query function should pass custom values. |
| | trnAmount | R | In the format 0.00. Max 2 decimal places. Max 9 digits total. | This is the total dollar value of the purchase. This should represent the total of all taxes, shipping charges and other product/service costs as applicable. |

| | | | | |
|---|---|---|---|---|
| | errorPage | R | URL (encoded). Max 128 a/n characters. | Not for use with server to server integrations. If a standard transaction request contains errors in billing or credit card information, the customer's browser will be re-directed to this page. Error messages will prompt the user to correct their data. |
| | approvedPage | O | URL (encoded). Unlimited a/n characters. | Beanstream provides default approved or declined transaction pages. For a seamless transaction flow, design unique pages and specify the approved transaction redirection URL here. |
| | declinedPage | O | URL (encoded). Unlimited a/n characters. | Specify the URL for your custom declined transaction notification page here. |
| **Credit Card Purchase**<br><br>* By default, billing address information and card owner name are required parameters. This default setting may be changed in the order settings module of the member area. | trnCardOwner | R* | Max 64 a/n characters | This field must contain the full name of the card holder exactly as it appears on their credit card. |
| | trnCardNumber | R | Max 20 digits | Capture the customer's credit card number. |
| | trnExpMonth | R | 2 digits (January = 01) | The card expiry month with January as 01 and December as 12. |
| | trnExpYear | R | 2 digits (2011=11) | Card expiry years must be entered as a number less than 50. In combination, trnExpYear and trnExpMonth must reflect a date in the future. |
| | trnCardCvd | O | 4 digits Amex, 3 digits all other cards | Include the three or four-digit CVD number from the back of the customer's credit card. This information may be made mandatory using the "Require CVD" option in the Beanstream Order Settings module. |

| | | | | |
|---|---|---|---|---|
| | ordName | R* | Max 64 a/n characters. | Capture the first and last name of the customer placing the order. This may be different from trnCardOwner. |
| | ordEmailAddress | R | Max 64 a/n characters in the format a@b.com. | The email address specified here will be used for sending automated email receipts. |
| | ordPhoneNumber | R* | Min 7 a/n characters Max 32 a/n characters | Collect a customer phone number for order follow-up. |
| | ordAddress1 | R* | Max 64 a/n characters | Collect a unique street address for billing purposes. |
| | ordAddress2 | O | Max 64 a/n characters | An optional variable is available for longer addresses. |
| | ordCity | R* | Max 32 a/n characters | The customer's billing city. |
| | ordProvince | R* | 2 characters | Province and state ID codes in this variable must match one of the available province and state codes. |
| | ordPostalCode | R* | 16 a/n characters | Indicates the customer's postal code for billing purposes. |

| | | | | |
|---|---|---|---|---|
| | ordCountry | R* | 2 characters | Country codes must match one of the available ISO country codes. |
| **Standard VBV/SC** | termURL | R | URL (encoded) | Specify the URL where the bank response codes will be collected after enters their VBV or SecureCode pin on the banking portal. |
| | vbvEnabled | O | 1 digit | When VBV service has been activated, Beanstream will attempt VBV authentication on all transactions. Use this variable to override our default settings and process VBV on selected transactions only. Pass vbvEnabled=1 to enable VBV authentication with an order. Pass vbvEnabled=0 to bypass VBV authentication on specific orders. |
| | scEnabled | O | 1 digit | When SecureCode service has been activated, Beanstream will attempt SC authentication on all transactions. Use this variable to override our default settings and process SC on selected transactions only. Pass scEnabled=1 to enable SC authentication with an order. Pass scEnabled=0 to bypass SC authentication on specific orders. |
| **VBV &SC**<br><br>for Self-Certified Merchants only | SecureXID | R | 20 digits | Include the 3D secure transaction identifier as issued by the bank following VBV or SecureCode authentication. |
| | SecureECI | R | 1 digit | Provide the ECI status. 5=transaction authenticated. 6= authentication attempted but not completed. |
| | SecireCAVV | R | 40 a/n characters | Include the cardholder authentication verification value as issued by the bank. |

## 6.4 Compatible Gateway Options

Credit Card purchases may be processed using a variety of other gateway tools to enhance security or help streamline the transaction process.

| Security Features | Order Customization | Advanced Processing Options |
|---|---|---|
| CVD (CVV) and AVS | Order Comments | Payment Profiles |
| Hash Validation | Custom Reference Variables | Recurring Billing |
| Username/Password | Shipping Details | Batch Processing |
| Inventory Validation | Product Details | |
| CAV *refer to our CAV Guide | | |

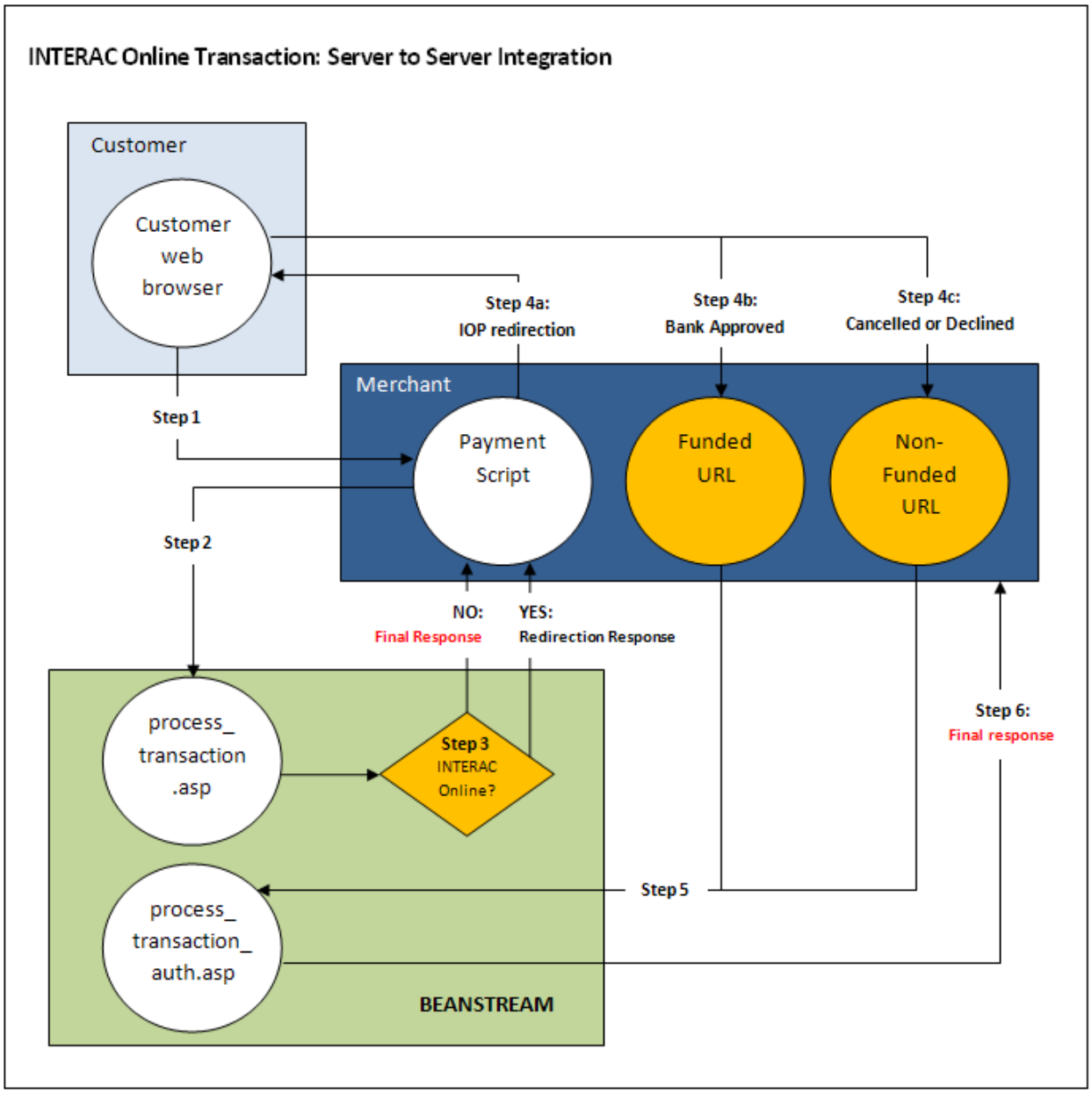
# 7 INTERAC Online Purchases

Beanstream's INTERAC Online service allows consumers to pay for purchases directly from their bank account as they would when using a debit card at a traditional bricks and mortar store. INTERAC Online transactions are authorized in real time; however the end customer is required to leave the merchant's site and go to their web banking portal to authorize their purchase.

The INTERAC Online service also has several unique design requirements that will be verified before Beanstream will authorize and activate this payment option on a live website. As part of your integration process, we recommend reviewing our supplemental INTERAC Online Guide for additional details on logo and wordmark use and required page elements.

## 7.1 Standard Purchase Flow

Like VBV and SecureCode, the INTERAC Online process requires two transaction requests: one to https://www.beanstream.com/scripts/process_transaction.asp and a second to https://www.beanstream.com/scripts/process_transaction_auth.asp. The transaction process takes place over six basic steps as described below.




## Step 1: Submitting the Transaction

The customer browses the merchant's website and navigates to an order payment page where they choose to make a purchase from the merchant's website using the INTERAC Online service. They complete their order information and submit the transaction to the merchant processing script.

## Step 2: Beanstream Process Transaction Request

The merchant's processing script forwards the transaction details to Beanstream. This time, the request does not include card information. Instead, a paymentMethod=IO variable is sent.

**Sample Request String (Server to Server)**

requestType=BACKEND&merchant_id=109040000&trnCardOwner=Paul+Randal& **paymentMethod**=IO &trnOrderNumber=1234&trnAmount=10.00&ordEmailAddress=prandal@mydomain.net&ordName=Paul +Randal&ordPhoneNumber=6042229999&ordAddress1=1045+Main+Street&ordAddress2=&ordCity=Van couver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCountry=CA

## Step 3: Beanstream Reviews and Responds

Beanstream confirms that the transaction is an INTERAC Online order (paymentMethod=IO). If the paymentMethod variable is not present, or if paymentMethod=CC, the transaction proceeds as a standard credit card transaction. For INTERAC Online orders, Beanstream responds with a JavaScript redirection response message. This response string includes the variable trnResponseType=R and a URL encoded pageContents variable.

**Sample Redirection Response**

**responseType**=R&**pageContents**=%3CHTML%3E%3CHEAD%3E%3C%2FHEAD%3E%3CBODY%3E%3CFORM %20action%3D%22https%3A%2F%2FiOnlinegateway.asp%22%20method%3DPOST%20id%3DfrmIOnline% 20name%3DfrmIOnline%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22IDEBIT_MERCHNUM %22%20%20value%3D%2212345678911%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22 IDEBIT_AMOUNT%22%20%20value%3D%2210000%22%3E%3Cinput%20type%3D%22hidden%22%20nam e%3D%22IDEBIT_TERMID%22%20value%3D%2212345678%22%3E%3Cinput%20type%3D%22hidden%22 %20name%3D%22IDEBIT_CURRENCY%22%20value%3D%22CAD%22%3E%3Cinput%20type%3D%22hidde n%22%20name%3D%22IDEBIT_INVOICE%22%20value%3D%22%22%3E%3Cinput%20type%3D%22hidden %22%20name%3D%22IDEBIT_MERCHDATA%22%20value%3D%222F86D946-5531-4495-9D82D7E6D83BA93%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22IDEBIT_FUNDEDURL %22%20value%3D%22http%3A%2F%2Fwww.myCompany.asp%3Ffunded%3D1%22%3E%3Cinput%20type %3D%22hidden%22%20name%3D%22IDEBIT_NOTFUNDEDURL%22%20value%3D%22http.www.myComp any.asp%3Ffunded%3D0%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%22merchant_nam e%22%20value%3D%22Test%20Company%22%3E%3Cinput%20type%3D%22hidden%22%20name%3D%2 2referHost%22%20value%3D%22http%3A%2F%2Fwww.myCompany.asp%22%3E%3Cinput%20type%3D% 22hidden%22%20name%3D%22referHost2%22%20value%3D%22%22%3E%3Cinput%20type%3D%22hidd en%22%20name%3D%22referHost3%22%20value%3D%22www.myCompany.asp%22%3E%3Cinput%20ty pe%3D%22hidden%22%20name%3D%22IDEBIT_MERCHLANG%22%20value%3D%22en%22%3E%3Cinput %20type%3D%22hidden%22%20name%3D%22IDEBIT_VERSION%22%20value%3D%221%22%3E%3C%2FF ORM%3E%3CSCRIPT%20language%3D%22JavaScript%22%3Edocument.frmIOnline.submit()%3B%3C%2FS CRIPT%3E%3C%2FBODY%3E%3C%2FHTML%3E

### Step 4a: Forward to the Bank Portal

The merchant's processing script URL decodes the response message and displays the information to the customer's web browser to forward the client to the INTERAC Online portal. From the INTERAC Online portal, the customer selects a bank, logs into their account and authorizes the transaction.

**Sample URL Decoded Response**

responseType=R&pageContents=<HTML><HEAD></HEAD><BODY><FORM action="https://iOnlinegateway.asp" method=POST id=frmIOnline name=frmIOnline><input type="hidden" name="IDEBIT_MERCHNUM" value="12345678911"><input type="hidden" name="IDEBIT_AMOUNT" value="10000"><input type="hidden" name="IDEBIT_TERMID" value="12345678"><input type="hidden" name="IDEBIT_CURRENCY" value="CAD"><input type="hidden" name="IDEBIT_INVOICE" value=""><input type="hidden" name="IDEBIT_MERCHDATA" value="2F86D946-5531-4495-9D82D7E6D83BA93"><input type="hidden" name="IDEBIT_FUNDEDURL" value="http://www.myCompany.asp?funded=1"><input type="hidden" name="IDEBIT_NOTFUNDEDURL" value="http.www.myCompany.asp?funded=0"><input type="hidden" name="merchant_name" value="Test Company"><input type="hidden" name="referHost" value="http://www.myCompany.asp"><input type="hidden" name="referHost2" value=""><input type="hidden" name="referHost3" value="www.myCompany.asp"><input type="hidden" name="IDEBIT_MERCHLANG" value="en"><input type="hidden" name="IDEBIT_VERSION" value="1"></FORM><SCRIPT language="JavaScript">document.frmIOnline.submit();</SCRIPT></BODY></HTML>

### Step 4b: Bank Response

If the transaction is cancelled or declined at any point, the bank forwards a response to the merchant's NON_FUNDED URL. Otherwise, the bank response is forwarded to the merchant's FUNDED URL. The funded and non-funded URLs are values that the merchant must provide to Beanstream before account activation. These values are stored internally by Beanstream.

**Sample Bank Response**

funded=1bank_choice=1&merchant_name=Flow+Demo+Test&confirmValue=&headerText=&IDEBIT_MERCHDATA=2F86D946-5531-4495-9D82D7E6D83BA93&IDEBIT_INVOICE=&IDEBIT_AMOUNT=10000&IDEBIT_FUNDEDURL=http%3A%2F%2F24.69.140.148%2Fasp%2Fdemo_scripts%2Fflow_demo.asp%3Ffunded%3D1&IDEBIT_NOTFUNDEDURL=http%3A%2F%2F24.69.140.148%2Fasp%2Fdemo_scripts%2Fflow_demo.asp%3Ffunded%3D0&IDEBIT_ISSLANG=en&IDEBIT_TRACK2=372802490654059121 4%3D12010123456789XYZ&IDEBIT_ISSCONF=CONF%23TEST&IDEBIT_ISSNAME=TestBank1&IDEBIT_VERSION=1&accountType=Chequing

### Step 5: Process Transaction Auth request

The merchant takes the data posted to the funded or non-funded URL and sends a new request string to www.beanstream.com/scripts/process_transaction_auth.asp . The following variables must be included:

| | |
|---|---|
| funded | IDEBIT_NOTFUNDEDURL |
| IDEBIT_TRACK2 | IDEBIT_ISSLANG |
| IDEBIT_VERSION | IDEBIT_ISSCONF |
| IDEBIT_MERCHANTDATA | IDEBIT_AMOUNT |
| IDEBIT_INVOICE | IDEBIT_FUNDEDURL |

### Step 6: Approval/Decline Response

Beanstream approves or declines the transaction and forwards a response message to the merchant. The transaction response includes a special INTERAC Online confirmation code (ioConfCode) and an INTERAC Online financial institution name (ioInstName). If the transactions was cancelled or rejected by the bank in Step 5, these variables will not be included in the response string.

**Sample Approved Transaction Response (funded transaction)**

trnApproved=1&trnId=10003067& **ioConfCode**=CONF%23TEST&**ioInstName**=TestBank1 messageId=1&messageText=Approved&trnOrderNumber=E40089&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=10%2E00&trnDate=1%2F17%2F2008+11%3A36%3A34+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&rspCodeCav=0&rspCavResult=0&rspCodeCredit1=0&rspCodeCredit2=0&rspCodeCredit3=0&rspCodeCredit4=0&rspCodeAddr1=0&rspCodeAddr2=0&rspCodeAddr3=0&rspCodeAddr4=0&rspCodeDob=0&rspCustomerDec=&trnType=P&paymentMethod=IO&ref1=&ref2=&ref3=&ref4=&ref5=

On receipt of the transaction response, the merchant must display the following information to the customer on a confirmation page:

- Transaction Id number
- Order Number
- Purchase Amount
- Currency
- Financial Institution Confirmation Code
- Financial Institution Name
- Response Message Text
- Transaction Date

## 7.2 INTERAC Online Input Variables

Server to Server  Basic HTTP Post

| | Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|---|
| **Basic API Call** | requestType | R | BACKEND | Enter requestType=BACKEND for the recommended server to server integration method. Note that server to server typically cannot be used when hosting forms in the Beanstream Secure Webspace. |
| | merchant_id | R | 9-digits | Beanstream assigns one merchant ID number for each processing currency. Include the 9-digit Beanstream ID number here. Additional accounts may also have been issued for special services. Complete one full integration for each of the merchant IDs issued. |
| | trnOrderNumber | Optional but Recommended | 30 alphanumeric (a/n) characters | Include a unique order reference number if desired. If no number is passed, Beanstream will place the default transaction identification number (trnId) in this field. Custom order numbers will be used in duplicate transaction error checking. Order numbers are also required for Server to Server transaction queries. Integrators that wish to use the query function should pass custom values. |
| | trnAmount | R | In the format 0.00. Max 2 decimal places. Max 9 digits total. | This is the total dollar value of the purchase. This should represent the total of all taxes, shipping charges and other product/service costs as applicable. |

| | | | | |
|---|---|---|---|---|
| | errorPage | R | URL (encoded). Max 128 a/n characters. | Not for use with server to server integrations. If a standard transaction request contains errors in billing or credit card information, the customer's browser will be re-directed to this page. Error messages will prompt the user to correct their data. |
| | approvedPage | O | URL (encoded). Unlimited a/n characters. | Beanstream provides default approved or declined transaction pages. For a seamless transaction flow, design unique pages and specify the approved transaction redirection URL here. |
| | declinedPage | O | URL (encoded). Unlimited a/n characters. | Specify the URL for your custom declined transaction notification page here. |
| **INTERAC Online Purchase**<br><br>* By default, billing address information and card owner name are required parameters. This default setting may be changed in the order settings module of the member area. | paymentMethod | R | 2 characters (IO) | Specify paymentMethod=IO to indicate that a transaction is an INTERAC online order. If this value is not passed, the transaction will default to CC for credit card. |
| | ordName | R* | Max 64 a/n characters. | Capture the first and last name of the customer placing the order. |
| | ordEmailAddress | R | Max 64 a/n characters in the format a@b.com. | The email address specified here will be used for sending automated email receipts. |
| | ordPhoneNumber | R* | Min 7 a/n characters<br>Max 32 a/n characters | Collect a customer phone number for order follow-up. |

| | | | | |
|---|---|---|---|---|
| | ordAddress1 | R* | Max 64 a/n characters | Collect a unique street address for billing purposes. |
| | ordAddress2 | O | Max 64 a/n characters | This optional variable is available for longer addresses. |
| | ordCity | R* | Max 32 a/n characters | Indicates the customer's city for billing purposes. |
| | ordProvince | R* | 2 characters | Values must match one of the available province and state codes. |
| | ordPostalCode | R* | 16 a/n characters | Indicates the customer's postal code for billing purposes. |
| | ordCountry | R* | 2 characters | Values must match one of the available ISO country codes. |

## 7.3 Compatible Gateway Options

INTERAC Online purchases can be processed using the following additional gateway features to enhance security or help streamline the transaction process if desired.

| Security Features | Order Customization | Advanced Processing Options |
|---|---|---|
| Hash Validation | Order Comments | None |
| Username/Password | Custom Reference Variables | |
| Inventory Validation | Shipping Details | |
| | Product Details | |

# 8 Pre-Authorizations and Adjustments

The Beanstream Process Transaction API may be used to process purchases, returns, voids, void returns, pre-authorizations and pre-auth completions. By default, the system allows purchase transactions only - the Beanstream member area includes a simple web interface for securely processing returns, voids and other adjustments. However, if merchants wish to also process pre-authorizations and adjustments via API, the option is available.

## 8.1 Pre-authorizations

Pre-authorizations (PA) are often used instead of purchase transactions as a method of reducing the risks associated with credit card processing. When you process a pre-authorization, a temporary hold is placed on the customer card. Merchants can then review customer-submitted data and identify high risk situations before processing the final pre-authorization completion transaction that will appear on a customer card statement. Pre-Authorizations may only be used for credit card transactions.

Be aware that some Visa merchant account providers may pass additional fees for pre-authorizations that are not completed within 72 hours. Contact your merchant account acquirer for details. Beanstream merchant account clients will be subject to additional "misuse of authorization" charges for failing to meet the accepted Visa pre-authorization protocol.

**Card Validation and Cancel Authorization Options**

TD Bank merchant account clients (and some Beanstream merchant account clients) are provided with two additional tools to assist with managing fees related to pre-authorization process.

- For these merchants, the Beanstream system supports $0 pre-authorizations. Merchants may process a $0 pre-authorization to validate a customer card without incurring additional fees or being required to process a separate "completion" transaction.

- These merchants are also provided with a "Cancel Authorization" tool for Visa card type specifically. For any pre-authorization over $1, a "Cancel Authorization" option will be available for a period of 72 hours from the original transaction time. To process a "Cancel Authorization," specify trnType=PAC and trnAmount=0.00 in your transaction request. Merchants are required to use this "Cancel Authorization" option or complete the pre-authorization within the 72 hour window. Cancelled authorizations will appear as $0 pre-authorization completions in Beanstream transaction reports.

If you have Beanstream issued merchant accounts, contact support@beanstream.com for

more information about the availability of these options.

Prior to processing a pre-authorization through the API, you must modify the transaction settings in your Beanstream merchant member area to allow for this transaction type.

- Log in to the Beanstream online member area at www.beanstream.com/admin/sDefault.asp.
- Navigate to *administration* → *account admin* → *order settings* in the left menu.
- Under the heading "Restrict Internet Transaction Processing Types," select either of the last two options. The "Purchases or Pre-Authorization Only" option will allow you to process both types of transaction through your web interface. De-selecting the "Restrict Internet Transaction Processing Types" checkbox will allow you to process all types of transactions including returns, voids and pre-auth completions.

Pre-Authorization request strings are similar to a standard credit card purchase. In addition to the standard fields, a trnType field must be included specifying the value PA for Pre-Authorization. A trnAmount threshold is also imposed on pre-authorizations. This value will vary depending on the merchant account acquirer.

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| trnAmount | R | 0.00 | For Beanstream Canada and TD Visa & MasterCard merchant accounts this value may be $0 or $1 or more. For all other scenarios, this value must be $0.50 or greater. |
| trnType | R | 2 characters (PA) | Specify trnType=PA to process a pre-authorization against a customer's credit card. If omitted, this option will default to P for purchase. |
| In addition to trnAmount and trnType=PA, include all of the variables required for a credit card purchase. | | | |

## Sample Transaction Request

**https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id=123456789&requestType=BACKEND&**trnType=PA**&paymentMethod=CC&trnOrderNumber=1234TEST&trnAmount=5.00&trnCardOwner=Joe+Test&trnCardNumber=4030000010001234&trnExpMonth=10&trnExpYear=10&ordName=Joe+Test&ordAddress1=123+Test+Street&ordCity=Victoria&ordProvince=BC&ordCountry=CA&ordPostalCode=V8T2E7&ordPhoneNumber=5555555555&ordEmailAddress=joe%40testemail.com

### Sample Transaction Response

trnApproved=1&trnId=10001364&messageId=1&messageText=Approved&trnOrderNumber=1234TEST&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=5%2E00&trnDate=7%2F31%2F2009+11%3A57%3A12+AM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=P&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

## 8.2 Adjustments

Prior to processing adjustment transactions through the API, you must modify the transaction settings in your Beanstream merchant member area.

**Step 1: Disable Transaction Restrictions**

Log in to the Beanstream online member area at www.beanstream.com/admin/sDefault.asp.

- Navigate to Administration → Account Admin → Order Settings in the left menu.
- De-select "Restrict Internet Transaction Processing Types"

**Step 2: Activate Enhanced Security:**

- Scroll down the Order Settings page.
- Select "Use username/password validation against transaction." Enter a secure user name and password. Maximum 16 alphanumeric characters per field.
- **OR** enable Hash Validation
- Click "Update" at the bottom of the page

### Pre-Authorization Completions and "Cancel Authorizations"

A Pre-Authorization Completion (PAC) is the second part of a pre-authorization. A PAC has a shorter transaction string than the original authorization as no card or billing information is required. The request must include an adjId variable that identifies the original PA transaction number – it must also include variables for either username/password validation or hash validation. Once you have chosen to use either username/password validation or hash, you must include these options on ALL requests to the Process Transaction API.

**A "Cancel Authorization" option** is also available for those with Beanstream Canada Visa or TD Visa merchant accounts only. This option allows the merchant to reverse a Visa pre-authorization without charging the customer card. Be aware that these merchants are required to process either a pre-auth completion or a cancel authorization within 72 hours of the original Visa pre-authorization transaction. Pre-Authorizations may be cancelled by processing a standard pre-auth completion with trnAmount=0.00 to the Direct Interface API within the allotted time period.

## Returns, Void Purchase, Void Return*

*The INTERAC Online® service supports only purchases and basic returns.

Returns (R), Void Purchases (VP) and Void Returns (VR) all adjust a purchase that has already been processed and approved by the Beanstream system. Voids are used to cancel a transaction before the item is registered against a customer credit card account. Cardholders will never see a voided transaction on their credit card statement. As a result, voids can only be attempted on the same day as the original transaction. After the end of day (roughly 11:59 pm EST/EDT), void requests will be rejected from the API if attempted. Returns may be used to refund a full or partial transaction amount at any time. Return transactions will always appear on a customer statement.

The request strings for these three types of transactions will vary only in the value passed in the trnType field (R=Return, VP=Void Purchase, VR=Void Return). They all require username/password validation (or Hash validation), all require an adjId, and all require a transaction amount. Keep in mind that a void is the removal of the entire amount, while a return will allow you do partial to full refunds of a transaction. The amount sent in needs to reflect this, otherwise it will be rejected from our system.

## Sample Return Request String (Return)

**https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id=123456789&requestType=BACKEND&**trnType**=R&**username**=user1234&**password**=pass1234&trnOrderNumber=1234&trnAmount=1.00&**adjId**=10002115

*The string shown above uses Server to Server integration with Username and Password validation.

## Sample Approved Response String

trnApproved=1&trnId=10002118&messageId=1&messageText=Approved&trnOrderNumber=1234R&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=1%2E00&trnDate=8%2F17%2F2009+1%3A44%3A56+PM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=R&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

## Sample Declined Response (Void)

trnApproved=0&trnId=10002120&**messageId**=205&**messageText**=Transaction+only+voidable+on+the+date+proce

ssed&trnOrderNumber=1234RETURNTEST&authCode=&errorType=N&errorFields=&responseType=T&trnAmount=30%2E45&trnDate=8%2F17%2F2009+2%3A02%3A34+PM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=VP&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

## 8.3 Adjustment Input Variables

Server to Server  Basic HTTP Post

| | Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|---|
| **Basic API Call** | requestType | R | BACKEND | Enter requestType=BACKEND for the recommended server to server integration method. Note that server to server typically cannot be used when hosting forms in the Beanstream Secure Webspace. |
| | merchant_id | R | 9-digits | Beanstream assigns one merchant ID number for each processing currency. Include the 9-digit Beanstream ID number here. Additional accounts may also have been issued for special services. Complete one full integration for each of the merchant IDs issued. |
| | trnOrderNumber | Optional but Recommended | 30 alphanumeric (a/n) characters | Include a unique order reference number if desired. If no number is passed, Beanstream will place the default transaction identification number (trnId) in this field. Custom order numbers will be used in duplicate transaction error checking. Order numbers are also required for Server to Server transaction queries. Integrators that wish to use the query function should pass custom values. |
| | trnAmount | R | In the format 0.00. Max 2 decimal places. Max 9 digits total. | Specify a $ value for the adjustment. Voids must be processed for the full original purchase amount. Refunds may be any value up to the original purchase amount. Beanstream Canada Visa or TD Visa pre-auth completions may be either $0 (for a cancel authorization) or the original pre-authorization amount. Pre-auth completions for all other card types may be any value $0.50 or greater. |

| | | | | |
|---|---|---|---|---|
| | errorPage | R | URL (encoded). Max 128 a/n characters. | Not for use with server to server integrations. If a standard transaction request contains errors in billing or credit card information, the customer's browser will be re-directed to this page. Error messages will prompt the user to correct their data. |
| | approvedPage | O | URL (encoded). Unlimited a/n characters. | Beanstream provides default approved or declined transaction pages. For a seamless transaction flow, design unique pages and specify the approved transaction redirection URL here. |
| | declinedPage | O | URL (encoded). Unlimited a/n characters. | Specify the URL for your custom declined transaction notification page here. |
| **Returns/Voids/PACs** | trnType | R | 2 characters | Indicate the type of transaction to perform.<br>R=Return VR=Void Return<br>V=Void VP=Void Purchase<br>PAC=Pre-Authorization Completion<br><br>If omitted, this field will default to P for purchase. Please note that "R" is the only valid adjustment for INTERAC Online. |
| | adjId | R | 12 digits | Reference the transaction identification number (trnId) from the original purchase. |
| **Adjustments must be performed with either Hash validation or Username & Password validation.** | | | | |

| | | | | |
|---|---|---|---|---|
| **Hash Validation** | hashValue | R | Unlimited a/n characters. | To use this field, activate the hash validation option in the Beanstream Order Settings module. This field value is used as a security measure to ensure that the transaction post to the API has not been modified. The value for hashValue is generated by appending a hash key to the transaction request query string and using a hash algorithm (either MD5 or SHA-1) on the resulting string. |
| | hashExpiry | O | 12 digits | Indicate that a Hashed request has an expiry time. This value must be passed as the current system time in the Pacific time zone (08W00). The system will validate that the request has been received earlier than the date and time value stored in this field. If the session has expired the request will be rejected. The format of the field must be passed as YYYYMMDDHHMM. Example June 18, 2008 2:34 PM would be submitted as 200806181434. |
| | **Variable** | **Required/ Optional** | **Data Type** | **Description** |
| **Username & Password Validation** | username | R | 16 a/n characters | The username passed in this field must match the security settings in the Beanstream Order Settings module. |
| | password | R | 16 a/n characters | The password passed in this field must match the security settings in the Beanstream Order Settings module. |

# 9 Direct Debit/Direct Payment and ACH

Direct Debit/Direct Payment (DD/DP) and Automated Clearing House (ACH) are used to allow payment from one North American bank to another. Unlike INTERAC Online, these are direct bank to bank payments and are not processed in real time. It takes three full business days to process a DD/DP or ACH transaction from end to end. Use this service to:

- Credit a recipient account using funds from your main business bank account (Credit Service)
- Void a credit before 11 am Pacific Time on a scheduled processing date.
- Debit an account and deposit funds into your main business bank account (Debit Service)
- Void a debit before 11 am Pacific Time on a scheduled processing date.

## 9.1 Bank Credit/Debit Transaction Flow

In DD/DP and ACH, only the original transaction request is handled through the API. Final processing occurs offline and is handled by the bank over several business days. Note that customers must sign a pre-authorized debit or credit consent form before you are legally allowed to process transactions using their banking information.

### Step 1: Authorization and Form Post

The customer provides the merchant with full bank account information and provides their authorized consent for a debit or credit. These banking details are POSTed to Beanstream in place of credit card details.

**Sample Canadian Dollar Account Debit Request String**

**https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id=123456789&requestType=BACKEND&trnType=D&username=user1234&password=pass1234&trnOrderNumber=1234&trnAmount=10.00&institutionId=002&transitNumber=12345&accountNumber=12345678910

**Sample US Dollar Account Credit Request String**

**https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id=123456789&requestType=BACKEND&trnType=C&username=user1234&password=pass1234&trnOrderNumber=1234&trnAmount=10.00&routingNumber=123456789&accountNumber=12345678910

### Step 2: File Scheduled and Confirmed

Beanstream checks for errors and schedules the file to start processing on the following business day. A response message is returned to indicate if the item was successful. If transaction details are incorrectly formatted, the merchant is notified in the response string and the item is rejected.

### Step 3: Bank Processing (3 business days)

On the scheduled processing date, the bank receives the file and begins the authorization process. This takes two full business days. Files may be rejected or returned by the bank at any point during this period. The merchant is notified by email and results are logged in the Beanstream reporting module.

### Step 4: Bank Deposit

On the third day after the scheduled processing date, funds are deposited to the recipient account and the results are logged in the Beanstream reporting module.

*A fee is charged for all returned items. Rejected and returned DD/DP and ACH transactions logged in Beanstream's reports.

## 9.2 DD/DP & ACH Input Variables

Server to Server Transactions  Basic HTTP Post

| | Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|---|
| **Basic API Call** | requestType | R | BACKEND | Enter requestType=BACKEND for the recommended server to server integration method. Note that server to server typically cannot be used when hosting forms in the Beanstream Secure Webspace. |
| | merchant_id | R | 9-digits | Beanstream assigns one merchant ID number for each processing currency. Include the 9-digit Beanstream ID number here. Additional accounts may also have been issued for special services. Complete one full integration for each of the merchant IDs issued. |
| | trnOrderNumber | Optional but Recommended | 30 alphanumeric (a/n) characters | Include a unique order reference number if desired. If no number is passed, Beanstream will place the default transaction identification number (trnId) in this field. Custom order numbers will be used in duplicate transaction error checking. Order numbers are also required for Server to Server transaction queries. Integrators that wish to use the query function should pass custom values. |
| | trnAmount | R | In the format 0.00. Max 2 decimal places. Max 9 digits total. | This is the total dollar value of the purchase. This should represent the total of all taxes, shipping charges and other product/service costs as applicable. |

| | | | | |
|---|---|---|---|---|
| | errorPage | R | URL (encoded). Max 128 a/n characters. | Not for use with server to server integrations. If a standard transaction request contains errors in billing or credit card information, the customer's browser will be re-directed to this page. Error messages will prompt the user to correct their data. |
| | approvedPage | O | URL (encoded). Unlimited a/n characters. | Beanstream provides default approved or declined transaction pages. For a seamless transaction flow, design unique pages and specify the approved transaction redirection URL here. |
| | declinedPage | O | URL (encoded). Unlimited a/n characters. | Specify the URL for your custom declined transaction notification page here. |
| **DD/DP (EBP) & ACH** | trnType | R | 1-2 characters | Specify one of the four following options:<br>D=Debit an outside bank account (receive money in your own account)<br>C=Credit an outside bank account (pay from your own account)<br>VD=Void Debit<br>VC=Void Credit |
| | institutionId | Required for D & C type transactions (Canada) | 3 digits | This is the three digit financial institution number associated with Canadian bank accounts. Specify the number associated with the account to be debited (trnType=D) or credited (trnType=C). |
| | transitNumber | Required for D & C type transactions (Canada)) | 5 digits | This is the five-digit transit number associated with Canadian bank accounts. Specify the number associated with the account to be debited (trnType=D) or credited (trnType=C). |

| | | | | |
|---|---|---|---|---|
| | routingNumber | Required for D & C type transactions (US Only) | Max 9 digits | Routing numbers for US bank accounts may be nine digits or less. Specify the routing number for the account to be debited (trnType=D) or credited (trnType=C). |
| | accountNumber | Required for D & C type transactions (all locations) | Max 12 digits | Account numbers may vary in length up to 12 digits. Specify the routing number for the account to be debited (trnType=D) or credited (trnType=C). |
| | ordName | R for D & C type transactions | Max 64 a/n characters | Capture the first and last name of the customer placing the order. |
| **Direct Debit/Direct Payment and ACH must be performed with either Hash validation or Username & Password validation.** | | | | |
| **Hash Validation** | hashValue | R | Unlimited a/n characters. | To use this field, activate the hash validation option in the Beanstream Order Settings module. This field value is used as a security measure to ensure that the transaction post to the API has not been modified. The value for hashValue is generated by appending a hash key to the transaction request query string and using a hash algorithm (either MD5 or SHA-1) on the resulting string. |
| | hashExpiry | O | 12 digits | Indicate that a Hashed request has an expiry time. This value must be passed as the current system time in the Pacific time zone (08W00). The system will validate that the request has been received earlier than the date and time value stored in this field. If the session has expired the request will be rejected. The format of the field must be passed as YYYYMMDDHHMM. Example June 18, 2008 2:34 PM would be submitted as 200806181434. |

| | | | | |
|---|---|---|---|---|
| **Username & Password Validation** | username | R | 16 a/n characters | The username passed in this field must match the security settings in the Beanstream Order Settings module. |
| | password | R | 16 a/n characters | The password passed in this field must match the security settings in the Beanstream Order Settings module. |

## 9.3 Compatible Gateway Options

| Transaction Types | Security Features | Billing Options |
|---|---|---|
| Bank Debits | Hash Validation | Batch Processing |
| Bank Credits | Username/Password | |
| Void Debits and Credits | Inventory Validation | |

# 10 Additional Order Information

## 10.1 Shipping Details

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| shipName | O | Maximum 64 alphanumeric (a/n) characters | Specify a unique shipping name. |
| shipEmailAddress | O | Maximum 64 a/n characters in the format a@b.com | The shipping email address may be collected for follow-up purposes. Automated email receipts will not be sent to this address. |
| shipPhoneNumber | O | Maximum 32 a/n characters | Collect a phone number specific to the shipping contact. |
| shipAddress1 | O | Maximum 64 a/n characters | Collect a unique street address for shipping purposes. |
| shipAddress2 | O | Maximum 64 a/n characters | Additional shipping address field available for long addresses |
| shipCity | O | Maximum 32 a/n characters | Indicates the customer's city for shipping purposes. |
| shipProvince | O | 2 characters | Values passed in this field must match the available province/state codes. |
| shipPostalCode | O | 16 a/n characters | Indicates the customer's postal code for shipping purposes. |
| shipCountry | O | 2 characters | Values passed in this field must match the available ISO country codes. |
| shippingMethod | O | Maximum 64 a/n characters | Include a description of the shipping method to be used for the order. |

| | | | |
|---|---|---|---|
| deliveryEstimate | O | Maximum 9 digits | Specify an estimated delivery time in days. |
| shippingRequired | O | 1 digit | When set to "1", customers must enter all shipping fields to submit their order. |
| shipSameAsOrd | O | 1 digit | When set to "1", all shipping address fields will be auto-populated with the customer's billing information. |

## 10.2 Product Details

Product and pricing variables are used for reporting purposes and have no affect on the dollar amount charged to the card holder. Pass this information to include product details on the customer email receipt (%productInfo% must be included on the receipt template) and to store product related information in the Beanstream Transaction Report. If you are using Beanstream's inventory module and these parameters are passed, items will be added to the inventory if they do not already exist. Items marked with an asterisk (*) are also used in conjunction with Inventory Validation.

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| prod_id_n * | Required with Inventory validation, otherwise optional. | Maximum 32 alphanumeric characters | Indicates the product ID or SKU number used to uniquely identify a product. There is no limit to the number of product fields that may be used. All field names must be numbered starting from 1. Fields must be numbered from 1-n. Replace the "n" with the numbered reference (ie. prod_id_1, prod_id_2). |
| prod_name_n | O | Maximum 64 a/n characters | Captures a unique product description for each item in an order. Replace the "n" with numbered reference (ie. prod_name_1, prod_name_2). |
| prod_quantity_n * | Required with Inventory validation. | Maximum 9 digits | Captures the quantity of each item in an order. Fields must be numbered from 1-n. Replace the "n" with number reference (ie. prod_quantity_1, prod_quantity_2). |
| prod_shipping_n | O | Maximum 9 digits in the format 0.00 | Indicates the cost of shipping on a per product basis. Fields must be numbered from 1-n. Replace the "n" with a numbered reference (ie. prod_shipping_1, prod_shipping_2). |
| prod_cost_n | O | Maximum 9 | Specifies the per item cost in an order. |

| | | | |
|---|---|---|---|
| | | digits in the format 0.00 | Fields must be numbered from 1-n. Replace the "n" with number reference (ie. prod_cost_1, prod_cost_2). |
| ordItemPrice * | Required with Inventory validation. | Maximum 9 digits in the format 0.00 | The total price of all items in the order, taking into account product quantities. |
| ordTax1Price * | Required with Inventory validation. | Maximum 9 digits in the format 0.00 | Use this variable to record the total amount of the primary tax (ex: GST) applied to the order. This is for record keeping and/or inventory validation purposes only. This amount will NOT be added to trnAmount. |
| ordTax2Price * | Required with Inventory validation. | Maximum 9 digits in the format 0.00 | Use this variable to record the total amount of the secondary tax (ex: PST) applied to the order. This is for record keeping and/or inventory validation purposes only. This amount will NOT be added to trnAmount. |
| ordShippingPrice* | Required with Inventory validation. | Maximum 9 digits in the format 0.00 | The total of all shipping charges |

## 10.3 Language Details

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| trnLanguage | O | 3 digit ISO code | This value is used to trigger English or French customer email receipts. This value is passed back to the approval/decline page untouched. Valid values are FRE or ENG. |

## 10.4 Custom Data

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| refn | O | 256 alphanumeric characters | Capture custom order information with up to five reference fields. Details are sent back in the response string untouched. Data is also stored in Beanstream's database and is available |

| | | | |
|---|---|---|---|
| | | | through report downloads and APIs. Replace "n" with a value from 1 to 5 (ie: ref1, ref2, ref3, ref4 and ref5). If you are using Beanstream's Dynamic DBA service, one of ref1-ref4 may already be allocated for passing custom transaction identifiers - contact Beanstream Support for confirmation. |
| trnComments | O | Maximum 8000 alphanumeric characters | Include an optional comment with each order. Comments will be displayed in online reports but are not available through downloads or reporting APIs. |
| customerIP | O | Standard IP format | Pass the customer's IP address with the order request. In Server to Server type integrations, the IP address of the transaction source will reflect the IP of the merchant's server UNLESS this variable is included in the transaction request. When included, customerIP will be used in transaction validation tools such as IP filtering and Risk Scoring. This variable must be used in conjunction with requestType=backend and either username/password or HASH validation. |

# 11 Processing with Payment Profiles

The Secure Payment Profiles is an additional paid service that allows merchants to create secure payment accounts for their customers. Ensure that Beanstream has enabled this option on your account before integrating.

With this tool, merchants can process transactions against customer profiles that reside on Beanstream's secure servers. As all information is stored by Beanstream, merchants avoid retaining confidential information such as contact and credit card details on their own systems. Repeat shoppers are also not required to re-enter payment information with each purchase. By integrating the Secure Payment Profiles system via API, merchants can ensure that customers are not transferred offsite during the purchase process.

Secure Payment Profiles uses two types of API calls. Profile creation or modification requests are sent to a dedicated service URL at https://www.beanstream.com/scripts/payment_profile.asp. For details on performing profile creation or modification requests, review our Secure Payment Profiles Guide. Once a profile has been created, transactions may be processed against the customer account using the Process Transaction API. During the account creation process, each profile is assigned a unique customer code (customerCode). When processing a transaction using the Process Transaction API, this customerCode variable is passed in lieu of standard billing and payment information. Payment profiles may be used for Credit Card processing or DD/DP & ACH processing only. This service cannot be used with the INTERAC Online service.

| Transaction Types | customerCode Replaces |
|---|---|
| Credit Card | trnCardOwner, trnCardNumber, trnExpMonth, trnExpYear |
| DD/DP (Canadian Bank Transactions) | institutionId, transitNumber, accountNumber |
| ACH (US Bank Transactions) | institutionId, transitNumber, accountNumber |
| If billing address information has been stored your Secure Payment Profiles then the *customerCode* parameter will also be used in place of the following billing address request parameters: *ordName, ordEmailAddress, ordPhoneNumber, ordAddress1, ordAddress2, ordCity, ordProvince, ordPostalCode,* and *ordCountry.* | |
| Additional Integration Requirements | |
| Secure Payment Profile transaction requests must be performed with either Hash validation or username/password validation. | |

## Sample Request String

**https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id=123456789&requestType=BACKEND&**trnType=P**&&trnOrderNumber=1234TEST&trnAmount=5.00&**customerCode**=6tw1c4p438TA9P0jU8A

## Sample Response String

trnApproved=1&trnId=12345678&messageId=1&messageText=Approved&trnOrderNumber=1234TEST&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=0%2E50&trnDate=7%2F31%2F2009+3%3A13%3A52+PM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=PAC&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

# 12 Recurring Billing

Recurring billing allows merchants to set automated billing schedules for customers. Credit card information is collected a single time and stored on Beanstream's secure servers. Recurring billing accounts can be created manually through the Beanstream member area or through the Process Transaction API. In order to modify, disable or delete an existing account, a special request must be sent to the dedicated Recurring Billing API. For a complete description of these two operations, please refer to the Recurring Billing documentation.

Beanstream is phasing out the old design of our recurring billing to offer a new.

# 13 Transaction Queries

Transaction queries can be used to retrieve transaction responses in Server to Server integrations. Queries are typically used in cases where a transaction request has been submitted to the Beanstream system, however latency issues or a dropped connection has stopped the merchant's processing script from receiving the API response. The orderNumber field is a required field for transaction queries; therefore, developers wishing to use the query function must pass this value with the transaction string. When a query request is received, Beanstream will attempt to locate the last processed transaction with a matching amount, card owner name, card number, expiry date, and order number. Transactions that are considered duplicate will not be included. If multiple matches are found or if no matching data is retrieved, an error message will be returned.

To process a query, pass the following required parameters

- requestType=BACKEND
- trnType=Q
- merchant_id=*merchant's 9-digit Beanstream account id*
- trnOrderNumber=*unique order id number for the transaction being queried*

Additional optional values include:

- trnAmount
- trnCardOwner
- trnCardNumber
- trnExpMonth
- trnExpYear
- customerCode (for Payment Profile integrations only)

## Sample Request String

https://www.beanstream.com/scripts/process_transaction.asp?**merchant_id**=123456789&**requestType**=BACKEND&**trnType**=Q&username-user1234&password=pass1234&**trnOrderNumber**=12322

## Sample Response String

trnApproved=1&trnId=100021208&messageId=1&messageText=Approved&trnOrderNumber=12322R&authCode=TEST&errorType=N&errorFields=&responseType=T&trnAmount=1%2E00&trnDate=8%2F17%2F2009+1%3A44%3A56+PM&avsProcessed=0&avsId=0&avsResult=0&avsAddrMatch=0&avsPostalMatch=0&avsMessage=Address+Verification+not+performed+for+this+transaction%2E&cardType=VI&trnType=R&paymentMethod=CC&ref1=&ref2=&ref3=&ref4=&ref5=

# 14 Enabling API Security Features

## 14.1 Require CVD Numbers

By requiring CVD numbers, all credit card payments must be submitted with the 3 or 4-digit CVD (or CVV) code from the back of the purchaser's card. This security tool helps to ensure that customers have a card in hand at the time of purchase.

### To make CVD a required field:

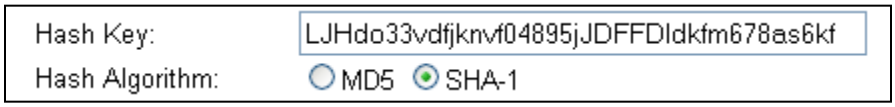
- Log in to the Beanstream online member area at www.beanstream.com/admin/sDefault.asp.
- Navigate to Administration → Account Settings → Order Settings in the left menu.
- Select "Require CVD number for credit card transactions."
- Include the trnCardCvd variable in all purchase requests.

## 14.2 Hash Validation

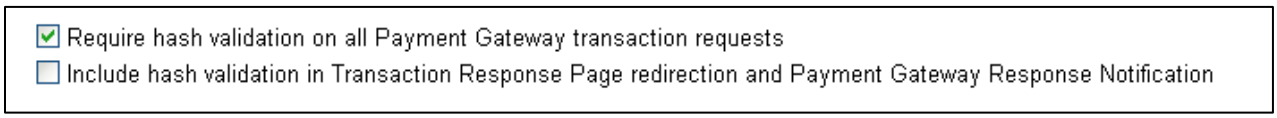
Hash validation is used to help protect the integrity of API transaction requests. Online Mart supports MD5 or SHA-1 hash encryption. Once you have enabled this option, you will have to use it on every single transaction you submit to the API. Hash Validation may not be used in conjunction with Username and Password validation.

## 14.3 To enable Hash Validation:

1) Navigate to administration > account settings > order settings in the left menu of your account. Enter a Hash Key in the field provided. On the same page, pick the Hash Algorithm that you want to use (MD5 or SHA-1).




2) Activate Hash Validation by selecting one of the two hash checkbox options (all Payment Gateway transaction requests or Transaction Response Page redirection). Use the first option for Server to Server integrations. Use the second option for Hosted Payment Page and other HTTP Post integrations.




3) Take the payment form or API request string (without the URL) and place the hash key in the string where you want the hash to be generated. This may be at the end of the string or after any complete variable within the string.

variable1=aaa&variable2=bbb&variable3=cccLJHdo33vdfjknvf04895jJDFFDldkfm678as6kf&variable4=ddd&variable5=eee

4) Generate a hash of the string up until the end of the hash key only. Use MD5 or SHA-1 encryption to match your selection in your Online Mart member account. A quick Google search will return a list of many free MD5 or SHA-1 hash generator tools if you do not already have one at hand or you may refer to our sample code.

5) Include your results in your string by placing a hashValue variable in the same location as you placed your hash key. Send this string to the Payment Form or Process Transaction API.

https://www.serviceURL.com/sample/asp?variable1=aaa&variable2=bbb&variable3=ccc&hashValue=f8468732a3c857acdb36cd631d0d1391&variable4=ddd&variable5=eee

## 14.4 Username/Password Validation

A unique API username and password may be passed with each transaction request string. When enabled, values passed in the username and password variables must match settings stored in Beanstream's member area in order for a transaction to be processed. Username and password validation may not be used with hash validation . Once you have enabled this option, you will have to use it on every single transaction you submit to the API.

### To enable username and password validation:

- Log in to the Beanstream online member area at www.beanstream.com/admin/sDefault.asp.
- Navigate to Administration → Account Admin → Order Settings in the left menu.
- Select "Use username/password validation against transaction.
- Enter a secure user name and password. Maximum 16 alphanumeric characters per field.
- Click "Update" at the bottom of the page to save your changes.

## 14.5 Validate Referring Host

Use this option to ensure that transactions originate only from a designated referring host. Integrators may specify a valid host in the Beanstream member area. If a transaction is submitted with a different host name, the transaction request will be automatically rejected before being sent to the bank for processing. Once enabled, this setting will apply to all transactions processed through the Beanstream system.

### To activate referring host validation:

- Log in to the Beanstream online member area at www.beanstream.com/admin/sDefault.asp.
- Navigate to Administration → Account Admin → Order Settings in the left menu.

- Scroll down the Order Settings page. Select the "Validate referring host address" checkbox.
- In the field provided, enter the domain of the payment page(s) that will be submitting transactions.

## 14.6 Inventory Validation

Inventory validation is used to verify that customer-submitted order information matches product inventory data stored in the Beanstream member area. Merchants must have items stored in the Beanstream inventory module in order to use this feature.

### To activate Beanstream inventory validation:

- Log in to the Beanstream member area at www.beanstream.com/admin/sDefault.asp.
- Navigate to Administration → Account Settings → Order Settings.
- On the Order Settings page, select the checkbox marked "Validate orders against inventory."

When inventory validation is activated, the following product fields MUST be passed with the transaction request:

| | | |
|---|---|---|
| prod_id_n | ordItemPrice | ordTax1Price |
| prod_quantity_n | ordShippingPrice | ordTax2Price |

## 14.7 Canadian Address Verification

Beanstream offers a value-added Canadian Address Verification Service which merchants may subscribe to. If you have signed up for this service, review our CAV documentation for detailed integration instructions. The following CAV service variables are available for passing with transaction requests.

| Variable | Required/ Optional | Data Type | Description |
|---|---|---|---|
| cavServiceVersion | Optional but recommended | 3 digits. | Specify version 1.3 for the latest Equifax messaging with up to four address and card validation responses and a separate date of birth response code. If no value is passed, version 1.0 will be used by default. Version 1.0 provides only a single address and credit card validation response. Review our CAV Integration guide for details. |

| | | | |
|---|---|---|---|
| cavEnabled | O | 1 digit | Use this variable to enable or disable CAV authentication on specific transactions. "Require CAV on all transactions" must be turned OFF in the Beanstream CAV module.<br>1 =enabled<br>0=disabled (default value) |
| cavPassCode | O | 32 alphanumeric characters | If you have specified an access passcode through the CAV module in the Beanstream member area, include your passcode here. Transactions that are attempted with a missing or mis-matched passcode will be rejected. |

## 14.8 PGP Signing and Encryption

PGP allows you to create a public key to sign and verify transaction data. PGP signing will take your public key and transaction data and generate a hash value. This hash value is wrapped around your transaction request when submitted to Beanstream for processing. If either the transaction data or hashed signature is modified, Beanstream will not be able to identify the signature and will decline the request. When activated, Beanstream will also sign all transaction responses using the Beanstream public key. You must validate the signature of all responses against the Beanstream public key in order to ensure that an approval/decline response has originated from the Beanstream transaction server. You may download the Beanstream public key from the following URL: https://www.beanstream.com/support/pgp/beanstream.asc.

### To activate PGP

- Go to Administration → Account Settings → Order Settings on the Beanstream membership page. Under "Transaction validation options," select the box called "Require PGP signing of all transactions."
- Enter your public key information in the field provided.
- Choose to enable PGP Encryption if an extra layer of security is required.
- Click "Update."

All transaction requests submitted through your account will now be rejected unless they have been correctly signed with a matching PGP Public key and PGP Key Id.

### Sample Encrypt Transaction Function (ASP)

The following script references NSDPGP.DLL v3.20 PGP functions.

```
<%
function EncryptTransactionString(trnString, passPhrase, signKeyId)

dim merchantId
dim decryptedFile
dim encryptedFile
dim beanstreamKeyId
dim appPath
dim trnString
dim objPgp
dim fs
dim f

beanstreamKeyId = "0x38180389"

'Collect the merchant id from the passed form data
merchantId = request("merchant_id")

'Create an instance of NSDPGP.DLL(v3.20) COM Interface to PGP 6.5.2
set objPgp = CreateObject("NSDPGP")

Set fs = CreateObject("Scripting.FileSystemObject")

'Get the tempory folder specified by the server's TMP environment variable
appPath = fs.GetSpecialFolder(2)

'Build tempory file names
decryptedFile = appPath & "\" & fs.GetTempName
encryptedFile = appPath & "\" & fs.GetTempName

'If decryptedFile already exists, delete it first to avoid errors when creating the file.
if fs.FileExists(decryptedFile) = true then fs.deleteFile decryptedFile

'Write the passed order form transaction string to the decrypted file
set f = fs.OpenTextFile(decryptedFile, 8, true, -2)
   f.writeline trnString
f.close

objPgp.EncryptFileEx beanstreamKeyId, signKeyId, decryptedFile, encryptedFile, passPhrase

'Read in the encrypted transaction string
set f = fs.OpenTextFile(encryptedFile)
   trnString = f.readall
f.close
```

```
'Remove the tempory files
objPgp.WipeFile(decryptedFile)
objPgp.WipeFile(encryptedFile)

set fs = nothing

EncryptTransactionString = trnString

end function
%>
```

## Sample Sign Transaction Function (ASP)

The following script references NSDPGP.DLL v3.20 PGP functions.

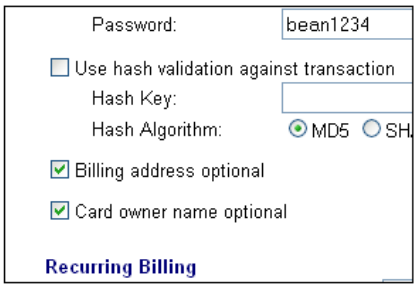
```
<%
function SignTransactionString(trnString, passPhrase, signKeyId)
   'Purpose: To sign a transaction with the merchants PGP key for submission to
'       the Beanstream Transaction Server.
'Pre:    PGP Security Suite has been installed on the web server that this
'       script is executing on. A PGP Key has been generated for use in
'       submitting transactions to Benastream. The NSDPGP COM object has
'       been installed and registered on the Web Server that this script
'       is executing on. The TMP Environment variable has been declaired
'       on the web server and points to a folder with write permissions.
   '       trnString contains all required transaction parameters to be
'       passed to the Beanstream Transaction Server.
   'Post:    None.
   'Returns: The signed trnString is returned to the calling application.

   dim unsignedFile
   dim signedFile
   dim appPath
   dim objPgp
   dim fs
   dim f

   'Create an instance of NSDPGP.DLL(v3.20) COM Interface to PGP 6.5.2
   set objPgp = CreateObject("NSDPGP")

   Set fs = CreateObject("Scripting.FileSystemObject")

   'Get the tempory folder specified by the server's TMP environment variable
   appPath = fs.GetSpecialFolder(2)
```

```
    'Build tempory file names
    unsignedFile = appPath & "\" & fs.GetTempName
    signedFile = appPath & "\" & fs.GetTempName

    'If unsignedFile already exists, delete it first to avoid errors when creating.
    if fs.FileExists(unsignedFile) = true then fs.deleteFile unsignedFile

    'Write the passed order form transaction string to the decrypted file
    set f = fs.OpenTextFile(unsignedFile, 8, true, -2)
            f.writeline trnString
    f.close

    'Sign the order form transaction string
    call objPgp.SignFile (2, signKeyId, unsignedFile, signedFile, passPhrase)

    'Read in the signed transaction string
    set f = fs.OpenTextFile(signedFile)
            SignTransactionString = f.readall
    f.close

    'Remove the tempory files
    objPgp.WipeFile(unsignedFile)
    objPgp.WipeFile(signedFile)

    set fs = nothing

end function
%>
```

# 15 Make Billing Address/Card Owner Name Optional

With default settings, Card Owner Name and all Billing Address fields are required with each transaction request. Merchants may update the Order Settings area to make these fields optional, eliminating the need to pass complete customer contact information with each order.

- To make card owner name and billing address information optional:
- Navigate to administration > order settings in the left menu of the member area.
- Scroll to the bottom of the page
- Select the Billing address optional or Card owner name optional checkboxes as required.



# 16 Table of Process Transaction Auth Input Variables

The Process Transaction Auth API is used for VBV and INTERAC Online transactions to return bank issued response messaging to Beanstream in order to complete the transaction process. The Process Transaction Auth service URL is https://www.beanstream.com/scripts/process_transaction.asp. The following bank values must be posted to this URL:

| | Variable |
|---|---|
| **INTERAC Online** | funded |
| | IDEBIT_TRACK2 |
| | IDEBIT_VERSION |
| | IDEBIT_MERCHANTDATA |
| | IDEBIT_INVOICE |
| | IDEBIT_NOTFUNDEDURL |
| | IDEBIT_ISSLANG |
| | IDEBIT_INVOICE |
| | IDEBIT_NOTFUNDEDURL |
| | IDEBIT_ISSLANG |

| | Variable |
|---|---|
| **Verified by Visa & SecureCode** | paRES |
| | MD |

# 17 Table of Beanstream Response Variables

Server to Server integrations only  Basic HTTP POST integrations only  **Red Text** Display value to customer

| Field Name | Type | Description |
|---|---|---|
| authCode | 0-32 alphanumeric characters | If the transaction is approved this parameter will contain a unique bank-issued code. |
| avsAddrMatch | 1 digit | 1 = Address match. The ordAddress1 parameter matches the address on file at the issuing bank.<br>0= Address mismatch. The address submitted with the order does not match information on file at the issuing bank. |
| avsId | 1 digit | An ID number referencing a specific AVS response message. Review Appendix A for details. |
| avsMessage | 128 a/n characters | This is a descriptive text message associated with the avsId response code. |
| avsPostalMatch | 1 digit | 1 if the ordPostalCode parameter matches the consumers address records at the issuing bank.<br>0 if the ordPostalCode parameter does not match the customer's address records or if AVS was not processed for the transaction. |
| avsProcessed | 1 digit | 1 if the issuing bank has successfully processed an AVS check on the transaction.<br>0 if no AVS check has been performed. |
| avsResult | 1 digit | 1 if AVS has been validated with both a match against address and a match against postal/ZIP code. |
| cvdId | 1 digit | 1=CVD Match<br>2=CVD Mismatch<br>3=CVD Not Verified<br>4=CVD Should have been present<br>5=CVD Issuer unable to process request<br>6=CVD Not Provided |

| | | |
|---|---|---|
| eci | 1 digit | For credit card transactions where Verified by Visa (VBV) and/or MasterCard SecureCode (MCSC) services have been integrated, the ECI code indicates if the chargeback liability has shifted away from the merchant to the card issuing credit card company because of VBV or MCSC verification. If VBV or MCSC have been enabled then this parameter will be returned for all transactions and integration types.<br><br>5 = Authentication successful. Liability shift occurs.<br>6 = Authentication attempted but the cardholder is not a program participant. Liability shift occurs.<br>7 = No VBV or SecureCode authentication attempted. No liability shift occurs. |
| errorFields | List of fields | In the case of a user generated error, this variable will include a list of fields that failed form validation. You will wish to notify the customer that they must correct these fields before the transaction can be completed. |
| errorType | 1 character | This field will return the value N, S, or U. |
| ioConfCode | 15 a/n characters | Where applicable, an INTERAC Online confirmation number will be returned by the customer's financial institution if the transaction has been processed successfully. This value must be displayed to the customer on a transaction confirmation page for INTERAC Transactions. |
| ioInstName | 30 a/n characters | The name of the customer's financial institution for INTERAC Online transactions. |
| messageId | 1-3 digits | The message id references a detailed approved/declined transaction response message. Review our gateway response message table for a full description of each message. |
| messageText | A | This field will return a basic approved/declined message which may be displayed to the customer on a confirmation page. Review our gateway response message table for details. |
| paymentMethod | 2 a/n characters | IO=INTERAC Online transaction<br>CC=Credit Card transaction |

| | | |
|---|---|---|
| rbAccountId | 10 digits | The identification number of the recurring billing profile. This value is only returned upon creation of the account. During a regular recurring transaction, the value returned will be "billingId". For complete details on response messaging for regular, recurring transactions, consult our Recurring Billing documentation. |
| ref1 | 256 a/n characters | The value of the ref1 field submitted in the transaction request. |
| ref2 | 256 a/n characters | The value of the ref2 field submitted in the transaction request. |
| ref3 | 256 a/n characters | The value of the ref3 field submitted in the transaction request. |
| ref4 | 256 a/n characters | The value of the ref4 field submitted in the transaction request. |
| ref5 | 256 a/n characters | The value of the ref5 field submitted in the transaction request. |
| responseType | 1 character | Set to the value of 'T' to indicate a transaction completion response. If VBV is enabled on the merchant account a value of 'R' may be returned to indicate a VBV redirection response. |
| rspCavResult | 1 digit | 1=transaction passed validation<br>0=address validation failed |
| rspCodeAddr*n* | 3 digits | One of several dedicated address-related CAV messages may be returned in this field. Use this information to understand the level of match that was obtained. Up to four address codes will be returned for each CAV item. (respCodeAddr1, rspCodeAddr2, etc) |
| rspCodeCav | 3 digits | If CAV service is enabled with service version 1.0, a single Equifax reswponse message will be returned here. |
| rspCodeCredit*n* | 3 digits | Once of several dedicated quickmatch responses pertaining to credit card information. Up to four credit card related responses may be returned for each CAV item (rspCodeCredit1, rspCodeCredit2, etc). |

| | | |
|---|---|---|
| rspCodeDob | 4 digits | A Date of Birth match response code. This information will only be returned when cavDOB was passed with the address verification/transaction request. |
| rspCodeSafeScan | 1 character | 1 to 20 detailed SafeScan codes will be returned in this parameter. Requires service subscription. In SafeScan versions 1.1 and higher, multiple codes are appended with a separator. |
| rspCodeSafeScanId | 1 character | 1 to 20 SafeScanID codes may be returned. Requires service subscription. In SafeScan ID version 1.1 and higher, multiple values are returned with a separator. |
| rspCustomerDec | alphanumeric | Provides information specific to any consumer declaration recorded on the consumer's credit file. |
| trnAmount | 9 digits | The amount of the transaction. |
| trnApproved | 1 digits | 0 = Transaction refused, 1 = Transaction approved |
| trnCustomerName | 32 a/n characters | The customer name as submitted with the transaction request. |
| trnDate | 20 a/n characters | The date and time that the transaction was processed. |
| trnEmailAddress | 64 a/n characters | The customer email address as submitted with the transaction request. |
| trnId | 8 digits | Unique id number used to identify an individual transaction. |
| trnLanguage | 3 characters | The language of correspondence as submitted with the transaction request. |
| trnOrderNumber | 1-30 a/n characters | The value of trnOrderNumber submitted in the transaction request. |
| trnPhoneNumber | 32 digits | The customer phone number as submitted with the transaction request. |
| trnType | 3 a/n characters | The original value sent to indicate the type of transaction to perform (i.e. P,R,VP,VR, PA, PAC, Q). |

| | | |
|---|---|---|
| cardType | 2A | The type of card used in the transaction.<br>VI=Visa, MC=MasterCard, AM=American Express NN=Discover, DI=Diners, JB=JCB, IO=INTERAC Online, ET=Direct Debit/Direct Payments/ACH |

# Appendix A: Reference Codes

## Country Codes

| ID | Name | ID | Name | ID | Name |
|---|---|---|---|---|---|
| AF | Afghanistan | GE | Georgia | MP | Northern Mariana Islands |
| AR | Argentina | DE | Germany | NO | Norway |
| AX | Åland Islands | GH | Ghana | OM | Oman |
| AL | Albania | GI | Gibraltar | PK | Pakistan |
| DZ | Algeria | GB | Great Britain | PW | Palau |
| AS | American Samoa | GR | Greece | PS | Palestinian Territory, Occupied |
| AD | Andorra | GL | Greenland | PA | Panama |
| AO | Angola | GD | Grenada | PG | Papua New Guinea |
| AI | Anguilla | GP | Guadeloupe | PY | Paraguay |
| AQ | Antarctica | GU | Guam | PE | Peru |
| AG | Antigua and Barbuda | GT | Guatemala | PH | Philippines |
| AM | Armenia | GN | Guinea | PN | Pitcairn |
| AW | Aruba | GW | Guinea Bissau | PL | Poland |
| AU | Australia | GY | Guyana | PT | Portugal |
| AT | Austria | HT | Haiti | PR | Puerto Rico |
| AZ | Azerbaijan | HM | Heard and McDonald Islands | QA | Qatar |
| BS | Bahamas | HN | Honduras | RE | Reunion |
| BH | Bahrain | HK | Hong Kong | RO | Romania |
| BD | Bangladesh | HU | Hungary | RU | Russian Federation |
| BB | Barbados | IS | Iceland | RW | Rwanda |
| BY | Belarus | IN | India | KN | Saint Kitts and Nevis |
| BE | Belgium | ID | Indonesia | LC | Saint Lucia |
| BZ | Belize | IR | Iran, Islamic Republic of | VC | Saint Vincent and the Grenadines |
| BJ | Benin | IQ | Iraq | WS | Samoa |
| BM | Bermuda | IE | Ireland | SM | San Marino |
| BT | Bhutan | IL | Israel | ST | Sao Tome and Principe |
| BO | Bolivia | IT | Italy | SA | Saudi Arabia |
| BA | Bosnia and Herzegovina | JM | Jamaica | SN | Senegal |
| BW | Botswana | JP | Japan | CS | Serbia and Montenegro |
| BV | Bouvet Island | JO | Jordan | SC | Seychelles |

| ID | Name | ID | Name | ID | Name |
|---|---|---|---|---|---|
| BR | Brazil | KZ | Kazakhstan | SL | Sierra Leone |
| IO | British Indian Ocean Territory | KE | Kenya | SG | Singapore |
| BN | Brunei Darussalam | KI | Kiribati | SK | Slovakia |
| BG | Bulgaria | KP | Korea, Democratic People's Republic | SI | Slovenia |
| BF | Burkina Faso | KR | Korea, Republic of | SB | Solomon Islands |
| BI | Burundi | KW | Kuwait | SO | Somalia |
| KH | Cambodia | KG | Kyrgyzstan | ZA | South Africa |
| CM | Cameroon | LA | Lao People's Democratic Republic | GS | South Georgia – South Sandwich Islands |
| CA | Canada | LV | Latvia | ES | Spain |
| CV | Cape Verde | LB | Lebanon | LK | Sri Lanka |
| KY | Cayman Islands | LI | Liechtenstein | SH | St. Helena |
| CF | Central African Republic | LS | Lesotho | PM | St. Pierre and Miquelon |
| TD | Chad | LR | Liberia | SD | Sudan |
| CL | Chile | LY | Libyan Arab Jamahiriya | SR | Suriname |
| CN | China | LT | Lithuania | SJ | Svalbard and Jan Mayen |
| CX | Christmas Island | LU | Luxembourg | SZ | Swaziland |
| CC | Cocos (Keeling) Islands | MO | Macau | SE | Sweden |
| CO | Columbia | MK | Macedonia, Former Yugoslav Republic of | CH | Switzerland |
| KM | Comoros | MG | Madagascar | SY | Syrian Arab Republic |
| CG | Congo | MW | Malawi | TW | Taiwan |
| CD | Congo, The Democratic Republic of the | MY | Malaysia | TJ | Tajikistan |
| CK | Cook Islands | MV | Maldives | TZ | Tanzania, United Republic of |
| CR | Costa Rica | ML | Mali | TH | Thailand |
| CI | Cote d'Ivoire – Really Ivory Coast | MT | Malta | TG | Togo |
| HR | Croatia | MH | Marshall Islands | TK | Tokelau |
| CU | Cuba | MQ | Martinique | TO | Tonga |
| CY | Cyprus | MR | Mauritania | TT | Trinidad and Tobago |
| CZ | Czech Republic | MU | Mauritius | TN | Tunisia |
| DK | Denmark | YT | Mayotte | TR | Turkey |
| DJ | Djibouti | MX | Mexico | TM | Turkmenistan |
| DM | Dominica | FM | Micronesia, Federated States of | TC | Turks and Caicos Islands |
| DO | Dominican Republic | MD | Moldova, Republic of | TV | Tuvalu |
| TL | East Timor | MC | Monaco | UG | Uganda |

| ID | Name | ID | Name | ID | Name |
|---|---|---|---|---|---|
| EC | Ecuador | MN | Mongolia | UA | Ukraine |
| EG | Egypt | MS | Montserrat | AE | United Arab Emirates |
| SV | El Salvador | MA | Morocco | US | United States |
| GQ | Equatorial Guinea | MZ | Mozambique | UM | United States Minor Outlying Islands |
| ER | Eritrea | MM | Myanmar | UY | Uruguay |
| EE | Estonia | NA | Namibia | UZ | Uzbekistan |
| ET | Ethiopia | NR | Nauru | VU | Vanuatu |
| FK | Falkland Islands (Malvinas) | NP | Nepal | VA | Vatican City state |
| FO | Faroe Islands | NL | Netherlands | VE | Venezuela |
| FJ | Fiji | AN | Netherlands Antilles | VN | Viet Nam |
| FI | Finland | NC | New Caledonia | VG | Virgin Islands (British) |
| FR | France | NZ | New Zealand | VI | Virgin Islands (US) |
| GF | French Guiana | NI | Nicaragua | WF | Wallis and Futuna |
| PF | French Polynesia | NE | Niger | EH | Western Sahara |
| TF | French Southern Territories | NG | Nigeria | YE | Yemen |
| GA | Gabon | NU | Niue | ZM | Zambia |
| GM | Gambia | NF | Norfolk Island | ZW | Zimbabwe |

# Province Codes

| ID | Name | ID | Name | ID | Name |
|---|---|---|---|---|---|
| AB | Alberta | ME | Maine | PR | Puerto Rico |
| AK | Alaska | MI | Michigan | QC | Quebec |
| AL | Alabama | FM | Micronesia | RI | Rhode Island |
| AS | American Samoa | MN | Minnesota | SC | South Carolina |
| AR | Arkansas | MO | Missouri | SD | South Dakota |
| AZ | Arizona | MS | Mississippi | SK | Saskatchewan |
| BC | British Columbia | MT | Montana | TN | Tennessee |
| CA | California | NB | New Brunswick | TX | Texas |
| CO | Colorado | NC | North Carolina | UT | Utah |
| CT | Connecticut | ND | North Dakota | VA | Virginia |
| DC | District of Columbia | NE | Nebraska | VI | Virgin Islands |
| DE | Delaware | NL | Newfoundland/Labrador | VT | Vermont |
| FL | Florida | NH | New Hampshire | WA | Washington |
| GA | Georgia | NJ | New Jersey | | |
| GU | Guam | NM | New Mexico | WI | Wisconsin |
| HI | Hawaii | NS | Nova Scotia | WV | West Virginia |
| IA | Iowa | NT | Northwest Territories | WY | Wyoming |
| ID | Idaho | NU | Nunavut | YT | Yukon |
| IL | Illinois | NV | Nevada | -- | Outside U.S./Canada |
| IN | Indiana | NY | New York | | |
| KS | Kansas | OH | Ohio | | |
| KY | Kentucky | OK | Oklahoma | | |
| LA | Louisiana | ON | Ontario | | |
| MA | Massachusetts | OR | Oregon | | |
| MB | Manitoba | PA | Pennsylvania | | |
| MD | Maryland | PE | Prince Edward Island | | |

# AVS Response Codes

| ID | Result | Processed | Address Match | Postal/ZIP Match | Message |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Address Verification not performed for this transaction. |
| 5 | 0 | 0 | 0 | 0 | Invalid AVS Response. |
| 9 | 0 | 0 | 0 | 0 | Address Verification Data contains edit error. |
| A | 0 | 1 | 1 | 0 | Street address matches, Postal/ZIP does not match. |
| B | 0 | 1 | 1 | 0 | Street address matches, Postal/ZIP not verified. |
| C | 0 | 1 | 0 | 0 | Street address and Postal/ZIP not verified. |
| D | 1 | 1 | 1 | 1 | Street address and Postal/ZIP match. |
| E | 0 | 0 | 0 | 0 | Transaction ineligible. |
| G | 0 | 0 | 0 | 0 | Non AVS participant. Information not verified. |
| I | 0 | 0 | 0 | 0 | Address information not verified for international transaction. |
| M | 1 | 1 | 1 | 1 | Street address and Postal/ZIP match. |
| N | 0 | 1 | 0 | 0 | Street address and Postal/ZIP do not match. |
| P | 0 | 1 | 0 | 1 | Postal/ZIP matches. Street address not verified. |
| R | 0 | 0 | 0 | 0 | System unavailable or timeout. |
| S | 0 | 0 | 0 | 0 | AVS not supported at this time. |
| U | 0 | 0 | 0 | 0 | Address information is unavailable. |
| W | 0 | 1 | 0 | 1 | Postal/ZIP matches, street address does not match. |
| X | 1 | 1 | 1 | 1 | Street address and Postal/ZIP match. |
| Y | 1 | 1 | 1 | 1 | Street address and Postal/ZIP match. |
| Z | 0 | 1 | 0 | 1 | Postal/ZIP matches, street address does not match. |

# CVD Response Codes

| ID | Message |
|---|---|
| 1 | CVD Match |
| 2 | CVD Mismatch |
| 3 | CVD Not Verified |
| 4 | CVD Should have been present |
| 5 | CVD Issuer unable to process request |
| 6 | CVD Not Provided |

# URL Encoding Chart

| Code | | Code | | Code | | Code | | Code | | Code | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| %00 | | %30 | 0 | %60 | ` | %90 | | %C0 | À | %F0 | ð |
| %01 | | %31 | 1 | %61 | a | %91 | ‘ | %C1 | Á | %F1 | ñ |
| %02 | | %32 | 2 | %62 | b | %92 | ’ | %C2 | Â | %F2 | ò |
| %03 | | %33 | 3 | %63 | c | %93 | “ | %C3 | Ã | %F3 | ó |
| %04 | | %34 | 4 | %64 | d | %94 | ” | %C4 | Ä | %F4 | ô |
| %05 | | %35 | 5 | %65 | e | %95 | • | %C5 | Å | %F5 | õ |
| %06 | | %36 | 6 | %66 | f | %96 | – | %C6 | Æ | %F6 | ö |
| %07 | | %37 | 7 | %67 | g | %97 | — | %C7 | Ç | %F7 | ÷ |
| %08 | | %38 | 8 | %68 | h | %98 | ˜ | %C8 | È | %F8 | ø |
| %09 | Tab | %39 | 9 | %69 | i | %99 | ™ | %C9 | É | %F9 | ù |
| %0A | LF | %3A | : | %6A | j | %9A | š | %CA | Ê | %FA | ú |
| %0B | | %3B | ; | %6B | k | %9B | › | %CB | Ë | %FB | û |
| %0C | | %3C | < | %6C | l | %9C | œ | %CC | Ì | %FC | ü |
| %0D | CR | %3D | = | %6D | m | %9D | | %CD | Í | %FD | ý |
| %0E | | %3E | > | %6E | n | %9E | | %CE | Î | %FE | þ |
| %0F | | %3F | ? | %6F | o | %9F | Ÿ | %CF | Ï | %FF | ÿ |
| %10 | | %40 | @ | %70 | p | %A0 | | %D0 | Ð | | |
| %11 | | %41 | A | %71 | q | %A1 | ¡ | %D1 | Ñ | | |
| %12 | | %42 | B | %72 | r | %A2 | ¢ | %D2 | Ò | | |
| %13 | | %43 | C | %73 | s | %A3 | £ | %D3 | Ó | | |
| %14 | | %44 | D | %74 | t | %A4 | ¤ | %D4 | Ô | | |
| %15 | | %45 | E | %75 | u | %A5 | ¥ | %D5 | Õ | | |
| %16 | | %46 | F | %76 | v | %A6 | ¦ | %D6 | Ö | | |
| %17 | | %47 | G | %77 | w | %A7 | § | %D7 | × | | |
| %18 | | %48 | H | %78 | x | %A8 | ¨ | %D8 | Ø | | |
| %19 | | %49 | I | %79 | y | %A9 | © | %D9 | Ù | | |
| %1A | | %4A | J | %7A | z | %AA | ª | %DA | Ú | | |
| %1B | | %4B | K | %7B | { | %AB | « | %DB | Û | | |
| %1C | | %4C | L | %7C | \| | %AC | ¬ | %DC | Ü | | |

| Code | | Code | | Code | | Code | | Code | | Code | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| %1D | | %4D | M | %7D | } | %AD | | %DD | Ý | | |
| %1E | | %4E | N | %7E | ~ | %AE | ® | %DE | Þ | | |
| %1F | | %4F | O | %7F | | %AF | ¯ | %DF | ß | | |
| %20 | space | %50 | P | %80 | | %B0 | ° | %E0 | à | | |
| %21 | ! | %51 | Q | %81 | | %B1 | ± | %E1 | á | | |
| %22" | " | %52 | R | %82 | , | %B2 | ² | %E2 | â | | |
| %23 | # | %53 | S | %83 | ƒ | %B3 | ³ | %E3 | ã | | |
| %24 | $ | %54 | T | %84 | „ | %B4 | ´ | %E4 | ä | | |
| %25 | % | %55 | U | %85 | … | %B5 | µ | %E5 | å | | |
| %26 | & | %56 | V | %86 | † | %B6 | ¶ | %E6 | æ | | |
| %27 | ' | %57 | W | %87 | ‡ | %B7 | · | %E7 | ç | | |
| %28 | ( | %58 | X | %88 | ˆ | %B8 | ¸ | %E8 | è | | |
| %29 | ) | %59 | Y | %89 | ‰ | %B9 | ¹ | %E9 | é | | |
| %2A | * | %5A | Z | %8A | Š | %BA | º | %EA | ê | | |
| %2B | + | %5B | [ | %8B | ‹ | %BB | » | %EB | ë | | |
| %2C | , | %5C | \ | %8C | Œ | %BC | ¼ | %EC | ì | | |
| %2D | - | %5D | ] | %8D | | %BD | ½ | %ED | í | | |
| %2E | . | %5E | ^ | %8E | | %BE | ¾ | %EE | î | | |
| %2F | / | %5F _ | _ | %8F | | %BF | ¿ | %EF | ï | | |

# Appendix B: Sample Script

Sample code is provided to assist developers and is not designed to be used without modification.

The following examples demonstrate how to submit a transaction to the Beanstream server via the Server-To-Server method using various programming languages. In each of these examples, the following sample parameters will be submitted to the Process Transaction API:

requestType=BACKEND&merchant_id=109040000&trnCardOwner=Paul+Randal&trnCardNumber=6220982130610767738&trnOrderNumber=2232&trnAmount=10.00&ordEmailAddress=prandal@mydomain.net&ordName=Paul+Randal&ordPhoneNumber=9999999&ordAddress1=1045+Main+Street&ordAddress2=&ordCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCountry=CA

## Sample ASP Code 1

The following example uses ASP and the Microsoft XML Core Services (MSXML) version 4.0. (MSXML is also known as the Microsoft XML Parser). We do not recommend using WinInet to do the POST because WinInet is not thread safe, and hence is not suitable for use in server applications.

To work with this example, you must have MSXML 3.0 or 4.0 installed on your server. For more information, see the MSDN documentation at http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml

```
<%
option explicit

'Set to the address of the Beanstream server.
const BEANSTREAM_SERVER = "www.beanstream.com"
const MERCHANT_ID      = 109040000
const TERM_URL         = "https://www.merchantserver.com/auth_script.asp"

dim objXMLHTTP
dim beanstreamResponse
dim postData

'Send transaction request string to be posted to the Beanstream system
postData=
"requestType=BACKEND&trnType=P&trnCardNumber=6220982130610517737&trnExpMonth=01&trnExpYear=2012& trnAmount=1%2e00&merchant_id=" & MERCHANT_ID &
"&trnCardOwner=Paul+Randal&trnOrderNumber=1a&ordEmailAddress=prandal@mydomain.net&ordName=Paul+Randal&ordPhoneNumber=60411234567&ordAddress1=1045+Main+Street&ordAddress2=&or
```

```
dCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCountry=CA&termUrl=" &
server.urlEncode(TERM_URL) & "&sessionId=" & request("sessionId")

'Create the ServerXMLHTTP object
set objXMLHTTP = Server.CreateObject( "MSXML2.ServerXMLHTTP.4.0" )

'This is the location of the Beanstream payment gateway
objXMLHTTP.Open "POST", "https://" & BEANSTREAM_SERVER & "/scripts/process_transaction.asp",
false

'Set the HTTP header's content type
objXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"

'Submit the transaction request to the Beanstream server
objXMLHTTP.Send( postData )

'Read the transaction response returned from the Beanstream system
beanstreamResponse = objXMLHTTP.ResponseText

'We have now received a response from Beanstream. Now check if this response is a Redirection
'Response Page by checking the value of the responseType parameter. If the responseType paramter
'is set to "R" it is a redirection repsonse. If the response type parameter is a "T" it is a
'transaction approved/delined response. For datawave cards the system should always return a
'redirection response.

'response.write beanstreamResponse : response.end
if GetQueryValue(beanstreamResponse, "responseType" ) = "R" then
        'We have a Redirection Response Page, so show it to the browser to redirec the user to datawave
for verification
        response.write GetQueryValue(beanstreamResponse, "pageContents")
else
         'This is a normal transaction, so beanstreamResponse contains the results of the transaction.
         if GetQueryValue(beanstreamResponse, "trnApproved" ) = "1" then
                response.write "Transaction Approved"
         else
                response.write "Transaction Declined: " & beanstreamResponse
         end if
end if


Function GetQueryValue(queryString, paramName)
 'Purpose: To return the value of a parameter in an HTTP query string.
 'Pre:     queryString is set to the full query string of url encoded name value pairs. ex:
"value1=one&value2=two&value3=3"
 '    paramName is set to the name of one of the parameters in the queryString. ex: "value2"
 'Post:    None
```

```
 'Returns: The function returns the query string value assigned to the paramName parameter. ex: "two"

   Dim pos1
   dim pos2
   Dim qString

   qString = "&" & queryString & "&"
   pos1 = InStr(1, qString, paramName & "=")
   If pos1 > 0 Then
      pos1 = pos1 + Len(paramName) + 1
      pos2 = InStr(pos1, qString, "&")
      If pos2 > 0 Then
         GetQueryValue = DecodeQueryValue(Mid(qString, pos1, pos2 - pos1))
      End If
   End If

End Function

Function DecodeQueryValue(qValue)
 'Purpose: To URL decode a string
 'Pre:     qValue is set to a url encoded value of a query string parameter. ex: "one+two"
 'Post:    none
 'Returns: Returns the url decoded value of qValue. ex: "one two"

   Dim i
   Dim qChar
   dim newString

            if IsNull(qValue) = false then
               For i = 1 To Len(qValue)
                  qChar = Mid(qValue, i, 1)
                  If qChar = "%" Then
                                 on error resume next
                    newString = newString & Chr("&H" & Mid(qValue, i + 1, 2))
                                 on error goto 0
                    i = i + 2
                  ElseIf qChar = "+" Then
                     newString = newString & " "
                  Else
                     newString = newString & qChar
                  End If
               Next
                        DecodeQueryValue = newString
            else
                        DecodeQueryValue = ""
            end if
```

```
End Function
%>
```

## Sample PHP code

The following example uses PHP and the libcurl CURL library. To work with this example, you must install the CURL package. CURL allows you to connect to servers using a variety of protocols, and in this example, it uses it to communicate with Beanstream via HTTPS POST. For information on how to install CURL, see the PHP manual at http://www.php.net/manual/en/ref.curl.php.

```
<?php
// Initialize curl
$ch = curl_init();

// Get curl to POST
curl_setopt( $ch, CURLOPT_POST, 1 );
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST,0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);


// Instruct curl to suppress the output from Beanstream, and to directly
// return the transfer instead. (Output will be stored in $txResult.)
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, 1 );

// This is the location of the Beanstream payment gateway
curl_setopt( $ch, CURLOPT_URL, "https://www.beanstream.com/scripts/process_transaction.asp" );

// These are the transaction parameters that we will POST
curl_setopt( $ch, CURLOPT_POSTFIELDS,
"requestType=BACKEND&merchant_id=109040000&trnCardOwner=Paul+Randal&trnCardNumber=51000
00010001004&trnExpMonth=01&trnExpYear=05&trnOrderNumber=2232&trnAmount=10.00&ordEmailA
ddress=prandal@mydomain.net&ordName=Paul+Randal&ordPhoneNumber=9999999&ordAddress1=104
5+Main+Street&ordAddress2=&ordCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCount
ry=CA" );

// Now POST the transaction. $txResult will contain Beanstream's response
$txResult = curl_exec( $ch );

echo "Result:<BR>";
echo $txResult;

curl_close( $ch );
?>
```

## Sample Java Code

The section contains an example of how to POST a transaction to the Beanstream server using Java. It has been tested with JDK 1.3 and 1.4.

```
import java.io.*;
import java.net.*;
import javax.net.ssl.*;

public class HttpsPost
{
	public static void main( String[] args ) throws Exception
	{
		int ch;

		// These are the transaction parameters that we will POST
		String messageString =
"requestType=BACKEND&merchant_id=109040000&trnCardOwner=Paul+Randal&trnCardNumber=51000
00010001004&trnExpMonth=01&trnExpYear=05&trnOrderNumber=2232&trnAmount=10.00&ordEmailA
ddress=prandal@mydomain.net&ordName=Paul+Randal&ordPhoneNumber=9999999&ordAddress1=104
5+Main+Street&ordAddress2=&ordCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCount
ry=CA";

		// Set the location of the Beanstream payment gateway
		URL url = new URL( "https://www.beanstream.com/scripts/process_transaction.asp" );

		// Open the connection
		URLConnection conn = url.openConnection();

		// Set the DoOutput flag to true because we intend
		// to use the URL connection for output
		conn.setDoOutput( true );

		// Send the transaction via HTTPS POST
		OutputStream ostream = conn.getOutputStream();
		ostream.write( messageString.getBytes() );
		ostream.close();

		// Get the response from Beanstream
		InputStream istream = conn.getInputStream();
		while( ( ch = istream.read() ) != -1 )
		{
			System.out.print( ( char )ch );
		}
		istream.close();
	}
```

```
}
```

### To Use This Example:

In order use the sample code, you will need to complete the following:

- ✓ Install the Java Secure Socket Extension (JSSE) if you are using a version of the JDK earlier than 1.4
- ✓ Ensure that jsse.jar, jnet.jar and jcert.jar are in your classpath if using a version of the JDK earlier than 1.4
- ✓ Ensure that the java.security file is complete
- ✓ Import the Equifax certificate to the client's (your computer's) trusted certificate keystore

### Installing JSSE

If you are using a version of the JDK that is earlier than version 1.4, you will need to download and install the Java Secure Socket Extension. This will implement a Java version of Secure Sockets Layer (SSL), which is required to securely communicate with the Beanstream server. You can download it from the Sun website at http://java.sun.com/products/jsse/ .

### Setting the Classpath

If you are using a version of the JDK that is earlier than version 1.4, you will need to ensure that jsse.jar, jnet.jar and jcert.jar are in your classpath. In Windows, this is done by modifying the CLASSPATH environment variable in Control Panel → System → Advanced tab. Under the *Advanced* tab, click the *Environment Variables* button to bring up the *Environment Variables* dialog. In the *System Variables* section of this dialog, make sure there is a variable called CLASSPATH and that it contains paths to jsse.jar, jnet.jar and jcert.jar.

***In UNIX/Linux***, there are two ways set the CLASSPATH environment variable, depending on your shell. In csh, the CLASSPATH is modified with the setenv command. For example:
setenv CLASSPATH=/usr/java/jdk1.3.1_01/jre/lib/jsse.jar

***In sh***, the CLASSPATH is modified with these commands:
CLASSPATH=/usr/java/jdk1.3.1_01/jre/lib/jsse.jar export CLASSPATH

### Modify java.security

Your java.security file should contain the following lines. If not, you will need to add them.

security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.net.ssl.internal.ssl.Provider
security.provider.3=com.sun.rsajca.Provider

## Adding the Equifax Certificate to the Keystore

Beanstream uses a certificate provided by Equifax, which Java does not recognize. Because of this, you will need to add the Equifax certificate (provided by Beanstream) to your computer's trusted certificate keystore, which is a file called cacerts. To do this, use the keytool utility provided by the JDK. For example: keytool -import -alias equifax -keystore cacerts -file ESCA.cer

The above example will work if you are in the directory where the cacerts file is located and have copied the ESCA.cer certificate to the same directory. If this is not the case, you will need to specify the correct pathnames to these files.

In UNIX/Linux, the cacerts file is located in your JDK directory under ./jre/lib/security/. In Windows, there may be two copies of the cacerts file—one in the JDK directory under .\jre\lib\security, and one in the Program Files directory under .\java\j2re1.4.0_01\lib\security (JDK 1.3) or .\java\j2re1.4.0_01\lib\security (JDK 1.4). Usually, the cacerts file in the Program Files directory is the one that is used, but if that doesn't work for you, try the one in the JDK directory.

If you do not have the ESCA.cer file, you can download it from Beanstream via the following URL: https://www.beanstream.com/admin/support/ESCA.cer

## Troubleshooting

***Issue*** I've imported the Equifax certificate into my cacerts file, but I still get the error: "Exception in thread "main" javax.net.ssl.SSLHandshakeException: Could not find trusted certificate".

***Resolution*** You may not have added the certificate to the existing cacerts file. If you run the keytool utility to install the certificate and keystore cannot find the cacerts file, it will create a new one in the current directory. Make sure that you have added the certificate to the existing cacerts file by specifying the correct path to the cacerts file when running the keytool utility, or by running the keytool utility while in the directory where cacerts is located.

Also, if you are using Windows, there may be more than one cacerts file. It is commonly located in both the JDK directory and in Program Files\Javasoft (JDK 1.3) or Program Files\Java (JDK 1.4). This may be the reason that the Java runtime reports that the certificate has not been imported into the cacerts file.

***Issue*** I get the following error: "java.net.MalformedURLException: unknown protocol: https".

***Resolution*** You need to install the Java Secure Socket Extension (JSSE). You can download it from the Sun website at http://java.sun.com/products/jsse/.

## ASP Example with Verified by Visa

The following script is an example of how to integrate a Verified by Visa-capable solution using ASP and the Microsoft XML Core Services (MSXML) version 4.0. (MSXML is also known as the Microsoft XML Parser).

This piece of code will perform the initial transaction request, and if a redirection response page is found in the response, will show this page to the client's web browser. The Terminal URL page used here is https://www.beanstream.com/samples/sample_s2s_vbv_auth.asp. You will have to change this to your actual Terminal URL for this example to work. (The line containing the location of the Terminal URL page has been bolded for your convenience.)

***To use this example***, you must have MSXML 3.0 or 4.0 installed on your server. For more information on how to download and install MSXML, see the MSDN documentation at http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml

```
<%
option explicit

'Set to the address of the Beanstream server.
const BEANSTREAM_SERVER = "www.beanstream.com"
const MERCHANT_ID      = 107380000
const TERM_URL         = "https://www.beanstream.com/samples/sample_s2s_vbv_auth.asp"

dim objXMLHTTP
dim beanstreamResponse
dim postData

'Send transaction request string to be posted to the Beanstream system
postData=
"requestType=BACKEND&trnType=P&trnCardNumber=4030000010001234&trnExpMonth=12&trnExpYea
r=22&trnAmount=1%2e00&merchant_id=" & MERCHANT_ID &
"&trnCardOwner=Paul+Randal&trnOrderNumber=1a&ordEmailAddress=prandal@mydomain.net&ordNa
me=Paul+Randal&ordPhoneNumber=60411234567&ordAddress1=1045+Main+Street&ordAddress2=&or
dCity=Vancouver&ordProvince=BC&ordPostalCode=V8R+1J6&ordCountry=CA&termUrl=" &
server.urlEncode(TERM_URL)

'Create the ServerXMLHTTP object
set objXMLHTTP = Server.CreateObject( "MSXML2.ServerXMLHTTP.4.0" )
objXMLHTTP.setOption(2) = 4096
objXMLHTTP.setOption(3) = ""

'This is the location of the Beanstream payment gateway
```

```
objXMLHTTP.Open "POST", "https://" & BEANSTREAM_SERVER & "/scripts/process_transaction.asp",
false

'Set the HTTP header's content type
objXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"

'Submit the transaction request to the Beanstream server
objXMLHTTP.Send( postData )

'Read the transaction response returned from the Beanstream system
beanstreamResponse = objXMLHTTP.ResponseText

'We have now received a response from Beanstream. Now check if this response is a Redirection
'Response Page by checking the value of the responseType parameter. If the responseType paramter
'is set to "R" it is a redirection repsonse. If the response type parameter is a "T" it is a
'transaction approved/delined response.

'response.write beanstreamResponse : response.end
if GetQueryValue(beanstreamResponse, "responseType" ) = "R" then
        'We have a Redirection Response Page, so show it to the browser
        response.write GetQueryValue(beanstreamResponse, "pageContents")
else
        'This is a normal transaction, so beanstreamResponse contains the results of the transaction.
        if GetQueryValue(beanstreamResponse, "trnApproved" ) = "1" then
                response.write "Transaction Approved"
        else
                response.write "Transaction Declined: " & beanstreamResponse
        end if
end if


Function GetQueryValue(queryString, paramName)
 'Purpose: To return the value of a parameter in an HTTP query string.
 'Pre:    queryString is set to the full query string of url encoded name value pairs. ex:
"value1=one&value2=two&value3=3"
 '    paramName is set to the name of one of the parameters in the queryString. ex: "value2"
 'Post:    None
 'Returns: The function returns the query string value assigned to the paramName parameter. ex: "two"

   Dim pos1
   dim pos2
   Dim qString

   qString = "&" & queryString & "&"
   pos1 = InStr(1, qString, paramName & "=")
   If pos1 > 0 Then
```

```
      pos1 = pos1 + Len(paramName) + 1
      pos2 = InStr(pos1, qString, "&")
      If pos2 > 0 Then
        GetQueryValue = DecodeQueryValue(Mid(qString, pos1, pos2 - pos1))
      End If
    End If

End Function

Function DecodeQueryValue(qValue)
 'Purpose: To URL decode a string
 'Pre:     qValue is set to a url encoded value of a query string parameter. ex: "one+two"
 'Post:    none
 'Returns: Returns the url decoded value of qValue. ex: "one two"

    Dim i
    Dim qChar
    dim newString

            if IsNull(qValue) = false then
              For i = 1 To Len(qValue)
                qChar = Mid(qValue, i, 1)
                If qChar = "%" Then
                                on error resume next
                  newString = newString & Chr("&H" & Mid(qValue, i + 1, 2))
                                on error goto 0
                  i = i + 2
                ElseIf qChar = "+" Then
                  newString = newString & " "
                Else
                  newString = newString & qChar
                End If
              Next
                    DecodeQueryValue = newString
            else
                    DecodeQueryValue = ""
            end if

End Function
%>
```

# ASP Terminal URL Page Sample

```
<%
'This is a sample Terminal URL page that the merchant must have on their web
'server. The Issuer Access Control Server (ACS) will redirect to this page
'during the Authentication stage (after the customer enters his password).

set objXMLHTTP = Server.CreateObject("MSXML2.ServerXMLHTTP.4.0")
objXMLHTTP.Open "POST", "https://www.beanstream.com/scripts/process_transaction_auth.asp", false
objXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
objXMLHTTP.Send("PaRes=" & request("PaRes") & "&MD=" & request("MD"))
response.write objXMLHTTP.ResponseText
set objXMLHTTP = nothing
%>
```

## HASH Validation

The following code could be used to calculate a hashValue for your transaction request. This example uses an SHA-1 hash algorithm. Once the hashValue is generated, you would need to append this value to your transaction string as per the instructions in section 12.2.

```
' PURPOSE:
' Creating a secure identifier from person-identifiable data
'
' The function SecureHash generates a 160-bit (20-hex-digit) message digest for a given message (String).
'
' SAMPLE:
' Message: "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"
' Returns Digest: "84983E441C3BD26EBAAE4AA1F95129E5E54670F1"
' Message: "abc"
' Returns Digest: "A9993E364706816ABA3E25717850C26C9CD0D89D"

Function AndW(w1, w2)
Dim arr(3)
arr(0) = w1(0) And w2(0)
arr(1) = w1(1) And w2(1)
arr(2) = w1(2) And w2(2)
arr(3) = w1(3) And w2(3)
AndW = arr
End Function
Function OrW(w1, w2)
Dim arr(3)
arr(0) = w1(0) Or w2(0)
arr(1) = w1(1) Or w2(1)
arr(2) = w1(2) Or w2(2)
arr(3) = w1(3) Or w2(3)
OrW = arr
End Function
Function XorW(w1, w2)
Dim arr(3)
arr(0) = w1(0) Xor w2(0)
arr(1) = w1(1) Xor w2(1)
arr(2) = w1(2) Xor w2(2)
arr(3) = w1(3) Xor w2(3)
XorW = arr
End Function
Function NotW(w)
Dim arr(3)
arr(0) = Not w(0)
arr(1) = Not w(1)
arr(2) = Not w(2)
arr(3) = Not w(3)
NotW = arr
End Function
Function AddW(w1, w2)
```

```
Dim I, arr(3)

I = CLng(w1(3)) + w2(3)
arr(3) = I Mod 256
I = CLng(w1(2)) + w2(2) + (I \ 256)
arr(2) = I Mod 256
I = CLng(w1(1)) + w2(1) + (I \ 256)
arr(1) = I Mod 256
I = CLng(w1(0)) + w2(0) + (I \ 256)
arr(0) = I Mod 256

AddW = arr
End Function
Function CircShiftLeftW(w, n)
Dim d1, d2

d1 = WordToDouble(w)
d2 = d1
d1 = d1 * (2 ^ n)
d2 = d2 / (2 ^ (32 - n))
CircShiftLeftW = OrW(DoubleToWord(d1), DoubleToWord(d2))
End Function
Function WordToHex(w)
WordToHex = Right("0" & Hex(w(0)), 2) & Right("0" & Hex(w(1)), 2) & Right("0" & Hex(w(2)), 2) & Right("0" & Hex(w(3)), 2)
End Function
Function HexToWord(H)
HexToWord = DoubleToWord(CDbl("&H" & H))
End Function
Function DoubleToWord(n)
Dim arr(3)
arr(0) = Int(DMod(n, 2 ^ 32) / (2 ^ 24))
arr(1) = Int(DMod(n, 2 ^ 24) / (2 ^ 16))
arr(2) = Int(DMod(n, 2 ^ 16) / (2 ^ 8))
arr(3) = Int(DMod(n, 2 ^ 8))
DoubleToWord = arr
End Function
Function WordToDouble(w)
WordToDouble = (w(0) * (2 ^ 24)) + (w(1) * (2 ^ 16)) + (w(2) * (2 ^ 8)) + w(3)
End Function
Function DMod(value, divisor)
DMod = value - (Int(value / divisor) * divisor)
If DMod < 0 Then DMod = DMod + divisor
End Function
Function F(t, B, C, D)
Dim casenum
If t <= 19 Then casenum = 1
If t <= 39 And t > 19 Then casenum = 2
If t <= 59 And t > 39 Then casenum = 3
If t > 59 Then casenum = 4
Select Case casenum
Case 1
```

```
F = OrW(AndW(B, C), AndW(NotW(B), D))
Case 2
F = XorW(XorW(B, C), D)
Case 3
F = OrW(OrW(AndW(B, C), AndW(B, D)), AndW(C, D))
Case 4
F = XorW(XorW(B, C), D)
End Select
End Function
Function sha1(inMessage)

Dim inLenW
Dim w(79)
Dim temp
Dim A, B, C, D, E
Dim H0, H1, H2, H3, H4
Dim K(3)
Dim arr(3)
Dim inLen, padMessage, numBlocks, blockText, wordText, I, t

inLen = Len(inMessage)
inLenW = DoubleToWord(CDbl(inLen) * 8)

padMessage = inMessage & Chr(128) & String((128 - (inLen Mod 64) - 9) Mod 64, Chr(0)) & String(4, Chr(0)) & Chr(inLenW(0)) &
Chr(inLenW(1)) & Chr(inLenW(2)) & Chr(inLenW(3))

numBlocks = Len(padMessage) / 64

' initialize constants
K(0) = HexToWord("5A827999")
K(1) = HexToWord("6ED9EBA1")
K(2) = HexToWord("8F1BBCDC")
K(3) = HexToWord("CA62C1D6")

'initialize 160-bit (5 words) buffer
H0 = HexToWord("67452301")
H1 = HexToWord("EFCDAB89")
H2 = HexToWord("98BADCFE")
H3 = HexToWord("10325476")
H4 = HexToWord("C3D2E1F0")

'each 512 byte message block consists of 16 words (W) but W is expanded to 80 words
For I = 0 To numBlocks - 1
blockText = Mid(padMessage, (I * 64) + 1, 64)
'initialize a message block
For t = 0 To 15
wordText = Mid(blockText, (t * 4) + 1, 4)
arr(0) = Asc(Mid(wordText, 1, 1))
arr(1) = Asc(Mid(wordText, 2, 1))
arr(2) = Asc(Mid(wordText, 3, 1))
arr(3) = Asc(Mid(wordText, 4, 1))
```

```
w(t) = arr
Next
'create extra words from the message block
For t = 16 To 79
'W(t) = S^1 (W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16))
w(t) = CircShiftLeftW(XorW(XorW(XorW(w(t - 3), w(t - 8)), w(t - 14)), w(t - 16)), 1)
Next

'make initial assignments to the buffer
A = H0
B = H1
C = H2
D = H3
E = H4

'process the block
For t = 0 To 79
temp = AddW(AddW(AddW(AddW(CircShiftLeftW(A, 5), F(t, B, C, D)), E), w(t)), K(t \ 20))
E = D
D = C
C = CircShiftLeftW(B, 30)
B = A
A = temp
Next

H0 = AddW(H0, A)
H1 = AddW(H1, B)
H2 = AddW(H2, C)
H3 = AddW(H3, D)
H4 = AddW(H4, E)
Next

sha1 = WordToHex(H0) & WordToHex(H1) & WordToHex(H2) & WordToHex(H3) & WordToHex(H4)
End Function
```

## Sample URL Decode Function

```
Function GetQueryValue(queryString, queryParam)
  Dim pos1, pos2
  Dim qString
  qString = "&" & queryString & "&"
  pos1 = InStr(1, qString, queryParam)
  If pos1 > 0 Then
    pos1 = pos1 + Len(queryParam) + 1
    pos2 = InStr(pos1, qString, "&")
    If pos2 > 0 Then
      GetQueryValue = UrlDecode(Mid(qString, pos1, pos2 - pos1))
            else
```

```
                        response.write "pos2 less than or equal to 0" : response.end
    End If
        else
                response.write "pos1 less than or equal to 0" : response.end
  End If
End Function

Function UrlDecode(qValue)

  Dim i
  Dim qChar
        dim newString

        if IsNull(qValue) = false then
          For i = 1 To Len(qValue)
            qChar = Mid(qValue, i, 1)
            If qChar = "%" Then
                                on error resume next
              newString = newString & Chr("&H" & Mid(qValue, i + 1, 2))
                                on error goto 0
              i = i + 2
            ElseIf qChar = "+" Then
              newString = newString & " "
            Else
              newString = newString & qChar
            End If
          Next
                UrlDecode = newString
  else
                UrlDecode = ""
        end if

End Function

If GetQueryValue(trnResponse, "responseType")) = "R" then
response.redirect(URLDecode3(GetQueryValue(trnResponse, "pageContents")))
end if
```