Developer Guide: VirtualMerchant

One Concourse Parkway, Suite 300, Atlanta, GA 30328

Copyright

Disclaimer

Elavon, Inc., provides this publication “as is” without warranty of any kind, either expressed or implied. This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Elavon, Inc., may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time.

Trademarks

VirtualMerchant is a registered trademark of Elavon, Inc. All other brand and product names are trademarks or registered trademarks of their respective companies.

Related Documents

VirtualMerchant User Guide (Document # DG-001-10-DO-Orig)

Typographical Conventions

Throughout this user guide you will see words and phrases that appear in different fonts and formats. The following table describes the typographical conventions used in this user guide.

Item	Convention	Example
Notes containing important information	Denoted by a change in font and possibly color; bold	**NOTE:** This is an example.
Reference document titles	Bold italics	Batch Management tasks are covered in a separate document, the ***Batch Management User Guide***.
Sample Code	Fixed-width font (Courier New)	`Sample Code 123 45`
Typed Commands	Fixed-width font (Courier New); bold	Type **`cd \Elavon\pbase`** in a command window.

Contents

Chapter 1 Introduction

The VirtualMerchant application is a secure, server-based system that supports transaction processing (authorization and settlement) in real-time. There are two ways to submit transactions to the VirtualMerchant application:

- Through the Virtual Terminal
- Through an integrated application using the VirtualMerchant API

The Virtual Terminal allows the use of a standard Web browser to process transactions as a cost-effective payment solution. This is where you can manage your payment account, submit transactions, monitor and review unsettled transactions, search for and view settled transactions, configure account settings, and more. For help with the Merchant Interface features and settings, see the Virtualmerchant User Guide.

The VirtualMerchant API enables you (the developer) to write a point of sale application (website, software application, shopping cart, and so forth) that interfaces with VirtualMerchant payment gateway to process—a full range of payment types including credit card, debit card, food stamp, cash benefit, electronic check, gift card, and recurring and installment transactions.

The integration supports communicating to the VirtualMerchant gateway for Ecommerce, mail order and retail for both card present and non-present environments. This guide focuses on the processes and settings available to the developer to interface to the VirtualMerchant payment gateway.

VirtualMerchant can accept as few as four pieces of data from your application, and do the rest of the work on its own by gathering information directly from the customer and using the settings that have been configured in the VirtualMerchant administration section. Alternately, you could use VirtualMerchant as a backend feature to your integrated application, completely transparent to your customers, by writing the process that gathers all of the pertinent customer information and the receipt page that displays the outcome of the transaction processing to the user. We find that most merchants fall somewhere in the middle of these scenarios, gathering some data from their customers before sending them out to VirtualMerchant and then letting VirtualMerchant gather more information from the customer and display the receipt after transaction approval.

Chapter 2 Getting Started

The **Getting Started** chapter describes how to:

- Get a unique test account
- Communicate with VirtualMerchant

Getting a Unique Test Account

Prior to beginning a VirtualMerchant integration, integrators must request a unique test account with the **Enable HTTP Transaction** option enabled, to be able to perform transactions from an integrated solution. Contact Elavon Internet product support group at internetproductsupport@merchantconnect.com or 1-800-377-3962 to submit your request for a test account.

The test account consists of a combination of login credentials that can be used to authenticate to the VirtualMerchant application. Once a test account is created, a VID, user ID and user password are generated and can be used to access the merchant virtual terminal at https://demo.myvirtualmerchant.com/VirtualMerchantDemo/login.do. The use of this unique VID keeps transactions separated by account. The integrator can then retrieve a unique PIN that can be used to send transactions through the API by utilizing `process.do` or `processxml.do`. This allows integrators to test all aspects of the integration, as well as access the user interface from transaction processing, to batch settlement and reporting, to reconciliation.

The following information must be provided to the support group:

- Company name
- Primary contact name
- Primary contact phone
- Primary e-mail address

Communicating with VirtualMerchant

VirtualMerchant accepts information sent using HTTPS, either by the HTTP GET or POST method. The data you send, along with VirtualMerchant settings, will determine:

- How transactions are handled
- The appearance and styling of VirtualMerchant's payment form
- How VirtualMerchant handles receipt and error pages, among other things

VirtualMerchant currently supports two different ways to integrate:

- **`process.do`** using Key / Value Pair in an HTML formatted request

  Example: `ssl_amount=1.00`

  Or

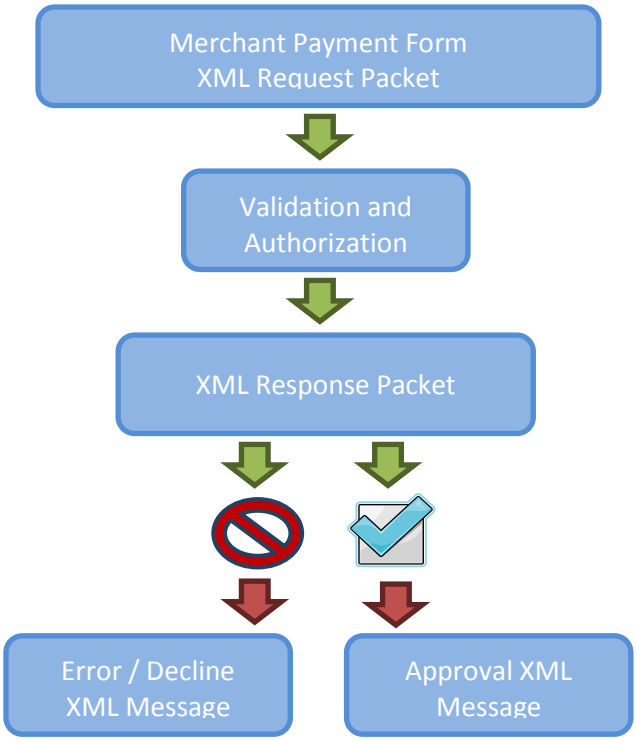
- **`processxml.do`** using Key / Value Pair in an XML formatted request

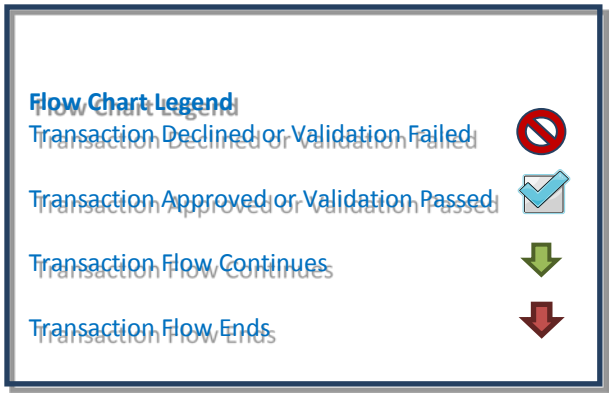
  Example: `<ssl_amount>1.00</ssl_amount>`

The programming language used can be any language that supports Hypertext Transfer Protocol Secure (HTTPS). HTTPS is a combination of the Hypertext Transfer Protocol with the Secure Socket Layer/Transport Layer Security (SSL/TLS) protocol to provide encryption and transaction submission.

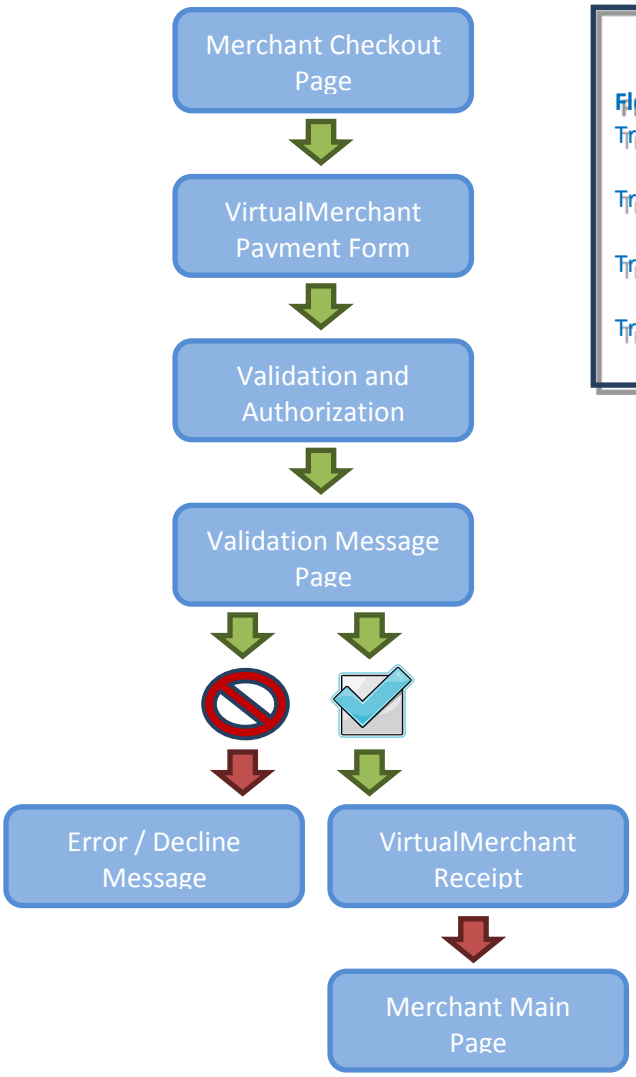
The following three flow charts illustrate the process of using integration for processing transactions:

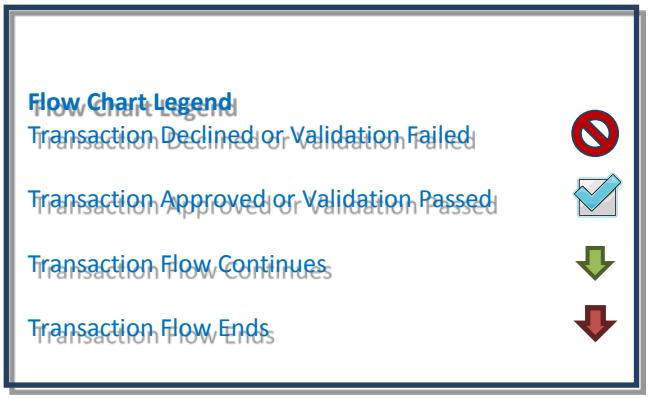
Processxml.do



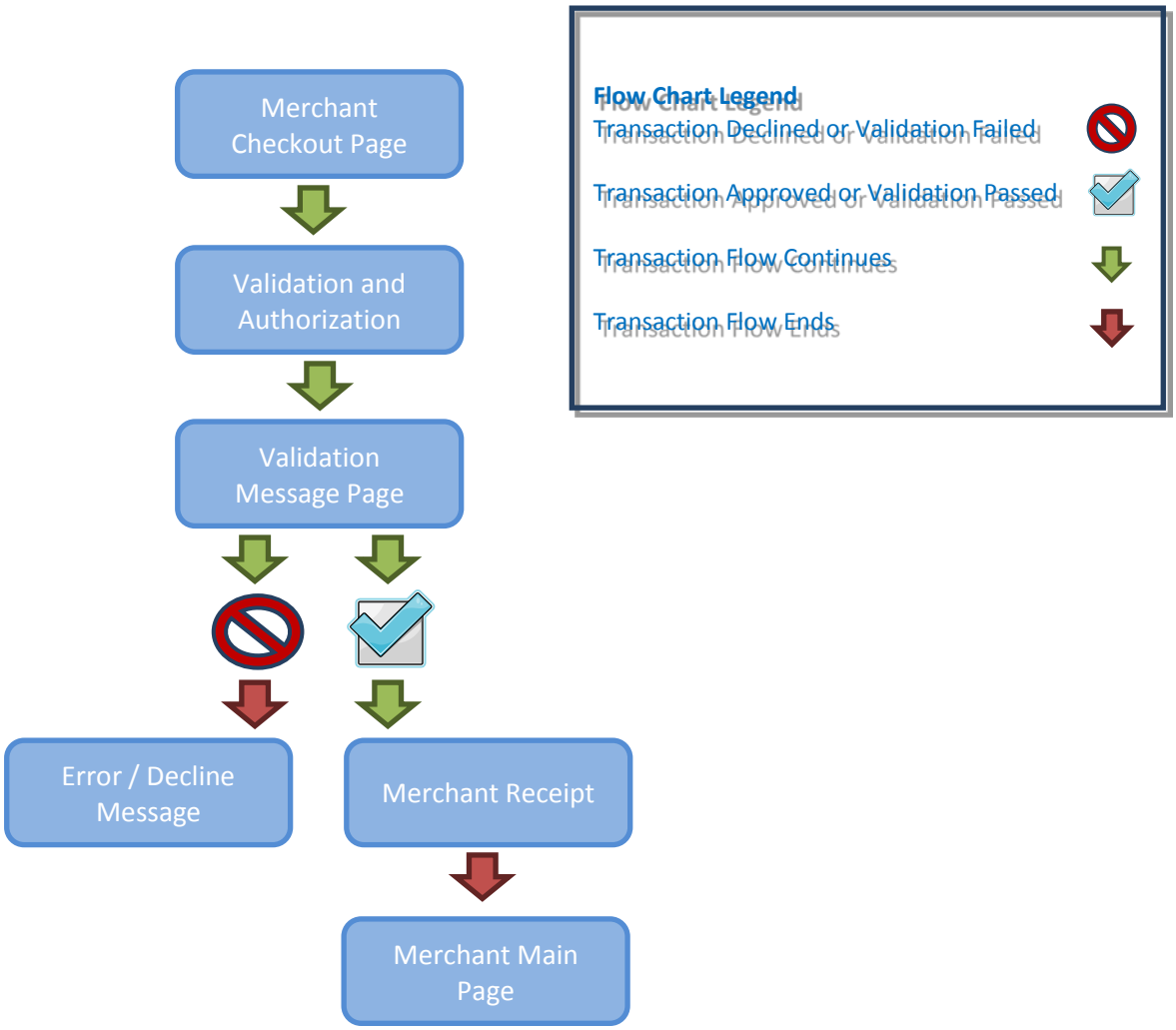


Process.do (ssl_show_form = true)





Process.do (ssl_show_form = false)



Chapter 3 Processing Transactions

This chapter reviews the basic guidelines to submit credit, debit, food stamp, cash benefits, electronic check, gift card, recurring, and installment transactions. Steps to implement those transactions using `process.do` and `processxml.do` are outlined in the transaction flows and API reference chapters.

Topics include:

- Processing standard transactions
- Processing recurring and installment transactions

Processing Standard Transactions

The following section provides the guidelines on how to submit and modify transactions through the VirtualMerchant gateway.

To Submit Standard Transactions

Credit Card Transactions

- All credit card transactions sent for approval to VirtualMerchant must pass the following basic cardholder information:
  - Account Data
    - Card Number along with an expiration date for hand keyed transactions
    - Track Data for swiped transactions
  - Amount
- The more cardholder information passed during authorization, the better. CVV and AVS are used in charge back and fraud protection, as well as to determine transaction fees. Therefore, it is highly recommended that the following data be passed by the payment application during the authorization process on hand-keyed credit transactions:
  - AVS Data
  - CVV2 / CVC / CID value
  - Invoice Number
  - Card Present Indicator
- Commercial cards and purchasing cards require additional data to receive lower processing fees. The basic "Level II" data that should be passed at time of authorization is as follows:
  - Customer Code
  - Sales Tax

Card Verification Value (CVV)

- CVV is one of the credit card industry's several acronyms for the credit card security code that helps to verify the legitimacy of a credit card. Depending on the card, the security code can be a three-digit or four-digit number, printed either on either the back or the front of the card. The card verification value code CID/CVV2 value is optional for Visa, MasterCard, Discover, and American Express. Terminals must be set up to allow passing CID/CVV2. The integrated application should not store or print the CVV2, CVC2, or CID data from the back or front of credit cards. This value is used for fraud protection and is recommended to be passed on card not present environments such as e-commerce and MOTO. When CVV2 data is passed, a CVV2 Response Code is returned in the Authorization Response. Refer to the Authorization Response Codes section for more information.

Address Verification Service (AVS) Data

- An integrated application should pass the AVS data on card not present environments such as e-commerce and MOTO to qualify for better rates by using the Address Verification Service (AVS). AVS captures ZIP codes and the cardholder's billing address. AVS information is then compared to the cardholder's ZIP code and address that the card issuer has on file. Address and ZIP code mismatches help the merchant to decide whether or not to complete the transaction. Refer to the Authorization Response Codes section for more information and for a complete list of the AVS Response codes.

Debit Card Transactions

- Debit card transactions must be swiped and require an encrypted device for PIN entry. The device must be encrypted with Elavon Keys. Track data, along with the following information derived must be passed from the device:
  - DUKPT Value: This is the value returned by the PIN pad device which was used to encrypt the cardholder's personal identification number (PIN), using the Derived Unique Key per Transaction (DUKPT) method.
  - PIN Block: The encrypted PIN entered by a Debit / EBT cardholder as identification for a transaction.

Partial Approvals

- An integrated solution with VirtualMerchant must indicate the support of partial approval by sending "1" in the `ssl_partial_auth_indicator` field in the authorization request. This solution must then be capable of reading the new added returned response fields to facilitate split tender if needed. This field is ignored on those transactions that do not apply.

  When a transaction is partially approved, VirtualMerchant will return the following to `processxml.do` and `process.do`:
  - "PARTIAL APPROVAL" response in the `ssl_result_message` field
  - The authorized approved amount in the `ssl_amount` field
  - The amount originally requested in the `ssl_requested_amount` field
  - The remaining balance due in the `ssl_balance_due` field. This is the difference of the amount requested versus the amount authorized that the merchant has to collect from the consumer
  - The `ssl_account_balance`, which is always set to "0.00" for a partially authorized transaction

The point of sale application may reverse a partially approved transaction by sending either a transaction type of `ccdelete` (if processing under a terminal-based terminal) or `ccvoid` (if processing under an host-based terminal). The transaction ID associated with the partially approved transaction must be included.

Custom Defined Fields

- VirtualMerchant allows merchants and developers to set up custom defined fields that can be used through the gateway interface. The custom field size is limited to 999 alphanumeric characters. Those fields can be set up in the payment fields section of the Virtual Terminal user interface. Only 25 fields can be set up for any given terminal and special characters should not be used. It is not allowed to pass any sensitive data, including but not limited to PAN data such as full card number, expiration date, track data, or CID/CVV2 data from a credit card into a custom field. Additionally, customer account numbers, social security numbers, and other private data should not be passed unmasked or unprotected.

Open Batches

- Open batches should be closed on a daily basis. End of day procedures should consist of reviewing open batches for accuracy, and then closing those batches out by settling through the VirtualMerchant GUI. Elavon highly recommends that batches be closed out on a daily basis. There are two options for batching based on merchant configuration. These options are either time-initiated batch close or merchant-initiated batch close. Both options are currently only supported through the VirtualMerchant GUI.

- Settings are available within the Admin portion of the Virtual Terminal that can block transactions from being added to a current open batch, if they do not meet certain qualifications. Elavon recommends that merchants review these settings prior to accepting transactions.

To Modify Standard Transactions

VirtualMerchant assigns a transaction ID to every transaction in the system. This is a unique number associated with a single transaction and is returned in the transaction response. To perform an update on a transaction that was previously authorized, VirtualMerchant requires the transaction ID to be passed to reference an existing transaction, eliminating the need to store or pass any sensitive data.

The transaction ID (`ssl_txn_id`) allows VirtualMerchant to pull all needed information associated with the original transaction (card number, expiration date, auth code, etc.) from the transaction record in the database to update the transaction.

Currently VirtualMerchant supports the following transactions:

- Void (`ccvoid/ecsvoid`)
- Delete (`ccdelete`)
- Add Signature (`ccsignature`)

Void a Transaction

VirtualMerchant allows voids using the original transaction ID from an approved transaction on the following payment and transaction types:

- Credit Card Sale (using `ccvoid`)
- Credit Card Return (using `ccvoid`)
- Credit Card Force (using `ccvoid`)
- E-Check Guarantee/Verification/Conversion Only (using `ecsvoid`)

NOTE: There is a 10-minute limit on voiding e-check transactions.

Delete a Transaction

A transaction type of `ccdelete` sends a reversal attempt to the issuer and deletes a **Sale** or **Auth Only** transaction from the **Auth Only** and **Main** current batches.

`ccdelete` may be used in a partial approval scenario. When a consumer decides not to continue with an additional tender type, the point of sale application must send a reversal to cancel the payment and restore the balance to the card. A reversal is achieved by sending a delete request when processing under terminal-based terminals or by sending a void request when processing under host-based terminals.

Reversals free up cardholders open to buy amounts by reducing issuer holds on available balances when transactions are not completed. This reduces declines at the point of sale and the amount of cardholder complaints that are unpleasant for all parties involved.

Add a Signature

A transaction type of `ccsignature` adds an electronic signature data to previously approved credit card and e-check transactions and uses any signature capable device. The following transaction types can be assigned a signature by passing the original transaction ID obtained from an approved transaction along with the signature data:

- Credit Card Sale
- Credit Card Force
- Credit Card Auth Only
- Credit Card Return
- E-Check (Guarantee, Conversion, Verification)

All signature images must be **BASE64** encoded in the following supported formats:

- TIFF
- Windows Supported Bitmap
- PNG
- JPEG
- JPG

Processing Recurring and Installment Transactions

The following section provides the guidelines on how to add, submit and modify recurring and installment transactions through the VirtualMerchant gateway.

To Add Recurring and Installment Transactions

Adding recurring and installment transactions will allow an application to setup automatic billing of credit cards on specific intervals (i.e. monthly, quarterly, annually) for an indeterminate or fixed number of payments. This will help to simplify the process of billing a cardholder for a product or a service that is provided on a continuous basis.

VirtualMerchant allows adding a recurring or installment transaction on the following transaction types:

- Add a Recurring Transaction (`ccaddrecurring`)
- Add an Installment Transaction (`ccaddinstall`)

All recurring and installment transactions sent to VirtualMerchant must pass the following required information:

- Account Data
  - Card Number
  - Expiration Date
- Amount: The amount to be charged for each recurring billing
- Billing Cycle: The frequency in which the system should process the charge. There are eleven different values available:
  - DAILY
  - WEEKLY
  - BI-WEEKLY
  - SEMI-MONTHLY
  - MONTHLY
  - BI-MONTHLY
  - QUARTERLY
  - SEMESTER
  - SEMI-ANNUALLY
  - ANNUALLY
  - SUSPENDED
- Next Payment Date: The date to start billing the customer; Format MM/DD/YYYY
- Total Number of Installment: For installment payment plans ONLY

It is highly recommended that the following optional data be passed by the payment application when adding a recurring or installment transaction:

- AVS Data
- Invoice Number
- Last Name
- First Name

The following data is optional:

- Skip Payment
- Custom Data Fields

The following data should never be stored, therefore cannot be passed to VirtualMerchant on recurring/installement transactions:

- CVV2 / CVC / CID value
- Track Data for swiped transactions

The following data is conditional and should be passed to VirtualMerchant on the scenarios described below:

- The Last Day of the Month
- Bill on Half Indicator

When the payment application provides any of the following dates as the Start Payment Date for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles, it must indicate if the recurring transaction will be processed on the last day of the month:

- 30-Apr
- 30-June
- 30-Sept
- 30-Nov
- 28-Feb (non-leap year)
- 29-Feb (leap year)

When the payment application provides any of the following dates as the Start Payment Date for the annually billing cycles, it must indicate if the recurring transaction will be processed on the last day of the month:

- 28-Feb (non-leap year)
- 29-Feb (leap year)

When the payment application provides a semi-monthly billing cycle, it must indicate if the transaction is to be processed on the the 1st and the 15th of the month or the 15th and the last day of the month using the Bill on Half indicator.

To Modify Recurring and Installment Transactions

Once a transaction is added, VirtualMerchant assigns and returns a unique recurring ID to every recurring transaction or a unique installment ID to every installment transaction setup in the system. To update or submit the recurring transaction for a payment, VirtualMerchant requires the recurring ID or installment ID to be passed to reference an existing recurring or installment transaction, eliminating the need to store or pass any sensitive data.

The recurring ID (`ssl_recurring_id`) or installment ID (`ssl_installment_id`) allows VirtualMerchant to pull all needed information associated with the original recurring or installment transaction (card number, expiration date, billing cycle, etc.) from the recurring or installment record in the database. It is stored by the application and used to modify an existing recurring or installment transaction.

Currently, VirtualMerchant allows updates on an existing recurring and installment transaction using one of the following transaction types:

- Update Recurring `(ccupdaterecurring)`
- Delete Recurring `(ccdeleterecurring)`
- Submit Recurring for Payment `(ccrecurringsale)`
- Update Installment `(ccupdateinstall)`
- Delete Installment `(ccdeleteinstall)`
- Submit Installment for Payment `(ccinstallsale)`

Update Recurring and Installment Transactions

In the event of a change in the customer information, VirtualMerchant allows updates to the original recurring or installment transaction on the following transaction types:

- Update a Recurring Transaction (using `ccupdaterecurring`)
- Update an Installment Transaction (using `ccupdateinstall`)

The following information must be passed when updating a recurring or installment transaction:

- The recurring ID (`ssl_recurring_id`) with `ccupdaterecurring`

  Or

- The installment ID (`ssl_installment_id`) with `ccupdateinstall`

One or more of the following values can be provided when updating a recurring or installment transaction:

- Account Data
  - Card Number
  - Expiration Date
- Amount
- Billing Cycle
- Next Payment Date
- AVS Data
- Invoice Number

- Last Name
- First Name
- Skip Payment
- Custom Data Fields
- Total Number of Installment: for Installment payment plans only

The following data are conditional and should be passed to VirtualMerchant on the scenarios described previously; please refer to "Adding Recurring and Installment Transactions" section:

- The Last Day of the Month
- Bill on Half Indicator

NOTES:

- Only the information that needs to be updated should to be passed.
- Records may be rendered "suspended,"meaning that the customer record and product information remains on file, but an authorization on the transaction will NOT be attempted on the "next payment" date. This temporarily suspends customer's transactions, if necessary, without losing data. To manually suspend a customer's record, update a transaction with billing cycle of "SUSPENDED."

Delete Recurring and Installment Transactions

If a recurring or installment record is no longer needed and a customer record must be removed, the following transaction types can be used:

- Delete a Recurring Transaction (`ccdeleterecurring`)
- Delete an Installment Transaction (`ccdeleteinstall`)

The following information must be passed when deleting a recurring or installment transaction:

- The recurring ID (`ssl_recurring_id`) with `ccdeleterecurring`

  Or

- The installment ID (`ssl_installment_id`) with `ccdeleteinstall`

NOTE: The data cannot be recovered once deleted.

Submit Recurring and Installment Transactions for Payment

A transaction type of `ccrecurringsale or ccinstallsale` sends a **Sale** authorization to bill the card on file outside of the specified billing cycle for the same amount setup previously in the record, thus facilitating on demand billing if needed. Once authorized, the payment number will increase, the next payment date will not change and payments will continue to run as usual in their billing cycle.

The following information must be passed when submitting a recurring or installment transaction for a payment:

- The recurring ID (`ssl_recurring_id`) with `ccrecurringsale`

  Or

- The installment ID (`ssl_installment_id`) with `ccinstallsale`

Chapter 4 Payment Integration

This chapter reviews the integration requirements for the VirtualMerchant API. Elavon encourages integrators to read through the entire manual prior to coding their Payment Applications.

REMINDER: Integrators must make certain that their applications meet all PA-DSS guidelines prior to use in a live merchant environment. For the most up-to-date information pertaining to guidelines, refer to the Transaction Security section at the end of this document.

The Payment Form is where customers enter the necessary or required personal and credit card information required to process transactions. It is also the page that sends transactions to the VirtualMerchant system for authorization processing. You can provide information in two ways:

- Merchant payment form
  If you provide your own payment form to the customer, your form must send all of the necessary data to complete the transaction into VirtualMerchant.
- VirtualMerchant payment form
  If VirtualMerchant provides the payment form to the customer on your behalf, you only need to give the system enough information to know who you are, along with any special information about your transaction that the customer is not going to enter.

NOTE: It is strongly recommended that you do not use the Admin user ID to process transactions through gateway integration. Create a user ID specifically for this purpose instead. This allows you to more accurately track how transactions occur and who submits them. This also protects you in the event of a security compromise by limiting what transaction types the user ID has access to.

Merchant Payment Form

This section explains how to send information for VirtualMerchant to process credit card transactions without additional input from your customer.

If you want to collect all of the data from the customer, and only send the information to VirtualMerchant after it has all been gathered, you can do so. To hide the payment form, you must send the parameter **`ssl_show_form`** with a value of **false**.

NOTE: This method is the preferred route for integrators that use server-side scripting and for those who want complete control of their point of sale systems. In this case, VirtualMerchant serves as a payment pass-thru or a gateway, completely transparent to the cardholder. This method offers flexibility to the integrator. However, because the card data will be entered into the the payment application (in this case, the point of sale system), it is subject to a higher level of PA-DSS scope.

The use of server-side scripting allows custom HTML to be delivered to a client machine. The code that generates the custom HTML is processed on the Web server before the HTML is sent to the user's machine over the Internet. Server-side scripting is an option for delivering customized HTML. This

method is in contrast to client-side scripting, where the HTML is modified (typically by Java-script, in the client's machine), after the HTML and Java are sent from the Web server. The primary strength of server-side scripting to VirtualMerchant integration is the ability to hide the sensitive processing credentials from the browser. This method is possible using any CGI language that has the ability to build and send HTTP POST messages. The API fields that comprise the sensitive processing credentials are:

- `ssl_merchant_id`
- `ssl_user_id`
- `ssl_pin`

Each account can have one merchant Admin user and can have multiple employee users, who may or may not have access to run transactions. When you specify a user ID, make sure that the submitted PIN matches the user ID that you are submitting and the terminal on which you wish to run the transaction. When an account has more than one terminal, it is the combination of `ssl_merchant_id`, `ssl_pin` and `ssl_user_id` that VirtualMerchant uses to determine which of the terminals the transaction is being submitted to. When `ssl_user_id` is omitted, the user ID is assumed to be the same as the `ssl_merchant_id`, the merchant Admin user.

NOTE: The following data is required for all transactions. For security purposes, Elavon recommends this data be sent using SSI through an SSL connection:

- `ssl_merchant_id` (referred to as your account ID or VirtualMerchant ID)
- `ssl_user_id`
- `ssl_pin`
- `ssl_transaction_type`
- `ssl_show_form (set to false)`

You must include some additional information when you use your customized payment form. The additional required fields that you must pass to VirtualMerchant are:

- `ssl_amount`
- `ssl_card_number` and `ssl_exp_date` for manually entered transactions

  Or
- `ssl_track_data` for swiped transactions

There are also conditional fields that should be supplied based on VirtualMerchant configuration information. These include card present indicator, AVS and CVV data. The conditional field requirements will be reviewed further in another section of this guide.

VirtualMerchant Payment Form

This section explains how to send information to have VirtualMerchant present a payment form to your customer. This payment form will gather information from your customer such as the name displayed on their credit card, card number, expiration date, billing and shipping address, as well as other fields you specify in your Web page's code or in the Terminal Setup section of your VirtualMerchant account. The `ssl_show_form` property must be set to "true" and it is only available thru `process.do`.

The first step is to submit the minimum information to VirtualMerchant. The minimum information required to provide a payment form to your customer is the following fields:

- `ssl_merchant_id` (referred to as your Account ID or VirtualMerchant `ID`
- `ssl_user_id`
- `ssl_pin`
- `ssl_transaction_type`
- `ssl_show_form (set to true)`

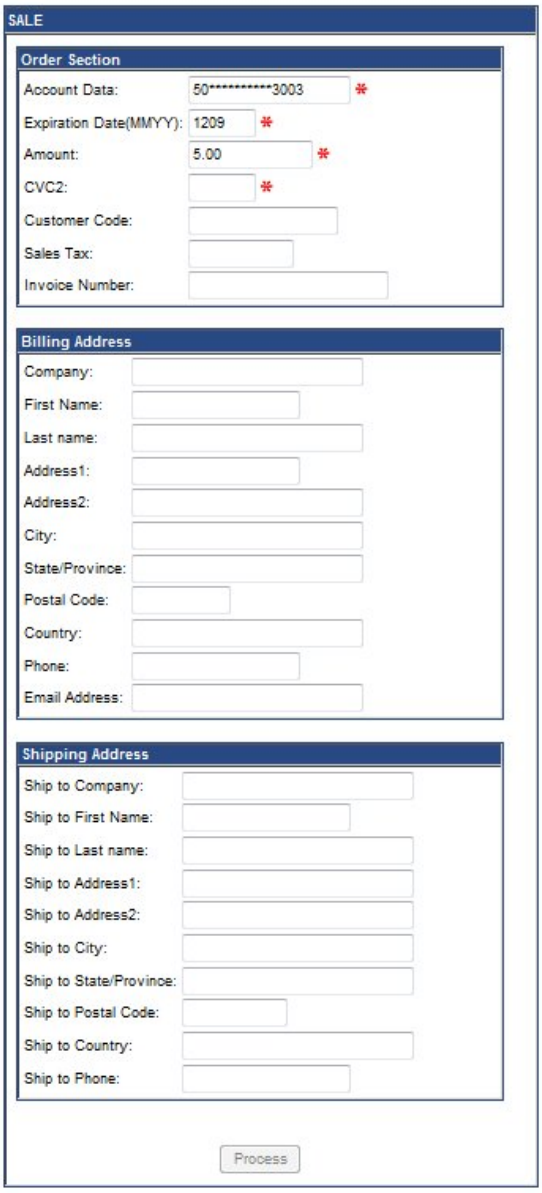
NOTE: Typically, this method is used when integrating using `process.do` in an e-commerce environment. This method, although less work for the integrator, is also less flexible. When you use this form to collect cardholder data such as card number, expiration date and CVV2, you can reduce the level of PA-DSS scrutiny.

If you have more than one terminal assigned to your account, you must ensure that the PIN you use corresponds to the correct terminal. With these two pieces of information, VirtualMerchant can display a payment form that allows your customers to enter all of the transaction data based on the settings you have pre-determined in your VirtualMerchant account.

If you want to integrate VirtualMerchant with a website that offers paid goods or services, and want to charge for those goods or services by credit card, use the following procedure:

1. Create a form on your website.
2. Set the action of the form to:
   `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do`
3. Set the method of the form to POST.
4. Add a field with the name `ssl_merchant_id`.
5. Set the value of `ssl_merchant_id` to the VirtualMerchant account ID.
6. Add a field with the name `ssl_pin`.
7. Set the value of `ssl_pin` to the merchant PIN associated with the VirtualMerchant ID.
8. Add a field with the name `ssl_amount`.
9. Set the value of `ssl_amount` to the desired amount.
10. Add a Submit button.

With `ssl_show_form` is set equal to **true**, a form similar to the following image (fields are displayed based on the Admin settings in the VirtualMerchant configuration) displays and contains all information submitted in the transaction request sent to `process.do`:



Chapter 5 Receipt Options

A receipt is the customer's documentation of the outcome of a transaction. The receipt can be displayed in two ways:

- VirtualMerchant receipt
  If VirtualMerchant draws the receipt for you, you do not need to include logic to parse through the VirtualMerchant result. However, your customer might not return to your website when the transaction is complete.
- Merchant receipt
  If you draw your own receipt, your form must handle the data received from VirtualMerchant to correctly communicate to your customers the outcome of their transactions.

VirtualMerchant Receipt

This section shows you how to have VirtualMerchant display the receipt to your customer. The receipt has many configuration possibilities that can be driven by code or by choices made in the Administration section of the VirtualMerchant website. Refer to the VirtualMerchant User Guide for more information about how to use the VirtualMerchant website to configure your receipt options.

Input

Four primary variables dictate how receipts are processed:

- `ssl_result_format`
- `ssl_receipt_link_method`
- `ssl_receipt_link_url`
- `ssl_receipt_link_text`

You also have the option to use variations of the last three variables to allow for a different type of receipt for approvals and declines. If you use the variables above, they will take precedence over the following parameters:

- `ssl_receipt_decl_method`
- `ssl_receipt_decl_get_url`
- `ssl_receipt_decl_post_url`
- `ssl_receipt_decl_text`
- `ssl_receipt_apprvl_method`
- `ssl_receipt_apprvl_get_url`
- `ssl_receipt_apprvl_post_url`
- `ssl_receipt_apprvl_link_text`

ssl_result_format

The ssl_result_format has two acceptable values: ASCII and HTML. If you do not specify the format, an HTML receipt will be returned. If you specify ASCII, only a list of key/value pairs will be returned, and the other receipt related parameters you sent will be ignored. The ASCII format is intended to be called by a separate application that will process the data, instead of directly by a webpage used by a customer that initiates a transaction.

ssl_receipt_link_method

The various ssl_receipt_link_method variables have four options:

- GET
- POST
- LINK
- REDG (REDirect GET)

The first two choices use the button at the bottom of the receipt for the customer to select whether to return to your website. The two options pass the transaction's data back to your site using the method chosen. LINK presents a hyperlink at the bottom of the VirtualMerchant receipt page and does not transmit data back to your website. REDG (RE-Direct Get) is covered in more details in the next section.

Output

An HTML page displays and notifies whether the transaction was approved or not. If the transaction was approved, the receipt displays the data elements that make up the transaction. A link back to your website is displayed at the bottom of the page. This link is configured based on the parameters you send or by the configuration settings specified in the VirtualMerchant administrative website. You can set the format to ASCII or override the receipt link parameter in your code. It is also possible to specify the behavior for the approvals separate from the behavior of the declines.

A receipt containing ssl_result = 0 represents an approved transaction. A receipt that contains any other value for ssl_result represents a declined transaction or a transaction that had an error that prevented it from being authorized. Refer to the Error Codes section for more information.

Merchant Receipt

This section explains what you need to do to show your customer a receipt of your own creation for a VirtualMerchant transaction. The receipt has many configuration possibilities that can be driven by code, or by choices made in the Administration section of the VirtualMerchant website. Refer to the VirtualMerchant User Guide for more information on using the VirtualMerchant website to configure your receipt options.

Input

Four primary variables dictate how receipts are processed:

- `ssl_result_format`
- `ssl_receipt_link_method`
- `ssl_receipt_link_url`
- `ssl_receipt_link_text`

In addition, you can use the variables below to allow for a different type of receipt for approvals and declines. If you use the variables above, they will take precedence over the following parameters:

- `ssl_receipt_decl_method`
- `ssl_receipt_decl_get_url`
- `ssl_receipt_decl_post_url`
- `ssl_receipt_decl_text`
- `ssl_receipt_apprvl_method`
- `ssl_receipt_apprvl_get_url`
- `ssl_receipt_apprvl_post_url`
- `ssl_receipt_apprvl_link_text`

Output

The `ssl_result_format` has two acceptable values:

- ASCII
- HTML

If you do not specify the result format, an HTML receipt will be returned. If you select ASCII, only a list of key/value pairs will be returned. The other receipt-related parameters you have set are ignored. The ASCII format is recommended if you are using an intermediary application to send transactions to VirtualMerchant, rather than sending transactions directly from an HTML form on a Web page that is driven by your customer's actions. The ASCII format will allow you to easily parse through the transaction data and choose what to display to your customer, and what data to use in other ways for your own application.

`Receipt Link Method: Re-direct Get`

There are four options for the various `ssl_receipt_link_method` variables. To display a receipt of your own you must use REDG (RE-Direct Get). REDG will redirect the customer's browser to the URL of your choosing, as soon as the transaction is processed by VirtualMerchant.

Using the various `ssl_receipt_link_url` variables, VirtualMerchant gives you the option to send approved and declined transactions to the same URL or to different URLs to handle them separately. If you use the REDG method and wish to have separate approved and declined behaviors, you must use the **get** versions of the `ssl_receipt_link_url` variables, to specify the destination URL. Specifically:

- `ssl_receipt_decl_get_url`
- `ssl_receipt_apprvl_get_url`

Chapter 6 Transaction Flows

This chapter reviews the transaction flows and provides implementation guidelines and examples to format and send transactions using `process.do` and `processxml.do`.

Topics include:

- Standard transactions
- Partially approved transactions
- Dynamic currency conversion (DCC) transactions
- 3D Secure transactions
- Recurring transactions
- Installment transactions

Standard Transactions

Transaction Flow

1. Submit a transaction request using HTTPS, either by the HTTP GET or POST with the values shown in the API Reference Chapter in this guide.

   Shown below are the key value pairs from the header by themselves for a credit card sale transaction:

   `ssl_merchant_id`=xxxxxx

   `ssl_user_id`=xxxxxxx

   `ssl_pin`=xxxxxx

   `ssl_show_form`=false

   `ssl_card_number`=5000000000000003

   `ssl_exp_date`=1208

   `ssl_amount`=1.00

   `ssl_error_url`=http://www.url.com/cgi-bin/testtran.cgi

   `ssl_result_format`=HTML

   `ssl_transaction_type`=ccsale

   `ssl_receipt_decl_method`=REDG

   `ssl_receipt_decl_get_url`=http://www.url.com/cgi-bin/testtran.cgi

   `ssl_receipt_apprvl_method`=REDG

   `ssl_receipt_apprvl_get_url`=http://www.url.com/cgi-bin/testtran.cgi

2. When VirtualMerchant receives this post, it starts to parse the data to look for a few key fields first. It validates the `ssl_merchant_id`, `ssl_user_id` and `ssl_pin` to authenticate the user.

3. If the supplied information is invalid, an error is returned that states that the information is invalid. If the data is valid, VirtualMerchant continues to validate the other supplied information such as the card number, expiration date, amount of the transaction, type of transaction, address information, and other custom data fields that are passed. Other fields that are passed with transactions states how the transactions should be handled from a communications standpoint. These fields are:

   `ssl_error_url`=http://www.url.com/cgi-bin/testtran.cgi

   `ssl_result_format`=HTML

   `ssl_receipt_decl_method`=REDG

   `ssl_receipt_decl_get_url`=http://www.url.com/cgi-bin/testtran.cgi

   `ssl_receipt_apprvl_method`=REDG

   `ssl_receipt_apprvl_get_url`=http://www.url.com/cgi-bin/testtran.cgi

   `ssl_show_form`=false

   NOTE: You can indicate in the error URL field where you would like VirtualMerchant to send all the errors that are encountered. Any response that is not approved or declined will be sent to the URL specified.

4. By specifying the http://www.url.com/cgi-bin/testtran.cgi url in the `ssl_apprvl_get_url` field for the redirect for the transaction above, the following values are returned for the approved transaction:

   `ssl_card_number`=50********0003

   `ssl_exp_date`=1208

   `ssl_amount`=1.00

   `ssl_result`=0

   `ssl_result_message`=APPROVAL

   `ssl_txn_id`=1016413275E60BB4EC-B4C6-FD4D-A878-F70C3372C986

   `ssl_approval_code`=CVI368

   `ssl_account_balance`=1.00

   `ssl_txn_time`=10/05/2008 10:50:55 AM

5. By specifying the http://www.url.com/cgi-bin/testtran.cgi url in the `ssl_error_url` field for the redirect for the transaction above, the following values (in the example below the PIN is invalid) are returned if the request is invalid:

   `errorCode`=4015

   `errorName`= PIN Invalid

   `errorMessage`= The PIN supplied in the authorization request is invalid

Transaction Examples

process.do

Example 1

In this example, the HTML code demonstrates the initiation of a minimal sale transaction in which VirtualMerchant payment form gathers the entire customer's billing information.

This code creates a button with the label **Click to Order**. When a user clicks the button, it takes the customer to the VirtualMerchant payment form.

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_show_form" value="true">
   <input type="hidden" name="ssl_amount" value="14.95">
   <input type="submit" value="Click to Order">
</form>
```

NOTE: In all of these examples, you will have to change the data values, such as **`my_virtualmerchant_id`**, **`my_user_id, my_pin`**, and the amount of **`14.95`** to values that match your VirtualMerchant account and meet the needs of your websites.

Example 2

The following HTML code demonstrates a very basic form that collects and passes the minimum required data for a complete VirtualMerchant transaction that will not display the VirtualMerchant payment form. This code creates a form that displays the customer's total and asks for their credit card number and expiration date, with a button labeled **Continue**. After the user enters the information and clicks the button, VirtualMerchant processes the transaction. The user is then taken directly to a receipt or result form that displays the outcome of the transaction.

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00">
   <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_show_form" value="false">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
```

```
   <br/>
   <input type="submit" value="Continue">
</form>
```

Example 3

The following HTML code is similar to Example 2 shown above, including additional fields required to pass AVS data and CVV2 / CVC2 data:

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
    .do" method="POST">
   <input type="hidden" name="ssl_merchant_id"
    value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_card_number" value="0000000000000000">
   <input type="hidden" name="ssl_exp_date" value="0000">
   <input type="hidden" name="ssl_amount" value="12.77">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_cvv2cvc2_indicator" value="1">
   <input type="hidden" name="ssl_cvv2cvc2" value="1234">
   <input type="hidden" name="ssl_avs_address" value="123 Main St.">
   <input type="hidden" name="ssl_avs_zip" value="01234">
   <input type="submit" value="Donate Now">
</form>
```

Example 4

The following HTML code passes receipt options in the transaction request to generate a receipt on the payment form to your customer with a link to return to your page:

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00">
   <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <br/>
   <input type="hidden" name="ssl_result_format" value="HTML">
```

```
   <input type="hidden" name="ssl_receipt_decl_method" value="POST">
   <input type="hidden" name="ssl_receipt_decl_post_url"
   value="http://www.website.com/decline.asp">
   <input type="hidden" name="ssl_receipt_apprvl_method" value="GET">
   <input type="hidden" name="ssl_receipt_apprvl_get_url"
   value="http://www.website.com/approval.asp">
   <input type="hidden" name="ssl_receipt_link_text" value="Continue">
   <input type="submit" value="Continue">
</form>
```

Approved Receipt

This generates a receipt that includes the following code for an approved transaction:

```
This is your Receipt<br><br>
…
<!--The visible portion of your receipt will appear here, according to
the configuration settings you applied in the VirtualMerchant
administrative Website.-->
…
<form action="http://www.website.com/approval.asp" method="GET">
   <input type="hidden" name="ssl_result" value="0">
   <input type="hidden" name="ssl_result_message" value="APPROVAL">
   <input type="hidden" name="ssl_txn_id" value="99C7884A-EDB6-4256-BE69-
   4547B8859D5B">
   <input type="hidden" name="ssl_approval_code" value="N29032">
   <input type="hidden" name="ssl_cvv2_response" value="">
   <input type="hidden" name="ssl_avs_response" value=" ">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_amount" value="5.00">
   <input type="hidden" name="ssl_email" value=" test@test.com">
   <br>
   <input type="submit" value="Continue" class="smallbutton">
</form>
```

Decline Receipt

The result could be a form that includes the following code for a declined transaction:

```
<b>An Error Occurred</b><br><br>
Number: 1<br>
Message: This transaction request has not been approved. You may elect to
use another form of payment to complete this transaction or contact
customer service for additional options.<br>
<form action="http://www.website.com/decline.asp" method="POST">
   <input type="hidden" name="ssl_result" value="1">
   <input type="hidden" name="ssl_result_message" value="DECLINED">
   <input type="hidden" name="ssl_txn_id" value="B6637C93-CA38-41C5-951A-
   C995BFFBD708">
   <input type="hidden" name="ssl_approval_code" value=" ">
   <input type="hidden" name="ssl_cvv2_response" value="">
   <input type="hidden" name="ssl_avs_response" value=" ">
```

```
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_amount" value="5.00">
   <input type="hidden" name="ssl_email" value=" test@test.com">
   <br>
   <input type="submit" value="Continue" class="smallbutton">
</form>
```

Example 5

The following HTML code passes receipt options in the transaction request to allow you to generate your own receipt based on ASCII values being returned:

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00">
   <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <br/>
   <input type="hidden" name="ssl_result_format" value="ASCII">
   <input type="submit" value="Continue">
</form>
```

Approved Receipt

This generates a receipt that includes the following key/value pairs for an approved transaction:

```
ssl_result=0
ssl_result_message=APPROVAL
ssl_txn_id=9621F9AD-E49E-4003-91BD-5C1B08569959
ssl_approval_code=N54032
ssl_cvv2_response=
ssl_avs_response=
ssl_invoice_number=123-ABC
ssl_amount=5.00
ssl_card_number=00*******0000
ssl_exp_date=0000
ssl_email=test@test.com
```

Declined Receipt

This generates a receipt that includes the following key/value pairs for declined transaction:

```
ssl_result=0
ssl_result_message=DECLINED
ssl_txn_id=10164132759743E096-C5C1-C6A3-7DE4-FE9525313D47
ssl_approval_code=
ssl_cvv2_response=
ssl_avs_response=
ssl_invoice_number=123-ABC
ssl_amount=5.00
ssl_card_number=00*******0000
ssl_exp_date=0000
ssl_email=test@test.com
```

Example 6

The following HTML code passes receipt options in the transaction request to redirect the customer to your own receipt on the website specified:

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00"> <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <br/>
   <input type="hidden" name="ssl_result_format" value="HTML">
   <input type="hidden" name="ssl_receipt_decl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_decl_get_url"
   value="http://www.website.com/decline.asp">
   <input type="hidden" name="ssl_receipt_apprvl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_apprvl_get_url"
   value="http://www.website.com/approval.asp">
   <input type="submit" value="Continue">
</form>
```

The customer would be redirected to “http://www.website.com/approval.asp” for an Approved transaction or to “http://www.website.com/decline.asp” for a Declined transaction. The transaction data will be passed along as Get variables in the query string of the URL.

processxml.do

The following XML code example demonstrates the initiation of a basic transaction request and response.

NOTE: In the case of XML, no additional information can be collected by VirtualMerchant when the request is sent. All required data must be sent in the transaction request to process the transaction.

Initial Request

```
<txn>
    <ssl_merchant_id> my_virtualmerchant_id</ssl_merchant_id>
    <ssl_user_id>my_user</ssl_user_id>
    <ssl_pin>my_pin</ssl_pin>
    <ssl_test_mode>false</ssl_test_mode>
    <ssl_transaction_type>ccsale</ssl_transaction_type>
    <ssl_card_number>5000000000000002</ssl_card_number>
    <ssl_exp_date>1215</ssl_exp_date>
    <ssl_amount>1.00</ssl_amount>
</txn>
```

Response Receipt

```
<txn>
   <ssl_card_number>50********0002</ssl_card_number>
   <ssl_exp_date>1215</ssl_exp_date>
   <ssl_amount>1.00</ssl_amount>
   <ssl_result>0</ssl_result>
   <ssl_result_message>APPROVAL</ssl_result_message>
   <ssl_txn_id>101641221593ACBA6-BAFD-76B7-4948-B3DE68CFD0CC</ssl_txn_id>
   <ssl_approval_code>CMC142</ssl_approval_code>
   <ssl_account_balance>1.00</ssl_account_balance>
   <ssl_txn_time>01/20/2011 01:07:23 PM</ssl_txn_time>
</txn>
```

Partially Approved Transactions

Transaction Flow

1. Submit a transaction request using HTTPS, either by the HTTP GET or POST with the values as shown in the API Reference Chapter for `ccsale` or `ccauthonly.` The `ssl_partial_auth_indicator` should be set to **1**, to indicate the support of partial approval processing and the requested amount should be sent in the `ssl_amount` field per current functionality.
2. Receive a response as shown in the API Reference Chapter for a `ccsale` or `ccauthonly`.
3. Check for a result of **`0`** in the `ssl_result` field as well as the values passed the following fields:
    - **Partial Approval** response in the `ssl_result_message` field.
    - The authorized approved amount in the `ssl_amount` field.
    - The amount originally requested in the `ssl_requested_amount` field.
    - The remaining balance due in the `ssl_balance_due` field. This is the difference of the amount requested versus the amount authorized that the merchant has to collect from the consumer.
    - The `ssl_account_balance`, which is always set to **`0.00`** for a partially authorized transaction.
4. Prompt the consumer to pay the remainder of the amount by initiating new `ccsale` or `ccauthonly` requests.
5. If cardholders decide not to proceed with the transaction, reverse a partially approved transaction by sending either a transaction type of `ccdelete` if processing as a terminal-based terminal or `ccvoid` if processing as a host-based terminal. The transaction ID associated with the partially approved transaction must be included.

Transaction Examples

process.do

Request

This demonstrates an example of a request with partial approval indicator:

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="10.10"> <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_partial_auth_indicator" value="1">
   <input type="hidden" name="ssl_show_form" value="true">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <br/>
   <input type="hidden" name="ssl_result_format" value="HTML">
   <input type="hidden" name="ssl_receipt_decl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_decl_get_url"
   value="http://www.website.com/decline.asp">
   <input type="hidden" name="ssl_receipt_apprvl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_apprvl_get_url"
   value="http://www.website.com/approval.asp">
   <input type="submit" value="Continue">
</form>
```

Response

The issuer in this case has indicated that the card balance did not cover the entire amount requested; however, the transaction was approved partially. The application must collect the remainder of the amount by initiating a new transaction.

```
This is your Receipt<br><br>
…
<!--The visible portion of your receipt will appear here, according to
the configuration settings you applied in the VirtualMerchant
administrative Website.-->
…
<form action="http://www.website.com/approval.asp" method="GET">
   <input type="hidden" name="ssl_result" value="0">
   <input type="hidden" name="ssl_result_message" value="PARTIAL
   APPROVAL">
   <input type="hidden" name="ssl_txn_id" value="99C7884A-EDB6-4256-BE69-
   4547B8859D5B">
   <input type="hidden" name="ssl_approval_code" value="N29032">
   <input type="hidden" name="ssl_cvv2_response" value="">
   <input type="hidden" name="ssl_avs_response" value=" ">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_amount" value="10.05">
   <input type="hidden" name="ssl_requested_amount" value="10.10">
   <input type="hidden" name="ssl_balance_due" value="0.05">
   <input type="hidden" name="ssl_account_balance" value="0.00">
   <input type="hidden" name="ssl_email" value=" test@test.com">
   <br>
   <input type="submit" value="Continue" class="smallbutton">
</form>
```

processxml.do

Send a ccsale request

The point of sale application will send a partial approval of **1** to indicate the support of partial approval processing. The requested amount is sent per current functionality in the `ssl_amount` field.

```
<txn>
  <ssl_merchant_id>My_Merchant_ID</ssl_merchant_id>
  <ssl_user_id>my_user_id</ssl_user_id>
  <ssl_pin>my_pin</ssl_pin>
  <ssl_test_mode>false</ssl_test_mode>
  <ssl_transaction_type>ccsale</ssl_transaction_type>
  <ssl_card_number>4111111111111111</ssl_card_number>
  <ssl_exp_date>1212</ssl_exp_date>
  <ssl_amount>10.10</ssl_amount>
  <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
  <ssl_cvv2cvc2>123</ssl_cvv2cvc2>
  <ssl_first_name>Test</ssl_first_name>
  <ssl_partial_auth_indicator>1</ssl_partial_auth_indicator>
 </txn>
```

Receive a partially approved transaction response

If the balance on a card is not enough to cover for the entire purchase, the point of sale application must be able to read the additional returned fields and collect balance that remains by other means. In this case, the amount requested was **10.10** and only **10.05** was approved. The integrated application must display the balance left to pay as **0.05**.

```
<txn>
   <ssl_card_number>41********1111</ssl_card_number>
   <ssl_exp_date>1212</ssl_exp_date>
   <ssl_amount>10.05</ssl_amount>
   <ssl_requested_amount>10.10</ssl_requested_amount>
   <ssl_balance_due>0.05</ssl_balance_due>
   <ssl_result>0</ssl_result>
   <ssl_result_message>PARTIAL APPROVAL</ssl_result_message>
   <ssl_txn_id>AA48439-9D9AFF76-AFEC-0B8D-DC7F-43A5F97BF81B</ssl_txn_id>
   <ssl_approval_code>CVI788</ssl_approval_code>
   <ssl_cvv2_response />
   <ssl_avs_response />
   <ssl_account_balance>0.00</ssl_account_balance>
   <ssl_txn_time>10/25/2011 10:49:52 PM</ssl_txn_time>
</txn>
```

If the consumers indicate that they do not wish to continue with the additional tender type, the point of sale application must send a reversal to cancel this payment and reestablish the balance back to the card. A reversal can be achieved by sending a delete in terminal-based terminals or void in host-based terminals.

```
<txn>
  <ssl_merchant_id>My_Merchant_ID</ssl_merchant_id>
  <ssl_user_id>my_user_id</ssl_user_id>
  <ssl_pin>my_pin</ssl_pin>
  <ssl_test_mode>false</ssl_test_mode>
  <ssl_transaction_type>ccdelete</ssl_transaction_type>
  <ssl_txn_id>AA48439-9D9AFF76-AFEC-0B8D-DC7F-43A5F97BF81B</ssl_txn_id>
</txn>
```

Dynamic Currency Conversion (DCC) Transactions

Transaction Flow

1. Submit the initial submission request as shown in the API Reference Chapter for `ccsale`, `ccforce`, `ccauthonly` or `cccredit`.
2. VirtualMerchant will determine if the card is a foreign card and will return the rate, the markup percentage, the merchant amount and the card holder amount.
3. Based on the integration type and the result format specified, either a DCC decision page will appear, or the point of sale application has to submit a second request to complete the transaction based on the consumer's choice to accept or reject DCC.
    a. If you set the field `ssl_result_format` to HTML, VirtualMerchant will handle transactions submitted to `process.do`. The customers will be taken to a DCC decision page, where they will need to decide what currency they want to process the transaction in. Once this is selected, the transaction will complete as normal.
    b. For `processxml.do` and `process.do` (if `ssl_result_format` field is set to ASCII), it is the developer's responsibility to provide the decision request by following up with a second request and providing a transaction ID along with DCC option (Y or N).
4. Receive a response as shown in the API Reference Chapter for a `ccsale`, `ccforce, ccauthonly` or `cccredit`.
5. If the consumer selects a foreign currency, the transaction will process as a DCC transaction and the receipt will reflect the exchange rate including fees, the U.S. dollar amount and the foreign transaction amount.

    NOTE: There is a 15-minute window to return a response.

Transaction Examples

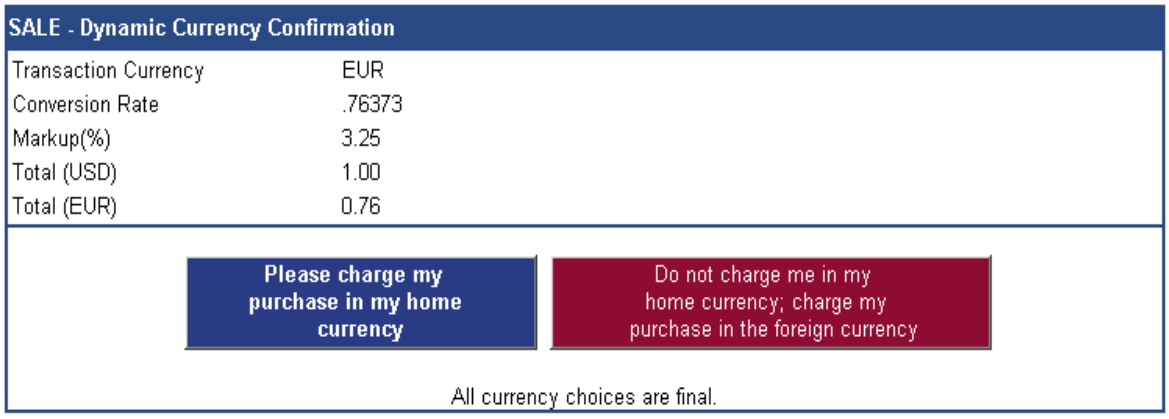
process.do

Send a ccsale request

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00"> <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <br/>
   <input type="hidden" name="ssl_result_format" value="HTML">
   <input type="hidden" name="ssl_receipt_decl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_decl_get_url"
   value="http://www.website.com/decline.asp">
   <input type="hidden" name="ssl_receipt_apprvl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_apprvl_get_url"
   value="http://www.website.com/approval.asp">
   <input type="submit" value="Continue">
</form>
```

Receive a DCC decision Page

The customer must select one of the buttons to continue to process the transaction.



processxml.do

Send a ccsale request

```
<txn>
 <ssl_merchant_id> my_virtualmerchant_id</ssl_merchant_id>
 <ssl_user_id>my_user_id</ssl_user_id>
 <ssl_pin>my_pin</ssl_pin>
 <ssl_test_mode>false</ssl_test_mode>
 <ssl_transaction_type>ccsale</ssl_transaction_type>
 <ssl_card_number>5000000000000002</ssl_card_number>
 <ssl_exp_date>1215</ssl_exp_date>
 <ssl_amount>1.00</ssl_amount>
</txn>
```

Receive a DCC decision response

```
<txn>
<id>ekU9j0L0iFO9m9FELAqK8E6</id>
<ssl_txn_currency_code>EUR</ssl_txn_currency_code>
<ssl_markup>3.25</ssl_markup>
<ssl_conversion_rate>.76373</ssl_conversion_rate>
<ssl_amount>1.00</ssl_amount>
<ssl_cardholder_amount>0.76</ssl_cardholder_amount> <dccoption>
<option label="Please charge my purchase in my home currency">Y</option>
<option label="Do not charge me in my home currency; charge my purchase
in US dollars">N</option>
</dccoption>
</txn>
```

Consumer accepts DCC and second request sent requesting card to be charged in foreign currency

```
<txn>
<ID>ekU9j0L0iFO9m9FELAqK8E6</ID>
<dccoption>Y</dccoption>
</txn>
```

Receive a final response and print receipt

```
<txn>
<ssl_card_number>50********0002</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_amount>1.00</ssl_amount>
<ssl_cardholder_amount>0.76</ssl_cardholder_amount>
<ssl_cardholder_currency>EUR</ssl_cardholder_currency>
<ssl_conversion_rate>.76373</ssl_conversion_rate>
<ssl_markup>3.25</ssl_markup>
<ssl_invoice_number />
<ssl_result>0</ssl_result>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_txn_id>101641221295DB876-F2A9-7B6A-B173-2737983B7693</ssl_txn_id>
<ssl_approval_code>N05465</ssl_approval_code>
<ssl_txn_time>01/20/2011 01:05:44 PM</ssl_txn_time>
</txn>
```

Consumer declines DCC and second request sent requesting card to be charged in Developer currency

```
<txn>
<ID>iQ4AezhcjmJznh3BYLZ0-W9</ID>
<dccoption>N</dccoption>
</txn>
```

Receive a final response

```
<txn>
<ssl_card_number>50********0002</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_amount>1.00</ssl_amount>
<ssl_result>0</ssl_result>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_txn_id>101641221593ACBA6-BAFD-76B7-4948-B3DE68CFD0CC</ssl_txn_id>
<ssl_approval_code>CMC142</ssl_approval_code>
<ssl_account_balance>1.00</ssl_account_balance>
<ssl_txn_time>01/20/2011 01:07:23 PM</ssl_txn_time>
</txn>
```

DCC Receipt Requirements

DCC Receipt Requirements for VISA

All Environments (Face-to-Face, MO/TO, e-commerce)

- The price of the goods or services in the merchant's home currency, accompanied by the currency symbol or code next to the amount.
- The total price in the transaction currency accompanied by the words **Transaction Currency** and the currency symbol or code next to the amount.
- The exchange rate used to convert the total price from the merchant's home currency to the transaction currency.
- Any mark-up or commission included in the exchange rate as an amount or a percentage. This information may be shown as a separate line item or in the disclosure verbiage on the transaction receipt.
- An "**I accept"** check box that indicates the cardholder's acceptance of the currency conversion.
- A statement in an area easily seen by the cardholder that states that DCC is offered by the merchant.

DCC Receipt Requirements for MasterCard

All Environments (Face-to-Face, MO/TO, e-commerce)

- Pre-conversion currency and amount.
- Conversion rate or method.
- Post-conversion currency and amount.
- A prescribed statement, "I have chosen not to use the MasterCard currency conversion method and I will have no recourse against MasterCard concerning currency conversion or its disclosure."

Special Requirements for Internet, Cardholder Activated Terminals and ATMs

- Screen messages at unattended point of sale terminals must not require the cardholder to select **Yes** or **No** when choosing currency conversion. Indirect means, such as the colors red and green, must not be used to influence the cardholder's choice.
- At attended point of sale terminals that require the cardholder to choose between **Yes** and **No**, the merchant must verbally explain the offer to the cardholder before presenting it on the point of sale terminal.

Receipt Example

Merchant Copy	Customer Copy
Header1 FRIENDLY TERMINAL XXXXXXXXXXXXXXXXXXXX Date: 11/04/2010 11:07:30 AM CREDIT CARD SALE CARD DATA: **********0002 S TRANSACTION CURRENCY: JPY CONVERSION RATE: 85.05321 CONVERSION MARKUP: 3.25% MERCHANT AMOUNT: $2.00 USD TRANSACTION AMOUNT: ¥170.00 JPY APPROVAL CD: N19586 CLERK ID: my_merchant_id RECORD #: 001 I HAVE BEEN OFFERED THE CHOICE OF CURRENCIES FOR PAYMENT INCLUDING THE MERCHANT'S LOCAL CURRENCY. MY DECISION TO ACCEPT CURRENCY CONVERSION ON THIS TRANSACTION IS FINAL. I ACCEPT THE RATE OF CONVERSION,(INCLUSIVE OF CONVERSION FEE 3.25%), FINAL AMOUNT, AND THAT THE FINAL SETTLED TRANSACTION CURRENCY IS {JPY}. I UNDERSTAND THAT VISA HAS A CURRENCY CONVERSION PROCESS, AND THAT I HAVE CHOSEN NOT TO USE THE VISA CURRENCY CONVERSION PROCESS. CURRENCY CONVERSION IS CONDUCTED BY THE MERCHANT AND IS NOT ASSOCIATED WITH OR ENDORSED BY VISA. I AGREE TO THE TRANSACTION RECEIPT INFORMATION BY MARKING THE ACCEPT BOX BELOW. [ ] I ACCEPT X_____ JPY STANDARD VI Trailer1 Merchant Copy	Header1 FRIENDLY TERMINAL XXXXXXXXXXXXXXXXXXXX Date: 11/04/2010 11:07:30 AM CREDIT CARD SALE CARD DATA: **********0002 S TRANSACTION CURRENCY: JPY CONVERSION RATE: 85.05321 CONVERSION MARKUP: 3.25% MERCHANT AMOUNT: $2.00 USD TRANSACTION AMOUNT: ¥170.00 JPY APPROVAL CD: N19586 CLERK ID: my_merchant_id RECORD #: 001 I HAVE BEEN OFFERED THE CHOICE OF CURRENCIES FOR PAYMENT INCLUDING THE MERCHANT'S LOCAL CURRENCY. MY DECISION TO ACCEPT CURRENCY CONVERSION ON THIS TRANSACTION IS FINAL. I ACCEPT THE RATE OF CONVERSION,(INCLUSIVE OF CONVERSION FEE 3.25%), FINAL AMOUNT, AND THAT THE FINAL SETTLED TRANSACTION CURRENCY IS {JPY}. I UNDERSTAND THAT VISA HAS A CURRENCY CONVERSION PROCESS, AND THAT I HAVE CHOSEN NOT TO USE THE VISA CURRENCY CONVERSION PROCESS. CURRENCY CONVERSION IS CONDUCTED BY THE MERCHANT AND IS NOT ASSOCIATED WITH OR ENDORSED BY VISA. Trailer1 Customer Copy

3D Secure Transactions

Transaction Flow

VirtualMerchant uses the Elavon eMPI engine to allow processing of 3D secure transactions. The eMPI is available at the following locations:

https://testempi.internetsecure.com/3DSecure
https://empi.internetsecure.com/3DSecure

If the merchant uses `process.do` to integrate to VirtualMerchant, there is no additional work required to utilize the eMPI, as the authentication piece is built in.

If the merchant uses `processxml.do` to integrate to VirtualMerchant, they may integrate to the eMPI to authenticate to 3D Secure.

The process of authentication is to retrieve and pass the following tags and values to the VirtualMerchant application in `ccsale` or `ccauthonly` requests:

`ssl_eci_ind`	E-commerce indicator or ECI Values. (fully authenticated) - There is a liability shift and the merchant is protected from chargeback. (VbV has been attempted) - There is a liability shift and the merchant is protected from chargeback. (Non-VbV transaction) – The merchant is no longer protected from chargeback.
`ssl_3dsecure_value`	Cardholder Authentication Verification Value or CAVV
`ssl_xid`	Unique transaction identifier assigned by eMPI

1. Developer sends a Verify Card Enrollment Request to eMPI to check if the card is enrolled for 3DSecure. eMPI does a look up with the directory server.

2. If the card is enrolled, eMPI returns the issuer URL and Payer Authentication Request (PaReq) to the merchant (Verify Card Enrollment Response).
   This completes the first stage in the authentication process.

3. The merchant redirects the cardholder to the issuer website to complete the authentication process (Inline authentication Windows with or Without Frames are recommended. Pop-up windows are NOT allowed).

4. After the cardholder completes the authentication process, the issuer redirects the cardholder browser back to the merchant. The Payer Authentication Result (PaRes) is posted to the merchant website.
   This completes the second stage in the authentication process.

5. The merchant now sends a Verify PaRes Request to eMPI with the PaRes.

6. eMPI processes the PaRes and responds with the CAVV and ECI values.
This completes the third and final stage in the authentication process.

7. The merchant now has to include the CAVV and ECI values in the Transaction Request `ccsale` and `ccauthonly` sent to the VirtualMerchant Gateway, along with the unique transaction identifier assigned by eMPI.

NOTE: There is a leading (**0**) in the ECI value returned from the eMPI. It must be removed prior to sending it to VirtualMerchant.

Transaction Example

process.do

Send a simple ccsale request to an eCommerce/ Internet terminal with VBV enabled – No additional work is needed – authentication is automatic

```
<form
action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do
" method="POST">
   Your Total: $5.00 <br/>
   <input type="hidden" name="ssl_amount" value="5.00"> <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_email" value="test@test.com">
   <input type="text" name="ssl_card_number">
   <input type="text" name="ssl_exp_date" size="4">
   <input type="hidden" name="ssl_result_format" value="HTML">
   <input type="hidden" name="ssl_receipt_decl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_decl_get_url"
   value="http://www.website.com/decline.asp">
   <input type="hidden" name="ssl_receipt_apprvl_method" value="REDG">
   <input type="hidden" name="ssl_receipt_apprvl_get_url"
   value="http://www.website.com/approval.asp">
   <input type="submit" value="Continue">
</form>
```

Receive a response page with the ECI indicator of "Full authenticated" after consumer has been directed to the issuer website for verification – this is transparent to the merchant website

```
<form action="http://www.website.com/approval.asp" method="GET">
   <input type="hidden" name="ssl_result" value="0">
   <input type="hidden" name="ssl_result_message" value="APPROVAL">
   <input type="hidden" name="ssl_txn_id" value="99C7884A-EDB6-4256-BE69-
   4547B8859D5B">
   <input type="hidden" name="ssl_approval_code" value="N29032">
   <input type="hidden" name="ssl_avs_response" value="D">
   <input type="hidden" name="ssl_eci_ind" value="Fully Authenticated">
   <input type="hidden" name=" ssl_cvv2_response" value="P">
   <input type="hidden" name="ssl_transaction_type" value="ccsale">
   <input type="hidden" name="ssl_invoice_number" value="123-ABC">
   <input type="hidden" name="ssl_amount" value="5.00">
   <input type="hidden" name="ssl_email" value=" test@test.com">
   <br>
   <input type="submit" value="Continue" class="smallbutton">
</form>
```

processxml.do

Verify card enrollment request

```
<request id="FA03303B8164AB5517C179F3D6D6">
   <version>1.0.0</version>
   <merchantId>EMPI-00001</merchantId>
   <applicationKey>8c7f5817-6ebe-4c6a-8757-8294a1de24d0</applicationKey>
   <verifyEnrollmentRequest>
   <accountData>
   <accountId>4000000000000001</accountId>
   <expiryYear>2011</expiryYear>
   <expiryMonth>03</expiryMonth>
   </accountData>
   <browser>
   <deviceCategory>0</deviceCategory>
   <accept>text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.
   8</accept>
   <userAgent>Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US;
rv:1.9.2.15) Gecko/20110303 Firefox/3.6.15 GTB7.1</userAgent>
   </browser>
   <purchaseDate>2011-03-09T00:00:00-05:00</purchaseDate>
   <purchaseAmount>100</purchaseAmount>
   <purchaseCurrency>124</purchaseCurrency>
   <orderDescription>test transaction</orderDescription>
   </verifyEnrollmentRequest>
</request>
```

Verify Card Enrollment Response (if card is enrolled and no errors were present)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response id="FA03303B8164AB5517C179F3D6D6">
   <xid>14FC923865.3B54A8-T1</xid>
   <verifyEnrollmentResponse>
   <inCardRange>true</inCardRange>
   <enrolled>Y</enrolled>
   <acsUrl>https://secure.issuerwebsite.com/Visaormastercard.jsp</acsUrl>
   <paReq>eJxdUu9PwjAQ/VeWfR/t5lQgtxIUf5AIiiBfSelOmLIO2iKMv952DkGTNrn3erl
   3967Q2ecr7wuVzgqZ+GGD+h5KUaSZXCT+2+Q+aPqeNlymfFVITPwStd9hMFkqxN4YxVYhg
   wFqzRfoZWnir7nCzewyovP35pwGSOM4iOcRDZo4j4Movb6KQpFehFctn8FL9xU3DGp9ZuU
   bEZAjtHWVWHJpGHCxuekP2dPzbfdpNu0DqQnIUfV7bNqnNATyA0DyHJlBbTzLVQBEsZVGl
   awZUyBHAFu1Yktj1m1CdrtdI5MGlUSjq7EaosiBuBQgp0Zeti7StuQ+S9lg8vowOnyWg8M
   iGo7jw+jjjY4mCzPt3SVAXAak3CCLaBjSC9ryKG1XB0jFA89dLyx0TB3D2kl0zx7OCbCtK
   buhkoVRbEc5IsD92i7IZlgDf2NIUYsfK4ziUnNhrLFW3NFATsPcPjqPhbH2DXvd3eD/HSW
   J87xKcFKZtc+NVGk5AMSVIPVCSf0jbPTnp3wDnkXPQA==</paReq>
   <deviceCategory>0</deviceCategory>
   </verifyEnrollmentResponse>
</response>
```

Sample Form to Redirect Customer to Issuer Website

NOTE: Do not use the GET method.

```
<Form action="https://secure.issuerwebsite.com/Visaormastercard.jsp"
method=POST>
   <input type="hidden" id="PaReq" name="PaReq"
   value="eJxdUu9PwjAQ/VeWfR/t5lQgtxIUf5AIiiBfSelOmLIO2iKMv952DkGTNrn3erl
   3967Q2ecr7wuVzgqZ+GGD+h5KUaSZXCT+2+Q+aPqeNlymfFVITPwStd9hMFkqxN4YxVYhg
   wFqzRfoZWnir7nCzewyovP35pwGSOM4iOcRDZo4j4Movb6KQpFehFctn8FL9xU3DGp9ZuU
   bEZAjtHWVWHJpGHCxuekP2dPzbfdpNu0DqQnIUfV7bNqnNATyA0DyHJlBbTzLVQBEsZVGl
   awZUyBHAFu1Yktj1m1CdrtdI5MGlUSjq7EaosiBuBQgp0Zeti7StuQ+S9lg8vowOnyWg8M
   iGo7jw+jjjY4mCzPt3SVAXAak3CCLaBjSC9ryKG1XB0jFA89dLyx0TB3D2kl0zx7OCbCtK
   buhkoVRbEc5IsD92i7IZlgDf2NIUYsfK4ziUnNhrLFW3NFATsPcPjqPhbH2DXvd3eD/HSW
   J87xKcFKZtc+NVGk5AMSVIPVCSf0jbPTnp3wDnkXPQA==">
   <input type="hidden" id="TermUrl" name="TermUrl"
   value="https://www.merchantwebsite.com/3DSReturn.jsp">
   <input type="hidden" id="MD" name="MD"
   value="FA03303B8164AB5517C179F3D6D6">
   <input type="submit" name="Proceed to Issuer Website">
/form>
```

After authentication, the issuer will redirect the customer to the merchant website and post the Payer Authentication Result (PaRes) and Merchant Data (MD).

```
<form method="POST"
   action="https://www.merchantwebsite.com/3DSReturn.jsp">
   <input type="hidden" name="PaRes"
   value="eJzFWFmPo8qSfu9f0er7aFWzYzhyl5QsxmCDAbO/HLGZfTGLMfz6wVVd3XV7jkZ
   z52Us+tBgeTr1Ma8NhEcp6+G011zO0rQjPxFSUOtlWBFltFdKF7Nmz+DNtaHffLAP+aQLe
   9i9MJxnE3BYmQ/OAKd3u7LBCJZi8sV0SENFNsB05nhYNtzpbPC4/dQtf+">
   <input type="hidden" name="MD" value="FA03303B8164AB5517C179F3D6D6">
   <input type="submit" value="Authenticate Cardholder">
</form>
```

Verify PaRes Request

The merchant must now post the PaRes to eMPI to obtain the decrypted ECI and 3DSecure value (if applicable).

```
<request id="FA03303B8164AB5517C179F3D6D6">
   <version>1.0.0</version>
   <merchantId>EMPI-00001</merchantId>
   <applicationKey>8c7f5817-6ebe-4c6a-8757-8294a1de24d0</applicationKey>
   <xid>14FC923865.3B54A8-T1</xid>
   <validateParesRequest>
   <paRes>eJzFWFmPo8qSfu9f0er7aFWzYzhyl5QsxmCDAbO/HLGZfTGLMfz6wVVd3XV7jkZ
   z52Us+tBgeTr1Ma8NhEcp6+G011zO0rQjPxFSUOtlWBFltFdKF7Nmz+DNtaHffLAP+aQLe
   9i9MJxnE3BYmQ/OAKd3u7LBCJZi8sV0SENFNsB05nhYNtzpbPC4/dQtf+">
   </paRes>
   </validateParesRequest>
</request>
```

Verify PaRes Response

eMPI will decrypt the PaRes, validate the message format, verify the digital signature of the transaction, and respond with the CAVV and ECI values.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response id="FA03303B8164AB5517C179F3D6D6">
    <xid>14FC923865.3B54A8-T1</xid>
    <validateParesResponse>
    <status>Y</status>
    <cavv>MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=</cavv>
    <eci>05</eci>
    <cavvAlgorithm>2</cavvAlgorithm>
    </validateParesResponse>
</response>
```

Send a ccsale request with ssl_eci_ind, ssl_3dsecure_value and ssl_xid values obtained from the previous steps

```
<txn> <ssl_merchant_id>my_virtualmerchant_id</ssl_merchant_id>
   <ssl_user_id>my_user_id</ssl_user_id>
   <ssl_pin>my_pin</ssl_pin>
   <ssl_test_mode>false</ssl_test_mode>
   <ssl_transaction_type>ccsale</ssl_transaction_type>
   <ssl_card_number>4111111111111111</ssl_card_number>
   <ssl_exp_date>1215</ssl_exp_date>
   <ssl_amount>12000.00</ssl_amount>
   <ssl_first_name>john</ssl_first_name>
   <ssl_last_name>Doe</ssl_last_name>
   <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
   <ssl_cvv2cvc2>789</ssl_cvv2cvc2>
   <ssl_company>01</ssl_company>
   <ssl_description>VBV Transaction</ssl_description>
   <ssl_eci_ind>5</ssl_eci_ind>
   <ssl_3dsecure_value> MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=</ssl_3dsecure_value>
   <ssl_xid>14FC923865.3B54A8-T1</ssl_xid>
 </txn>
```

Receive a ccsale response (transaction will show in batch with correct ECI indicator)

```
<txn>
   <ssl_card_number>41********1111</ssl_card_number>
   <ssl_exp_date>1215</ssl_exp_date>
   <ssl_amount>1.00</ssl_amount>
   <ssl_company>01</ssl_company>
   <ssl_first_name>john</ssl_first_name>
   <ssl_last_name>Doe</ssl_last_name>
   <ssl_result>0</ssl_result>
   <ssl_result_message>APPROVAL</ssl_result_message>
   <ssl_txn_id>AA48439-14AE8D51-2A60-DFA5-15A2-BD02D2FB08A5</ssl_txn_id>
   <ssl_approval_code>CVI127</ssl_approval_code>
   <ssl_cvv2_response>M</ssl_cvv2_response>
   <ssl_avs_response />
   <ssl_account_balance>1.00</ssl_account_balance>
   <ssl_txn_time>10/04/2011 10:09:11 AM</ssl_txn_time>
</txn>
```

Recurring and Installment Transactions

Transaction Flow

1. Submit a request to add a recurring or installment transaction using HTTPS, either by the HTTP GET or POST with the values shown in the API Reference Chapter in this guide.

   Shown below are the key value pairs from the header by themselves for adding a recurring transaction.

   ```
   ssl_merchant_id=xxxxxx
   ssl_user_id=xxxxxxx
   ssl_pin=xxxxxx
   ssl_show_form=false
   ssl_transaction_type=ccaddrecurring
   ssl_card_number=5000000000000003
   ssl_exp_date=1208
   ssl_amount=1.00
   ssl_billing_cycle=SEMESTER
   ssl_next_payment_date=09/02/2011
   ssl_skip_payment=Y
   ssl_total_installments=4
   ssl_avs_zip=70004
   ssl_invoice_number=1111
   ssl_customer_code=4444
   ssl_first_name=John
   ssl_last_name=Doe
   ```

2. When VirtualMerchant receives this post, it starts to parse the data to look for a few key fields first. It validates the `ssl_merchant_id`, `ssl_user_id`, and `ssl_pin` first to authenticate the user, then proceeds by validating the remaining content of the POST.

   - If the supplied information is invalid, an error is returned that states the reason and message for error
   - If the data is valid, VirtualMerchant stores the data in the recurring batch

NOTES:

- You can indicate in the error URL field where you would like VirtualMerchant to send all the errors that are encountered. Any response that is not approved or declined will be sent to the URL specified.
- By specifying the http://www.url.com/cgi-bin/testtran.cgi URL in the `ssl_error_url` field for the redirect for the transaction above, the following values (in the example below the PIN is invalid) are returned if the request is invalid:

  `errorCode`=4015

  `errorName`= PIN Invalid

  `errorMessage`= The PIN supplied in the authorization request is invalid

3. VirtualMerchant then returns a response to the POST. Shown below are the key value pairs returned when successfully adding a recurring transaction.

```
ssl_start_payment_date=12/12/2011
ssl_transaction_type=CCADDRECURRING
ssl_card_number=50**********3003
ssl_exp_date=1212
ssl_amount=10.00
ssl_next_payment_date=12/12/2011
ssl_billing_cycle=SEMESTER
ssl_result_message=SUCCESS
ssl_recurring_id=AA4844B-6345A73B-296A-03BB-226B-01A66829FA9F
ssl_number_of_payments=0
ssl_skip_payment=N
ssl_recurring_batch_count=6
```

NOTES:

- By specifying the http://www.url.com/cgi-bin/testtran.cgi URL in the `ssl_apprvl_get_url` field for the redirect for the transaction above, the following values are returned for the successful transaction.
- Based on the billing cycle and the start date supplied, the recurring transaction will run automatically in the system without further action from the merchant.

Transaction Examples

process.do

Example 1

In this example, the HTML code demonstrates the initiation of a minimal recurring transaction in which VirtualMerchant gathers the entire customer's billing information to automatically charge the customer $9.95 on a monthly basis. When customers purchase the initial product or service from a website, they can indicate their agreement to automatically renew their "subscription" and to process payment to their credit card. The VirtualMerchant Gateway will automatically add this customer to the recurring Billing database and run the payment every month.

Send a ccaddrecurring request

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type"
   value="ccaddrecurring">
   <input type="hidden" name="ssl_show_form" value="true">
   <input type="hidden" name="ssl_amount" value="9.95">
   <input type="hidden" name="ssl_billing" value="MONTHLY">
   <input type="submit" value="Subscribe">
</form>
```

Response

```
<form action="http://www.website.com/approval.asp" method="GET">
   <input type="hidden" name="ssl_result" value="0">
   <input type="hidden" name="ssl_start_payment_date" value="01/01/2012">
   <input type="hidden" name="ssl_recurring_id" value="AA484C3-B08B6F1B-
   4765-A1FF-C0BC-5722F21A0EB6">
   <input type="hidden" name="ssl_result_message" value="SUCCESS">
   <input type="hidden" name="ssl_card_number" value="50**********3003">
   <input type="hidden" name="ssl_exp_date" value="0212">
   <input type="hidden" name="ssl_amount" value="9.95">
   input type="hidden" name="ssl_next_payment_date" value="01/01/2012">
   <input type="hidden" name="ssl_billing" value="MONTHLY">
   <input type="hidden" name="ssl_next_installment" value="0">
   <input type="hidden" name="ssl_recurring_batch_count" value="63">
   <input type="hidden" name="ssl_skip_payment" value="NO">
   <input type="submit" value="Continue" class="smallbutton">
</form>
```

Example 2

The following HTML code demonstrates a basic form that collects and passes the minimum required data to setup an installment transaction twice a month for 10 total installments, starting from 01/01/2012 and processing on the 1st and 15th of every month. This code creates a form that displays the customer's payment and asks for their credit card number and expiration date, with a button labeled **Bill Me**. After the user enters the information and clicks the button, VirtualMerchant will automatically add this customer to the recurring billing database and run the payment twice a month for 10 consecutive payments. The user is then taken directly to a response page.

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   Your Semi-Monthly payment: $75.00 <br/>
   <input type="hidden" name="ssl_amount" value="75.00">
   <br/>
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="ccaddinstall">
   <input type="hidden" name="ssl_show_form" value="false">
   Credit Card Number: <input type="text" name="ssl_card_number"> <br/>
   Expiration Date (MMYY): <input type="text" name="ssl_exp_date"
   size="4"> <br/>
   <input type="hidden" name="ssl_billing" value="SEMIMONTHLY">
   <input type="text" name="ssl_next_payment_date" value="01/01/2012">
   <input type="text" name="ssl_bill_on_half" value="1">
   <input type="text" name="ssl_total_installments" value="10">
   <input type="submit" value="Bill Me">
</form>
```

Example 3

The following HTML code demonstrates a basic form that collects and passes the minimum required data to update a recurring transaction and set the billing payment to Suspended. The recurring ID obtained from the original transaction must be passed.

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type"
   value="ccupdaterecurring">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_recurring_id" value="AA484C3-B08B6F1B-
   4765-A1FF-C0BC-5722F21A0EB6">
   <input type="hidden" name="ssl_billing" value="SUSPENDED">
</form>
```

Example 4

The following HTML code demonstrates a basic form that collects and passes the required data to send a Sale outside of the billing cycle. The recurring ID obtained from the original transaction must be passed.

```
<form
   action="https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process
   .do" method="POST">
   <input type="hidden" name="ssl_merchant_id"
   value="my_virtualmerchant_id">
   <input type="hidden" name="ssl_user_id" value="my_user_id">
   <input type="hidden" name="ssl_pin" value="my_pin">
   <input type="hidden" name="ssl_transaction_type" value="
   ccrecurringsale">
   <input type="hidden" name="ssl_show_form" value="false">
   <input type="hidden" name="ssl_recurring_id" value="AA484C3-B08B6F1B-
   4765-A1FF-C0BC-5722F21A0EB6">
</form>
```

processxml.do

The following XML code example demonstrates an example of a basic transaction request and response.

Example 1
ccaddrecurring request

```
<txn>
   <ssl_transaction_type>ccaddrecurring</ssl_transaction_type>
   <ssl_card_number>5000300020003003</ssl_card_number>
   <ssl_exp_date>1212</ssl_exp_date>
   <ssl_amount>10.36</ssl_amount>
   <ssl_billing_cycle>MONTHLY</ssl_billing_cycle>
   <ssl_next_payment_date>01/31/2012</ssl_next_payment_date>
   <ssl_end_of_month>Y</ssl_end_of_month>
   <ssl_invoice_number>1111</ssl_invoice_number>
   <ssl_merchant_ID>my_merchant_id</ssl_merchant_ID>
   <ssl_user_id>my_user_id</ssl_user_id>
   <ssl_pin>my_pin</ssl_pin>
   <ssl_test_mode>False</ssl_test_mode>
</txn>
```

ccaddrecurring response

```
<txn>
   <ssl_start_payment_date>01/31/2012</ssl_start_payment_date>
   <ssl_transaction_type>CCADDRECURRING</ssl_transaction_type>
   <ssl_card_number>50**********3003</ssl_card_number>
   <ssl_exp_date>1212</ssl_exp_date>
   <ssl_amount>10.36</ssl_amount>
   <ssl_next_payment_date>01/31/2012</ssl_next_payment_date>
   <ssl_billing_cycle>MONTHLY</ssl_billing_cycle>
   <ssl_result_message>SUCCESS</ssl_result_message>
   <ssl_recurring_id>AA484C3-8E5D1201-A05E-824D-9DAD-
       E3534E83F078</ssl_recurring_id>
   <ssl_number_of_payments>0</ssl_number_of_payments>
   <ssl_skip_payment>N</ssl_skip_payment>
   <ssl_recurring_batch_count>65</ssl_recurring_batch_count>
</txn>
```

Chapter 7 API Reference

The **API Reference** section outlines the request and result fields for processing transactions with VirtualMerchant Gateway using XML thru `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/processxml.do` or HTML Key / Value Pair thru https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do.

Note that only the minimum required fields, as well as recommended fields are shown in this section. Additional fields may be passed at transaction run time. Required fields are based on the merchant account configuration within VirtualMerchant. Fields including information such as CVV/CVC/CID, AVS and custom defined fields may be required if the account is configured for these options. For best possible transaction rates, Elavon recommends passing as much information as possible. For an extensive list of available HTML/XML value pair input fields, refer to the Supported Transaction Input Fields section.

This section provides information on the following:

- Credit card transactions
- Recurring transactions
- Installment transactions
- Debit card transactions
- EBT transactions
- Gift card (EGC) transactions
- Electronic check transactions
- PINless debit purchase transactions

Credit Card Transactions

This message format is for whole track, track 1 or track 2 magnetic stripe read credit card data, or key-entered credit card data available for all supported market segments. Magnetic stripe data cannot be sent in the Mail Order/Telephone Order and E-commerce environments.

Sale (ccsale)

The `ccsale` is a transaction in which an authorization is obtained and the transaction is entered into the unsettled batch. This transaction is used to obtain real-time authorization for a credit card Sale transaction.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Sale (ccsale)**
`ssl_amount`	Transaction Sale Amount. Number with 2 decimal places. For example: 1.00. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transaction.
`ssl_card_present`	Recommended to be passed on hand-keyed transactions to indicate if the card was present at the time of the transaction. Valid values: Y or N. Example: Card present of Y in hand-keyed retail environment.
`ssl_exp_date`	Credit Card Expiry Date as it appears on credit card. **Required** for hand-keyed transaction.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_partial_auth_indicator`	The partial indicator flag must be sent to indicate that the application supports partial approval. **Optional**. Valid values: 1.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`) Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred, this field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The balance left in card, which is always “0.00” for a partially authorized transaction.
ssl_amount	The transaction authorized or approved amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_avs_response	The Address Verification Response Code (refer to Authorization Response Codes section for more information).
ssl_balance_due	The remaining balance due in the ssl_balance_due field. This is the difference of the amount requested versus the amount authorized that the merchant has to collect from the consumer on partial approval sonly.
ssl_cardholder_amount	Only returned on DCC transactions.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_conversion_rate	Only returned on DCC transactions.
ssl_cvv2_response	The Card Verification Code Response (refer to Authorization Response Codes section for more information).
ssl_eci_ind	Only returned on 3D Secure transactions. Valid values: Fully Authenticated Authentication Attempted
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_requested_amount	The amount originally requested on partial approval sonly.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_result_message	The transaction result message. Example: APPROVAL, PARTIAL APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Auth Only (ccauthonly)

Use the `ccauthonly` transaction to acquire an approval code for a Force transaction. This transaction will only reduce the cardholder's limit to buy. It does not place a transaction in the open batch.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Auth Only (ccauthonly)**
`ssl_amount`	Transaction Auth Amount. Number with 2 decimal places. For example: 1.00. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_card_present`	Recommended to be passed on hand-keyed transactions to indicate if the card was present at the time of the transaction. Valid values: Y or N. Example: Card present of Y in hand-keyed retail environment.
`ssl_exp_date`	Credit Card Expiry Date as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_partial_auth_indicator`	The partial indicator flag. Must be sent to indicate that the application supports partial approval. **Optional**. Valid values: 1.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`) Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The balance left in card, which is always "0.00" for a partially authorized transaction.
ssl_amount	The transaction authorized or approved amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_avs_response	The Address Verification Response Code. Refer to Authorization Response Codes section for more information.
ssl_balance_due	The remaining balance due in the ssl_balance_due field. This is the difference of the amount requested versus the amount authorized that the merchant has to collect from the consumer on partial approvals only.
ssl_cardholder_amount	Only Returned on DCC transactions.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_conversion_rate	Only Returned on DCC transactions.
ssl_cvv2_response	The Card Verification Code Response. Refer to Authorization Response Codes section for more information.
ssl_eci_ind	Only Returned on 3D Secure Transactions. Valid values: Fully Authenticated. Authentication Attempted
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_requested_amount	The amount originally requested on partial approvals only.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_result_message	The transaction result message. Example: APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

AVS Only (ccavsonly)

The `ccavsonly` transaction is used to verify the credit card account for AVS data. An AVS code is returned to indicate if the AVS data passed originally was correct and matched the cardholder statement billing address.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**AVS Only (ccavsonly)**
`ssl_avs_address`	Customer's address used to process AVS.
`ssl_avs_zip`	Customer's `ZIP` code used to process AVS.
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_card_present`	Recommended to be passed on hand-keyed transactions to indicate if the card was present at the time of the transaction. Valid values: Y or N. Example: Card present of Y in hand-keyed retail environment.
`ssl_exp_date`	Credit Card Expiry Date as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`) Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_approval_code	The transaction approval code.
ssl_avs_response	The Address Verification Response Code. Refer to Authorization Response Codes section for more information.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_cvv2_response	The Card Verification Code Response. Refer to Authorization Response Codes section for more information.
ssl_exp_date	Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This number is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Credit (cccredit)

The `cccredit` transaction is used to refund the amount of a previous purchase to the cardholder's credit card available balance.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Credit (cccredit).**
`ssl_amount`	Amount to be refunded. Number with 2 decimal places. For example: 1.00. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_card_present`	Recommended to be passed on hand-keyed transactions to indicate if the card was present at the time of the transaction. Valid values: Y or N. Example: Card present of Y in hand-keyed retail environment.
`ssl_exp_date`	Credit Card Expiry Date as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_avs_response	The Address Verification Response Code. Refer to Authorization Response Codes section for more information.
ssl_cardholder_amount	Only returned on DCC transactions.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_conversion_rate	Only returned on DCC transactions.
ssl_cvv2_response	The Card Verification Code Response. Refer to Authorization Response Codes section for more information.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Force (ccforce)

The `ccforce` is a transaction that places a previously authorized transaction into a current unsettled batch.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Force (ccforce).**
`ssl_amount`	Transaction amount. Number with 2 decimal places. For example: 1.00. **Required.**
`ssl_approval_code`	Previously received Authorization Approval Code. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_card_present`	Recommended to be passed on hand-keyed transaction to indicate if the card was present at the time of the transaction. Valid values: Y or N. Example: Card present of Y in hand-keyed retail environment.
`ssl_exp_date`	Credit Card Expiry Date as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_avs_response	The Address Verification Response Code. Refer to Authorization Response Codes section for more information.
ssl_cardholder_amount	Only returned on DCC transactions.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_conversion_rate	Only returned on DCC transactions.
ssl_cvv2_response	The Card Verification Code Response. Refer to Authorization Response Codes section for more information.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL. Refer to Authorization Response Codes for more information.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Balance Inquiry (ccbalinquiry)

The `ccbalinquiry` is a transaction that returns the balance of a pre-paid card to the merchant. This message format is for either a track 1 or a track 2 magnetic stripe read, or manually key entered pre-paid card.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Balance Inquiry (ccbalinquiry)**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_card_present`	Card present indicator. Preferred on hand-keyed transactions. Valid Values: Y or N.
`ssl_exp_date`	Credit Card Expiry date as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The account balance. Number with 2 decimal places.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_invoice_number	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Void (ccvoid)

The `ccvoid` is a transaction that removes a Sale, Credit or Force transaction from the open batch. No funds will be deposited into the merchant's bank account at settlement. The `ccvoid` command is typically used for same day returns or to correct cashier mistakes. This action can only be performed before the batch is settled. To perform a `ccvoid,` you must submit the transaction ID received from the original transaction. The `ssl_show_form` property does not apply on Void transactions.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Credit Void (ccvoid).**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_txn_id`	Unique identifier returned on the original transaction. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
`ssl_result_message`	The transaction result message. Example: APPROVAL.
`ssl_txn_time`	Date and time when the transaction was processed, Format: MM/DD/YYYY hh:mm:ss PM/AM. Eample: 03/18/2010 10:34:10 AM.
`ssl_txn_id`	The transaction identification number. This is a unique number used to identify the transaction.
`ssl_approval_code`	The transaction approval code.
`ssl_invoice_number`	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
`ssl_email`	Returned based on merchant setup.
`errorCode`	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
`errorName`	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
`errorMessage`	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.

Delete (ccdelete)

The `ccdelete` is a transaction that deletes and attempts a reversal on a Sale and Auth Only Credit transaction. A transaction that has been deleted from the batch cannot be recovered. This transaction type is typically used on terminal-based terminals. To perform a `ccdelete`, you must submit the transaction ID received from the original transaction. The `ssl_show_form` property does not apply on Delete transactions.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Credit Delete (ccdelete).**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.
`ssl_txn_id`	Unique identifier returned on the original transaction. **Required.**

OUTPUT FIELD NAME	DESCRIPTION
`errorCode`	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
`errorMessage`	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
`errorName`	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
`ssl_approval_code`	The transaction approval code.
`ssl_email`	Returned based on merchant setup.
`ssl_invoice_number`	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
`ssl_result_message`	The transaction result message. Example: APPROVAL.
`ssl_txn_id`	The transaction identification number. This is a unique number used to identify the transaction.
`ssl_txn_time`	Date and time when the transaction was processed, Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Signature (ccsignature)

The `ccsignature` is a transaction that adds signature data to a previously approved credit card transaction. To perform a `ccsignature`, you must submit the transaction ID received from the original transaction. The `ssl_show_form` property does not apply on adding a signature.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Credit Signature (ccsignature).**
`ssl_image_type`	Image format. **Required** Possible values, must be capital: GIF TIF JPG PNG
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_signature_image`	BASE 64 Encoded version of an IMAGE. **Required.**
`ssl_txn_id`	Unique identifier returned on the original transaction. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`errorCode`	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
`errorMessage`	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
`errorName`	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
`ssl_result`	Signature request Result code. A result of 0 indicates the signature was successfully uploaded and added; 1 indicates that the signature upload failed.
`ssl_result_message`	Signature upload Result message. A result of “SUCCESS” indicates the image was uploaded/ added successfully; “ERROR” indicates the image was not added/ imported successfully.
`ssl_user_id`	VirtualMerchant User ID.

Recurring Transactions

The recurring transaction message format allows you to set up recurring transactions in the system for all market segments. This message format is used to add, update, remove and submit recurring transactions. No track, track 1 or track 2 magnetic stripe read are allowed. Also, CVV data cannot be passed, as it should not be stored in the system.

Add Recurring Transaction (ccaddrecurring)

The `ccaddrecurring` is a transaction that adds a recurring record to VirtualMerchant recurring batch. Once added, the transaction will run automatically within the specified billing cycle on the scheduled payment day without the need to send it for authorization. Recurring transactions will run indefinitely, unless suspended by the user.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Add Recurring (ccaddrecurring)**
`ssl_amount`	Transaction amount. **Required.**
`ssl_billing_cycle`	Billing cycle. **Required.** Valid returned values, all caps and no hyphens: - DAILY - WEEKLY - BIWEEKLY - SEMIMONTHLY - MONTHLY - BIMONTHLY - QUARTERLY - SEMESTER - SEMIANNUALLY - ANNUALLY - SUSPENDED
`ssl_bill_on_half`	Half of the month or Semimonthly indicator. **Conditional**. Valid values are **1** and **2**: - 1 = the 1st and the 15th of the month - 2 = the 15th and the last day of the month
`ssl_card_number`	Card number. **Required.**
`ssl_end_of_month`	End of month indicator. **Conditional**. Valid values Y or N. Must be passed on an add or an update of a recurring/installment transaction to indicate if the transaction is to be processed on the last day of the month.
`ssl_exp_date`	Card expiry date. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_next_payment_date`	Next payment date; Format MM/DD/YYYY. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
ssl_show_form	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
ssl_skip_payment	Skip Payment field. **Optional.** Valid values: Y for YES or N for No. Defaulted to N.
ssl_user_id	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
ssl_amount	Recurring amount.
ssl_billing_cycle	Billing cycle.
ssl_card_number	Card number. Returned in hashed/masked format.
ssl_next_payment_date	Next payment due date.
ssl_recurring_batch_count	Current number of transactions sitting in the recurring batch after the recurring transaction has been added.
ssl_recurring_id	The ID number of the recurring record added returned on SUCCESS only.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents a successful transaction.
ssl_result_message	The transaction result message. A result of “SUCCESS” indicates the recurring was successfully added; “ERROR” indicates the recurring was not added successfully.
ssl_start_payment_date	Start payment date. Format MM/DD/YYYY. Date when the first payment started. If recently added, start date is same as next payment.

Update Recurring Transaction (ccupdaterecurring)

The `ccupdaterecurring` is a transaction that updates a recurring record in VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Update Recurring (ccupdaterecurring)**
`ssl_billing_cycle`	Billing cycle. **Optional.** Valid returned values, all caps and no hyphens: - DAILY - WEEKLY - BIWEEKLY - SEMIMONTHLY - MONTHLY - BIMONTHLY - QUARTERLY - SEMESTER - SEMIANNUALLY - ANNUALLY - SUSPENDED
`ssl_bill_on_half`	Half of the month or Semimonthly indicator. **Conditional**. Valid values are **1** and **2**: - 1 = the 1st and the 15th of the month - 2 = the 15th and the last day of the month
`ssl_end_of_month`	End of month indicator. **Conditional**. Valid values Y or N, must be passed on an add or an update of a recurring/installment transaction to indicate if the transaction is to be processed on the last day of the month.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_next_payment_date`	Next payment date. Format MM/DD/YYYY. **Optional.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_recurring_id`	The ID number of the recurring record to be updated. This value, was returned when the original recurring record was added. Alpha numeric. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
`ssl_skip_payment`	Skip Payment field. **Optional.** Valid values: Y for YES or N for No. Defaulted to N.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
ssl_amount	Recurring amount.
ssl_billing_cycle	Billing cycle.
ssl_card_number	Card number. Returned in hashed/masked format.
ssl_next_payment_date	Next payment due date.
ssl_number_of_payments	Current number of payments run so far. Numeric. Returned by VirtualMerchant. Represents the number of payments run on the system.
ssl_recurring_batch_count	Current number of transactions sitting in the recurring batch.
ssl_recurring_id	The ID number of the recurring record updated. Alphanumeric. Returned on SUCCESS only. This value is a unique tracking number that the application assigns internally to each recurring record in the database.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents a successful transaction.
ssl_result_message	The transaction result message. A result of “SUCCESS” indicates the recurring was successfully updated; “ERROR” indicates the recurring was not updated successfully.
ssl_start_payment_date	Start payment date. Format MM/DD/YYYY. Date when the first payment started. If recently added, start date is same as next payment.

Delete Recurring Transaction (ccdeleterecurring)

The `ccdeleterecurring` is a transaction that deletes a recurring record in VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Delete Recurring (ccdeleterecurring)**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_recurring_id`	The ID number of the recurring record to be updated. This value was returned when the original recurring record was added. Alphanumeric. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`ssl_recurring_batch_count`	Current number of transactions sitting in the recurring batch after the recurring transaction has been deleted.
`ssl_recurring_id`	The ID number of the recurring record deleted. Alphanumeric. Returned on SUCCESS only. If the recurring ID was successfully deleted from the database and is no longer showing in the current batch recurring. No Auth or automatic payment can be run on this ID.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents a successful transaction.
`ssl_result_message`	The transaction result message. A result of “SUCCESS” indicates the recurring was successfully deleted; “ERROR” indicates the recurring was not deleted successfully.

Submit Recurring Payment (ccrecurringsale)

The `ccrecurringsale` is a transaction that allows you to run a recurring payment outside of its billing cycle. This will increase the payment number.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Submit Recurring Payment (ccrecurringsale)**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_recurring_id`	The ID number of the recurring record to be updated. This value was returned when the original recurring record was added. Alphanumeric. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`ssl_amount`	The transaction authorized or approved amount. Returned based on merchant setup.
`ssl_approval_code`	The transaction approval code.
`ssl_billing_cycle`	Billing cycle.
`ssl_card_number`	The masked card number. Returned based on merchant setup.
`ssl_exp_date`	Returned based on merchant setup.
`ssl_invoice_number`	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
`ssl_next_payment_date`	Next payment due date.
`ssl_number_of_payments`	Current number of payments run so far. Numeric. Returned by VirtualMerchant. This number represents the number of payments run on the system.
`ssl_recurring_id`	The ID number of the recurring record; Alpha numeric; Returned on SUCCESS only.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents an Approved transaction. A response containing any other value for `ssl_result` represents a Declined transaction preventing it from being authorized.

OUTPUT FIELD NAME	DESCRIPTION
ssl_result_message	The transaction result message. Example: APPROVAL, PARTIAL APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 09/18/2011 10:34:10 AM.

Installment Transactions

An installment transaction message format allows you to set up installment transactions in the system for all market segments. This message format is used to add, update, remove and submit installment transactions. No track, track 1 or track 2 magnetic stripe read are allowed. Also, CVV data cannot be passed, as it should not be stored in the system.

Add Installment Transactions (ccaddinstall)

The `ccaddinstall` is a transaction that adds an installment record to VirtualMerchant recurring batch. Once added, the transaction will run automatically within the specified billing cycle on the scheduled payment day without the need to send it for authorization. Installment transactions will run for a fixed payment number, unless suspended by the user.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Add Installment (ccaddinstall)**
`ssl_amount`	Transaction amount. **Required.**
`ssl_billing_cycle`	Billing cycle. **Required.** Valid returned values, all caps and no hyphens: - DAILY - WEEKLY - BIWEEKLY - SEMIMONTHLY - MONTHLY - BIMONTHLY - QUARTERLY - SEMESTER - SEMIANNUALLY - ANNUALLY - SUSPENDED
`ssl_bill_on_half`	Half of the month or Semimonthly indicator. **Conditional**. Valid values are **1** and **2**: - 1 = the 1st and the 15th of the month - 2 = the 15th and the last day of the month
`ssl_card_number`	Card number. **Required.**
`ssl_end_of_month`	End of month indicator. **Conditional**. Valid values Y or N. Must be passed on an add or an update of a recurring/installment transaction to indicate if the transaction is to be processed on the last day of the month
`ssl_exp_date`	Card expiry date. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_next_payment_date`	Next payment date; Format MM/DD/YYYY. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
ssl_show_form	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
ssl_skip_payment	Skip Payment field. **Optional.** Valid values: Y for YES or N for No. Defaulted to N.
ssl_total_installments	Number of payments; Numeric. **Required.**
ssl_user_id	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
ssl_amount	Recurring amount.
ssl_billing_cycle	Billing cycle.
ssl_card_number	Card number. Returned in hashed/masked format.
ssl_installment_id	The ID number of the installment record added. Returned on SUCCESS only.
ssl_next_payment_date	Next payment due date.
ssl_recurring_batch_count	Current number of transactions sitting in the recurring batch after the installment transaction has been added.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents a successful transaction.
ssl_result_message	The transaction result message. A result of “SUCCESS” indicates the installment was successfully added; “ERROR” indicates the installment was not added successfully.
ssl_start_payment_date	Start payment date. Format MM/DD/YYYY. Date when the first payment started. If recently added, start date is same as next payment.
ssl_total_installments	Total number of payments. Numeric.

Update Installment Transactions (ccupdateinstall)

The `ccupdateinstall` is a transaction that updates an installment record in VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Update Installment (ccupdateinstall)**
`ssl_billing_cycle`	Billing cycle. **Optional.** Valid returned values, all caps and no hyphens: - DAILY - WEEKLY - BIWEEKLY - SEMIMONTHLY - MONTHLY - BIMONTHLY - QUARTERLY - SEMESTER - SEMIANNUALLY - ANNUALLY - SUSPENDED
`ssl_bill_on_half`	Half of the month or Semimonthly indicator. **Conditional**. Valid values are **1** and **2**: - 1 = the 1st and the 15th of the month - 2 = the 15th and the last day of the month
`ssl_end_of_month`	End of month indicator. **Conditional**. Valid values Y or N. Must be passed on an add or an update of a recurring/installment transaction to indicate if the transaction is to be processed on the last day of the month
`ssl_installment_id`	The ID number of the Installment record to be updated. Alphanumeric. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_next_payment_date`	Next payment date; Format MM/DD/YYYY. **Optional.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
`ssl_skip_payment`	Skip Payment field. **Optional.** Valid values: Y for YES or N for No. Defaulted to N
`ssl_total_installments`	Total number of payments; Numeric. **Optional.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
ssl_amount	Recurring amount.
ssl_billing_cycle	Billing cycle.
ssl_card_number	Card number. Returned in hashed/masked format.
ssl_installement_id	The ID number of the installment record updated. Returned on SUCCESS only.
ssl_next_payment_date	Next payment due date.
ssl_number_of_payments	Current number of payments run so far. Numeric. Returned by VirtualMerchant. This number represent the number of payments run on the system. It is less than or equal to the total installments setup originally in the system.
ssl_recurring_batch_count	Current number of transactions sitting in the recurring batch.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents a successful transaction.
ssl_result_message	The transaction result message. A result of “SUCCESS” indicates the installment was successfully updated; “ERROR” indicates the installment was not updated successfully.
ssl_start_payment_date	Start payment date. Format MM/DD/YYYY. Date when the first payment started. If recently added, start date is same as next payment.
ssl_total_installments	Total number of payments. Numeric.

Delete Installment Transactions (ccdeleteinstall)

The `ccdeleteinstall` is a transaction that deletes an installment record in VirtualMerchant.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Delete Installment (ccdeleteinstall)**
`ssl_installment_id`	The ID number of the recurring record to be deleted. Alphanumeric. **Required**.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant

OUTPUT FIELD NAME	DESCRIPTION
`ssl_installment_id`	The ID number of the installment record updated. Returned on SUCCESS, only if the installment record was deleted successfully.
`ssl_recurring_batch_count`	Current number of transactions sitting in the recurring batch after the installment transaction has been deleted.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents a successful transaction.
`ssl_result_message`	The transaction result message. A result of “SUCCESS” indicates the installment was successfully deleted; “ERROR” indicates the installment was not deleted successfully.

Submit Installment Payment (ccinstallsale)

The `ccinstallsale` is a transaction that allows you to run an installment payment outside of its billing cycle. This will increase the payment number.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Submit Installment Payment (ccinstallsale)**
`ssl_installment_id`	The ID number of the recurring record to be authorized. Alphanumeric. **Required**.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`ssl_amount`	The transaction authorized or approved amount. Returned based on merchant setup.
`ssl_approval_code`	The transaction approval code.
`ssl_billing_cycle`	Billing cycle.
`ssl_card_number`	The masked card number. Returned based on merchant setup.
`ssl_exp_date`	Returned based on merchant setup.
`ssl_installment_id`	The ID number of the installment record submitted. Returned on SUCCESS, only if the installment record was deleted successfully.
`ssl_invoice_number`	The invoice or ticket number sent originally on the request. Returned based on merchant setup.
`ssl_next_payment_date`	Next payment due date.
`ssl_number_of_payments`	Current number of payments run so far. Numeric. Returned by VirtualMerchant.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents an Approved transaction. A response containing any other value for `ssl_result` represents a Declined transaction preventing it from being authorized.
`ssl_result_message`	The transaction result message. Example: APPROVAL, PARTIAL APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
`ssl_start_payment_date`	Date when the first payment started. If recently added, start date is same as next payment.

OUTPUT FIELD NAME	DESCRIPTION
ssl_total_installments	Total number of payments. Numeric.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 09/18/2011 10:34:10 AM.

Debit Card Transactions

Debit transactions require integration to a PIN pad to retrieve the Pin Block and the Key serial number. The PIN pad must be injected with Elavon encryption keys. Debit transactions may only be performed in a Retail or "face to face" environment. Mail Order/Telephone Order and E-commerce businesses cannot perform online debit transactions.

A "Track II" card swipe is required. Manual entry is not allowed.

Debit Purchase (dbpurchase)

The `dbpurchase` causes the amount of the purchase to be deducted from the debit cardholder's checking or saving account, debiting the account immediately.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Debit Sale(dbpurchase).**
`ssl_account_type`	Account Type (0=checking, 1=saving). **Required.**
`ssl_amount`	Transaction base amount must be sent on request. This amount does not include the cash back amount, which must be sent separately. The system will then compute the total to send for authorization. For example: 1.00. **Required.**
`ssl_cashback_amount`	Cash back. The amount of cash back that the customer will receive. Must be a number with 2 decimal places, if sent. **Optional.**
`ssl_dukpt`	This is the value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored**. Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad Device. This value cannot be stored. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form. (Available only for `process.do`) Set to false otherwise.
`ssl_track_data`	The raw Track I or Track II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_type	Account Type (0=checking, 1=saving).
ssl_amount	Transaction total amount including surcharge or cash back. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_base_amount	Base amount. The amount sent originally on the request.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_cashback_amount	Cash back. The amount of cash back that the customer will receive.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction.
ssl_result_message	The transaction result message. For example: APPROVAL. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_surcharge_amount	Surcharge amount. The fee that a merchant can charge for transactions as a cost for doing business. It is configurable in the application.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. For example: 03/18/2010 10:34:10.AM.

Debit Return (dbreturn)

The `dbreturn` causes the amount of the transaction to be refunded back to the debit cardholder's checking or saving account. The balance is reflected in the account immediately. The Reference Number, Date and Time of Original Transaction should be passed.

NOTE: Pending host, this functionality is available in the application and currently pending the host for future implementation.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Debit Return(dbreturn)**
`ssl_account_type`	The debit Account Type (0=checking, 1=saving). **Required.**
`ssl_amount`	Transaction return Amount.
`ssl_dukpt`	The value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored under any circumstance. **Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_original_date`	Date of original transaction in MMDDYY format.
`ssl_original_time`	Time of original transaction in HHMMSS format.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored under any circumstances. **Required.**
`ssl_reference_number`	The Transaction Reference Number is returned in the authorization response message.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise
`ssl_track_data`	The raw Track I or Track II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_cashback_amount	Cash back. The amount of cash back that the customer will receive.
ssl_email	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction, which prevents it from being authorized.
ssl_result_message	The transaction result message. For example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Debit Inquiry (dbbainquiry)

The `dbbainquiry` returns the balance available in the cardholder's checking or saving account.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Debit Inquiry (dbbainquiry)**
`ssl_account_type`	Account Type (0=checking, 1=saving). **Required.**
`ssl_dukpt`	This is the value returned by the PIN pad device which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored**. Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise
`ssl_track_data`	The raw Track I or Track II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The balance on the card.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

EBT Transactions

EBT transactions require integration to a PIN pad to retrieve the Pin Block and the Key serial number. The PIN pad must be injected with Elavon encryption keys. EBT Transactions may only be performed in a Retail or "face-to-face" environment. Mail Order/Telephone Order and E-commerce businesses cannot perform EBT transactions. There are two types of EBT transactions: Food Stamp and Cash Benefits.

Food Stamp Purchase (fspurchase)

The `fspurchase` is a transaction in which an authorization is obtained on an EBT card. This message is for an EBT Food Stamp Card Purchase transaction, either magnetic stripe read (Track II) or manually entered. This transaction reduces the cardholder's limit to buy, and places the transaction into the open batch.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Food Stamp Purchase (fspurchase)**
`ssl_amount`	Transaction Amount, must be number with 2 decimal places. **Required.**
`ssl_dukpt`	This is the value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored. **Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw track I or II data only as read from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization, this is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Food Stamp Return (fsreturn)

The `fsreturn` is used to refund money to the cardholder. A Return transaction will increase the cardholder's limit to buy once the batch containing the return has been settled or closed. Use this transaction type to process a food stamp transaction to credit money back onto the EBT card. This transaction can be either magnetic stripe read (Track II) or manually entered. The Reference Number, Date and Time of Original Transaction are recommended to be passed.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Food Stamp return (fsreturn)**
`ssl_amount`	Transaction Amount. Must be number with 2 decimal places. **Required.**
`ssl_dukpt`	This is the value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored**. Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions, value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_original_date`	Date of original transaction in MMDDYY format.
`ssl_original_time`	Time of original transaction in HHMMSS format.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
`ssl_reference_number`	The Transaction Reference Number is returned in the authorization response message.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw Track I or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Food Stamp Inquiry (fsbainquiry)

This transaction allows the merchant to check the balance of a customer's account and to verify the amount of funds available. A Track II card swipe is required (no manual entry).

INPUT FIELD NAME XML/ HTML	DESCRIPTION
ssl_transaction_type	**Food Stamp Inquiry (fsbainquiry)**
ssl_account_type	Account Type (0=checking, 1=saving). **Required.**
ssl_dukpt	This is the value returned by the PIN Pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored**. Required.**
ssl_key_pointer	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. **Required.**
ssl_merchant_id	VirtualMerchant ID as provided by Elavon.
ssl_pin	VirtualMerchant PIN as configured within VirtualMerchant.
ssl_pin_block	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
ssl_show_form	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise
ssl_track_data	The raw Track I or Track II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required.**
ssl_user_id	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred, this field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL
ssl_txn_id	The transaction identification number. This field is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Food Stamp Force Purchase (fsforcepurchase)

This is the completion transaction for an EBT Food Stamp Purchase authorization obtained using phone. This is a manually entered card only. This transaction requires a 15-digit Voucher Clear Number from Merchant's EBT Food Stamp sales slip and the Voucher Clear Approval Code obtained previously using the phone. The PIN number is not prompted for on the Voucher Clear transactions.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Food Stamp Force Purchase (fsforcepurchase)**
`ssl_amount`	Transaction Amount. Must be number with 2 decimal places. **Required.**
`ssl_approval_code`	The Voucher Clear Approval Code obtained previously using the phone. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_exp_date`	Credit Card Expiry date as it appears on credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.
`ssl_voucher_number`	The 15-digit Voucher Clear Number from Merchant's EBT Food Stamp sales slip. **Required.**

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.
ssl_voucher_number	The Voucher Clear Number.

Food Stamp Force Return (fsforcereturn)

This message format is used to manually enter a Food Stamp Voucher Clear Return transaction. This is the completion transaction for an EBT Food Stamp Refund authorization obtained using the phone. This transaction requires a 15-digit Voucher Clear Number from Merchant's EBT Food Stamp sales slip and the Voucher Clear Approval Code obtained previously using the phone. The PIN number is not prompted for on the Voucher Clear transactions.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Food Stamp Force Return (fsforcereturn)**
`ssl_amount`	Transaction Amount. Must be number with 2 decimal places. **Required.**
`ssl_approval_code`	The Voucher Clear Approval Code obtained previously using the phone. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_exp_date`	Credit Card Expiry date as it appears on credit card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment form (Available only for `process.do`). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.
`ssl_voucher_number`	The 15-digit Voucher Clear Number from Merchant's EBT Food Stamp sales slip. **Required.**

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.
ssl_voucher_number	The Voucher Clear Number.

Cash Benefit Purchase (cbpurchase)

All EBT purchases that are not food stamp related should be processed as Cash EBT purchases. Cash EBT transactions are very similar to debit transactions because customers can receive cash back from transactions. This message format is for the EBT Cash Benefit Purchase transaction for either a magnetic stripe read (Track II) or manually entered card.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Cash Benefit Purchase (cbpurchase)**
`ssl_amount`	Transaction total Amount including surcharge or cash back. Must be number with 2 decimal places. **Required.**
`ssl_card_number`	Credit Card Number as it appears on the credit card. **Required** for hand-keyed transactions.
`ssl_cashback_amount`	Cash back. The amount of cash back that the customer will receive. Must be number with 2 decimal places, if sent. **Optional**
`ssl_dukpt`	This is the value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored. **Required**.
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw Track I or II data only as read from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_base_amount	Base amount, the amount sent originally on the request.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_cashback_amount	Cash back. The amount of cash back that the customer will receive passed from the original request.
ssl_email	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_surcharge_amount	Surcharge amount as configured by merchant.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Cash Benefit Inquiry (cbbainquiry)

This transaction allows the merchant inquiry into current available balance in specified EBT accounts. A Track II card swipe is required (no manual entry).

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Debit Sale(dbbainquiry)**
`ssl_dukpt`	This is the value returned by the PIN pad device, which was used to encrypt the cardholder's Personal Identification Number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method. This value cannot be stored**. Required.**
`ssl_key_pointer`	Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant. **Required.**
`ssl_pin_block`	The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. **Required.**
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw Track I or II data only as read from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_reference_number	The Transaction Reference Number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

EGC (Gift Card Transactions)

This message format is for either a whole track, track 1 or track 2 magnetic stripe read or manually key entered gift card transactions available for all supported market segments.

Activation (egcactivation)

Gift cards must be activated prior to use.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Activation (egcactivation)**
`ssl_amount`	Transaction Sale Amount, number with 2 decimal places. For example: 1.00. **Required.**
`ssl_card_number`	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
`ssl_egc_tender_type`	This field is used to pass the tender type used to pay for the gift card. Valid Values are as follows: 0=cash 1=credit 2=debit 3=check
`ssl_exp_date`	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on Merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_egc_tender_type	This field is used to pass the tender type used to pay for the gift card. Valid Values are as follows: Cash Credit Debit Check
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Sale/Redemption (egcsale)

The Gift Card Redemption transaction is used to make a purchase using the balance on the gift card account.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Sale/Redemption (egcsale)**
`ssl_amount`	Transaction Sale Amount, number with 2 decimal places. For example: 1.00. **Required.**
`ssl_card_number`	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
`ssl_exp_date`	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise
`ssl_track_data`	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on Merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Card Refund (egccardrefund)

This transaction is used to reset the balance of a gift card account to zero, and the card is no longer usable.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Card Refund (egccardrefund)**
`ssl_card_number`	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
`ssl_exp_date`	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_track_data`	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required** on Swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Replenishment/Reload (egcreload)

This transaction is used to increase the current balance of the gift card account.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
ssl_transaction_type	**Replenishment/Reload (egcreload)**
ssl_amount	Transaction Sale Amount, number with 2 decimal places. For example: 1.00. **Required.**
ssl_card_number	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
ssl_egc_tender_type	This field is used to pass the tender type used to pay for the gift card. Valid Values are as follows: 0=cash 1=credit 2=debit 3=check
ssl_exp_date	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
ssl_merchant_id	VirtualMerchant ID as provided by Elavon.
ssl_pin	VirtualMerchant PIN as configured within VirtualMerchant.
ssl_show_form	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do) Set to false otherwise.
ssl_track_data	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
ssl_user_id	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_egc_tender_type	This field is used to pass the tender type used to pay for the gift card. Valid values are as follows: Cash Credit Debit Check
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Card Balance Inquiry (egcbalinquiry)

This option is used to check the current balance of a gift card account.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Card Balance Inquiry (egcbalinquiry)**
`ssl_card_number`	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
`ssl_exp_date`	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`) Set to false otherwise.
`ssl_track_data`	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstances. **Required** on swipe.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred. Typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Credit (egccredit)

This transaction is used to refund money back to a gift card account.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
ssl_transaction_type	**Credit (egccredit)**
ssl_amount	Transaction Sale Amount, number with 2 decimal places. For example: 1.00. **Required.**
ssl_card_number	Gift Card Number as it appears on the gift card. **Required** for hand-keyed transactions.
ssl_exp_date	Gift Card Expiry date as it appears on gift card. **Required** for hand-keyed transactions.
ssl_merchant_id	VirtualMerchant ID as provided by Elavon.
ssl_pin	VirtualMerchant PIN as configured within VirtualMerchant.
ssl_show_form	Set value to true to show the VirtualMerchant Payment Form (Available only for process.do). Set to false otherwise.
ssl_track_data	The raw Track I and/or II data from the magnetic strip on the card. The track data captured from the swipe device cannot be manipulated and must be passed at the time of the transaction. This includes the beginning and ending sentinels that are included in the track data. Raw track data cannot be stored under any circumstance. **Required** on swipe.
ssl_user_id	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_balance	The remaining balance on the gift card.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_card_number	The masked card number. Returned based on merchant setup.
ssl_email	Returned based on merchant setup.
ssl_exp_date	Returned based on merchant setup.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Electronic Check Transactions

A check reader device is required to electronically read and capture the Magnetic Ink Character Recognition **(**MICR) data from a check. The unformatted MICR data will be the exact MICR line from the check, including spaces, except that the MICR symbols will be replaced as follows ("raw TOAD" format):

- The Transit symbol (⑆) must be replaced by the letter (T) in either upper or lower case.
- The On-US symbol (⑈) must be replaced by the letter (O) in either upper or lower case.
- The Amount symbol (⑇) must be replaced by the letter (A) in either upper or lower case.
- The Dash symbol (⑉) must be replaced by the letter (D) in either upper or lower case.

Electronic Check Purchase (ecspurchase)

The `ecspurchase` is a transaction in which money is debited from a checking account using a check electronically. Data is captured from a paper check using a check MICR reader device. The check image must belong to the MICR check data sent and all images must be BASE64 encoded. Failure to properly format the image in a BASE64 encode message will result in an error when attempting to post the message. The type of the check Sale transaction is determined by the customer setting: Purchase with Conversion, Purchase with Verification or Purchase with Guarantee.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Check Sale (ECSSALE)**
`ssl_amount`	Transaction Sale Amount. Must be number with 2 decimal places. **Required.**
`ssl_check_image`	Check Image Data base 64 TIFF image. **Required.**
`ssl_drivers_license_number`	The Driver's License number as entered by the user. Alphanumeric. **Required** on Verification.
`ssl_drivers_license_phone_number`	Customers 10 digit Phone number including the area code. **Required** on Verification.
`ssl_drivers_license_state`	State Code.
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_micr_data`	MICR Number as read through the check reader. **Required.**
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment Form (Available only for `process.do`). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_aba_number	Routing/Transit Number from the parsed MICR data.
ssl_amount	The transaction amount. Returned based on merchant setup.
ssl_approval_code	The transaction approval code.
ssl_bank_account_number	Bank account number from the parsed MICR data.
ssl_check_number	The check Number from the parsed MICR data.
ssl_drivers_license_number	The Driver's License number as entered by the user.
ssl_drivers_license_phone_number	Customers 10 digit Phone number including the area code.
ssl_drivers_license_state	State Code.
ssl_email	Returned based on merchant setup.
ssl_reference_number	The reference number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Electronic Check Void (ecsvoid)

The `ecsvoid` is a transaction that reverses a Check Sale. No funds will be deposited into the merchant's bank account. The `ecsvoid` command is typically used to correct cashier mistakes. There is a 10-minute limited window in which a Check Card void can be completed, commencing with the original transaction approval. To perform an `ecsvoid` you must submit the transaction ID submitted in the original transaction. The `ssl_show_form` property does not apply on Void transactions.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**Check Void (ecsvoid)**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_txn_id`	Unique identifier returned on the original transaction. **Required.**
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
`errorCode`	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
`errorMessage`	Detailed explanation of the error returned ONLY if an error occurred, this field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
`errorName`	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
`ssl_approval_code`	The transaction approval code.
`ssl_result`	Outcome of a transaction. A response that contains `ssl_result` of 0 represents an Approved transaction. A response containing any other value for `ssl_result` represents a Declined transaction preventing it from being authorized.
`ssl_result_message`	The transaction result message. Example: APPROVAL.
`ssl_txn_id`	The transaction identification number. This is a unique number used to identify the transaction.
`ssl_txn_time`	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

PINLess Debit Transactions

This message format is for PINLess Debit Card transactions. Those types of debit transactions do not require a PIN pad or a PIN entry.

PINLess Debit Purchase (pldpurchase)

Transaction used to obtain a real-time authorization for a Debit Card sale transaction without PIN number input.

INPUT FIELD NAME XML/ HTML	DESCRIPTION
`ssl_transaction_type`	**PINLess Debit Purchase (pldpurchase)**
`ssl_account_type`	Account Type (0=checking, 1=saving). **Required.**
`ssl_amount`	Transaction base amount must be sent on request. This amount does not include the surcharge amount. Decimal, for example: 1.00. **Required.**
`ssl_card_number`	Debit Card Number as it appears on the debit card. **Required.**
`ssl_customer_number`	This value is used to submit the Customer Account Number or Payee account number for PINLess Debit transactions. **Required.**
`ssl_exp_date`	Debit Card Expiry date as it appears on debit card. **Required.**
`ssl_merchant_id`	VirtualMerchant ID as provided by Elavon.
`ssl_pin`	VirtualMerchant PIN as configured within VirtualMerchant.
`ssl_show_form`	Set value to true to show the VirtualMerchant Payment form (Available only for `process.do`). Set to false otherwise.
`ssl_user_id`	VirtualMerchant User ID as configured on VirtualMerchant.

OUTPUT FIELD NAME	DESCRIPTION
errorCode	Error code returned ONLY if an error occurred, typically when the transaction failed validation or the request is incorrect. This will prevent the transaction from going to authorization. This is a numeric field. Refer to the Error Codes section for more information.
errorMessage	Detailed explanation of the error returned ONLY if an error occurred. This field may be changed based on merchant configuration in the user interface. Refer to the Error Codes section for more information.
errorName	Error name or reason for the error returned ONLY if an error occurred. Refer to the Error Codes section for more information.
ssl_account_type	Account Type (checking=0, saving=1). **Required.**
ssl_amount	The total transaction amount including surcharge amount if any.
ssl_approval_code	The transaction approval code.
ssl_base_amount	The base transaction amount.
ssl_card_number	The masked debit card number passed on the original request.
ssl_customer_number	The customer number.
ssl_result	Outcome of a transaction. A response that contains ssl_result of 0 represents an Approved transaction. A response containing any other value for ssl_result represents a Declined transaction preventing it from being authorized.
ssl_result_message	The transaction result message. Example: APPROVAL.
ssl_surcharge_amount	The surcharge amount as configured by merchant.
ssl_txn_id	The transaction identification number. This is a unique number used to identify the transaction.
ssl_txn_time	Date and time when the transaction was processed. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM.

Chapter 8 Supported Transaction Input Fields

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_3dsecure_value	28		N	Sent on 3D Secure authenticated transactions only. Cardholder Authentication Verification Value. (NOTE: Called UCAF for MasterCard - Universal Cardholder Authentication Field). Base 64 Encoded (28 characters).
ssl_account_type	1		Y	Account Type (0=checking, 1=saving).
ssl_address2	30		N	Customer's address line 2.
ssl_amount	13		Y	Transaction total amount.
ssl_approval_code	6		Y	Return code generated by credit card processor. Must be passed on force.
ssl_avs_address	30		N	Customer's address used to process AVS.
ssl_avs_response	1		N	Return code generated by service. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_avs_zip	9		N	Customer's ZIP code used to process AVS.
ssl_background_color	20	Set In Terminal	N	Any HTML color value.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_billing_cycle	12		Y	Billing cycle. Valid returned values, all caps and no hyphens: - DAILY - WEEKLY - BIWEEKLY - SEMIMONTHLY - MONTHLY - BIMONTHLY - QUARTERLY - SEMESTER - SEMIANNUALLY - ANNUALLY - SUSPENDED
ssl_bill_on_half	1		Y	Half of the month or semi-monthly indicator. Valid values are **1** and **2**: - 1 = the 1st and the 15th of the month - 2 = the 15th and the last day of the month
ssl_card_number	19		Y	Card Number. Maximum length 18 for electronic gift cards.
ssl_card_present	1			Card present indicator. Valid values: Y or N. Hand-keyed.
ssl_cashback_amount	10		N	The amount of cash back that the customer will receive.
ssl_check_image			Y	Check image data base 64 TIFF image.
ssl_city	30		N	Customer's city.
ssl_company	50		N	Customer's company name.
ssl_country	3		N	Customer's country.
ssl_customer_code	17		N	Customer code.
ssl_customer_number	25		Y	This value is used to submit the customer account number or payee account number for PINLess debit transactions.
ssl_cvv2_response	1		N	Return code generated by service. Refer to the Authorization Response Codes section for an extensive list of possible returned messages.
ssl_cvv2cvc2	4		N	CVV2 CVC2 value from the card
ssl_cvv2cvc2_indicator	1		N	CVV2 Indicator. 0=Bypassed; 1=present; 2=Illegible; 9=Not Present.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_description	255		N	Transaction description.
ssl_do_customer_email	1	Set In Terminal	N	T=true. F=false.
ssl_do_merchant_email	1	Set In Terminal	N	T=true. F=false.
ssl_drivers_license_number			N	The driver's license number as entered by the user. Alphanumeric.
ssl_drivers_license_phone_number			N	Customer's 10-digit phone number including the area code.
ssl_drivers_license_state			N	State code.
ssl_dukpt			Y	This is the value returned by the PIN Pad device, which was used to encrypt the cardholder's personal identification number (PIN) using the Derived Unique Key Per Transaction (DUKPT) method.
ssl_eci_ind	1		Y	Sent on 3D Secure authenticated transactions only.
ssl_egc_tender_type	1		Y	This field is used to pass the tender type used to pay for the gift card. Valid Values are as follows: 0=cash 1=credit 2=debit 3=check
ssl_email	100		N	Customer's email address.
ssl_email_apprvl_footer_html	255	Set In Terminal	N	Customer order confirmation email footer for approvals.
ssl_email_apprvl_header_html	255	Set In Terminal	N	Customer order confirmation email header for approvals.
ssl_email_decl_footer_html	255	Set In Terminal	N	Customer order confirmation email footer for declines.
ssl_email_decl_header_html	255	Set In Terminal	N	Customer order confirmation email header for declines.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_email_footer	255		N	Customer order confirmation email footer. If present, overrides values. for ssl_email_apprvl_footer_html and ssl_email_decl_footer_html.
ssl_email_header	255		N	Customer order confirmation email header. If present, overrides values. for ssl_email_apprvl_header_html and ssl_email_decl_header_html.
ssl_end_of_month	1		Y	End of month indicator. Valid values Y or N. Must be passed on an add or an update of a recurring/installment transaction to indicate if the transaction is to be processed on the last day of the month.
ssl_error_url	255	Set In Terminal	N	If present, the error will be redirected to the URL specified including the errorCode, errorName, and errorMessage fields. Refer to the Error Codes section for an extensive list of possible codes.
ssl_exp_date	4		Y	Card expiry date.
ssl_first_name	20		N	Customer's first name.
ssl_footer_html	255	Set In Terminal	N	Payment form footer. Ignored when ssl_show_form=false.
ssl_header_color	20	Set In Terminal	N	Any HTML color value.
ssl_header_html	255	Set In Terminal	N	Payment form header, Ignored when ssl_show_form=false.
ssl_image_type	3		Y	Image format, required for signature transactions. Possible values, must be capital: - GIF - TIF - JPG - PNG

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_installment_id	50		Y	The ID number of the installment record that has been. Required on update. Alphanumeric. This value was returned when the original installment was added.
ssl_invoice_number	25		N	Invoice number.
ssl_key_pointer	1	T	N	The pointer that indicates to Elavon which encryption key was used for US debit transactions and which key to use for the next transaction. Value always "T."
ssl_last_name	30		N	Customer's last name.
ssl_link_color	20	Set In Terminal	N	Any HTML Color Value.
ssl_merchant_email	100		N	Merchant's email address.
ssl_merchant_id	15		Y	VirtualMerchant ID as provided by Elavon.
ssl_micr_data			Y	MICR number as read through the check reader.
ssl_next_payment_date	10		Y	Next payment date in MM/DD/YYYY format.
ssl_original_date	6		N	Date of original transaction in MMDDYY format.
ssl_original_time	6		N	Time of original transaction in HHMMSS format.
ssl_partial_auth_indicator	1	0	N	Partial Auth indicator required to indicate the support of partial approval. Valid values: 1
ssl_phone	20		N	Customer's phone number.
ssl_pin	6		Y	VirtualMerchant PIN as configured within VirtualMerchant.
ssl_pin_block			Y	The encrypted personal identification number entered by debit / EBT cardholder as identification for transaction.
ssl_receipt_apprvl_footer_html	255	Set In Terminal	N	Receipt form footer for approved transaction. Ignored when ssl_result_format=ASCII.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_receipt_apprvl_get_url	255	Set In Terminal	N	Target of the link generated at the bottom of the receipt for an approval using the "GET" method, or the target of the redirect for an approval using the "REDG" method. Ignored when ssl_result_format=ASCII and ssl_receipt_link_method = LINK or POST.
ssl_receipt_apprvl_header_html	255	Set In Terminal	N	Receipt form header for approved transaction. Ignored when ssl_result_format=ASCII.
ssl_receipt_apprvl_method	4	Set In Terminal	N	REDG = No receipt displayed. Data redirected to ssl_receipt_link_url. LINK = Receipt displayed. Link provided to return to ssl_receipt_link_url. GET = Receipt displayed. Button provided to send GET data to ssl_receipt_link_url. POST = Receipt displayed. Button provided to send POST data to ssl_receipt_link_url. LINK, GET, POST ignored when ssl_result_format=ASCII. If present, overwrites ssl_receipt_decl_method and ssl_receipt_link_method.
ssl_receipt_apprvl_post_url	255	Set In Terminal	N	Target of the link generated at the bottom of the receipt for an approval using the "POST" method. Ignored when ssl_result_format=ASCII.
ssl_receipt_apprvl_text	40	Set In Terminal	N	Text that appears on the receipts of approved transactions.
ssl_receipt_decl_footer_html	255	Set In Terminal	N	Receipt form footer for declined transaction. Ignored when ssl_result_format=ASCII.
ssl_receipt_decl_get_url	255	Set In Terminal	N	Target of the link generated at the bottom of the receipt for a declined transaction using the "GET" method, or the target of the redirect for a declined transaction using the "REDG" method. Ignored when ssl_result_format=ASCII and ssl_receipt_link_method = LINK or POST.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_receipt_decl_header_html	255	Set In Terminal	N	Receipt form header for declined transaction. Ignored when ssl_result_format=ASCII.
ssl_receipt_decl_method	4	Set In Terminal	N	REDG = No receipt displayed. Data redirected to ssl_receipt_link_url. LINK = Receipt displayed. Link provided to return to ssl_receipt_link_url. GET = Receipt displayed. Button provided to send GET data to ssl_receipt_link_url. POST = Receipt displayed. Button provided to send POST data to ssl_receipt_link_url. LINK, GET, POST ignored when ssl_result_format=ASCII. If present, overwrites ssl_receipt_apprvl_method and ssl_receipt_link_method.
ssl_receipt_decl_post_url	255	Set In Terminal	N	Target of the link generated at the bottom of the receipt for a declined transaction using the “POST” method. Ignored when ssl_result_format=ASCII.
ssl_receipt_decl_text	40	Set In Terminal	N	Text that appears on the receipts of declined transactions.
ssl_receipt_footer_html	255	Set In Terminal	N	Receipt form footer. Ignored when ssl_result_format=ASCII.
ssl_receipt_header_html	255	Set In Terminal	N	Receipt form header. Ignored when ssl_result_format=ASCII.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_receipt_link_method	4	Set In Terminal	N	REDG = No receipt displayed. Data redirected to ssl_receipt_link_url. LINK = Receipt displayed. Link provided to return to ssl_receipt_link_url. GET = Receipt displayed. Button provided to send GET data to ssl_receipt_link_url. POST = Receipt displayed. Button provided to send POST data to ssl_receipt_link_url. LINK, GET, POST ignored when ssl_result_format=ASCII. If present, overwrites ssl_receipt_apprvl_method and ssl_receipt_decl_method.
ssl_receipt_link_text	40	Set In Terminal	N	Text in the link / on the submit button generated at the bottom of the receipt page. Ignored when ssl_result_format=ASCII. If present, overwrites ssl_receipt_apprvl_text and ssl_receipt_decl_text.
ssl_receipt_link_url	255	Set In Terminal	N	Target of the Redirect or the link generated at the bottom of the VirtualMerchant drawn receipt. Ignored when ssl_result_format=ASCII and ssl_receipt_link_method = GET, POST, or REDG. If present, overwrites ssl_receipt_apprvl_post_url, ssl_receipt_decl_post_url, ssl_receipt_apprvl_get_url, and ssl_receipt_decl_get_url.
ssl_recurring_batch_count	4		N	Current number of transactions sitting in the recurring batch after the installment transaction has been added.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_recurring_id	50		Y	The ID number of the recurring record to be updated. Required on update. Alphanumeric. This value was returned when the original credit card record was added, to be used for update or delete or Auth. It is a unique tracking number that the application assigns internally to each recurring record in the database. This number is returned in the authorization response message originally when a user adds a recurring or installment credit card.
ssl_reference_number	40		N	The transaction reference number is returned in the authorization response.
ssl_result	1		N	Result code for the transaction. A result of 0 indicates an approval. Any other result means that the transaction was not approved.
ssl_result_format	5		N	When set to ASCII, VirtualMerchant will generate a plain text key-value document.
ssl_result_message			N	Result message for the transaction. A result of “APPROVAL” indicates that a transaction was approved. Any other result means that the transaction was not approved.
ssl_salestax	10		N	Sales tax.
ssl_ship_to_address1	30		N	Ship To Address Line 1.
ssl_ship_to_address2	30		N	Ship To Address Line 2.
ssl_ship_to_city	30		N	Ship To City.
ssl_ship_to_company	50		N	Ship To Company Name.
ssl_ship_to_country	3		N	Ship To Country.
ssl_ship_to_first_name	20		N	Ship To First Name.
ssl_ship_to_last_name	30		N	Ship To Last Name.
ssl_ship_to_phone	20		N	Ship To Phone Number.
ssl_ship_to_state	2		N	Ship To State.
ssl_ship_to_zip	10		N	Ship To ZIP.

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_show_form	5	TRUE	N	When set to false, VirtualMerchant will not present the payment form but process the transaction directly.
ssl_signature_image			Y	BASE 64 Encoded version of an IMAGE required for signature transactions and has size limit.
ssl_skip_payment	1	N	N	Skip payment field.
ssl_start_payment_date	10	N	N	Start payment date with format MM/DD/YYYY. Date when the first payment started. If recently added, start date is same as next payment.
ssl_state	2		N	Customer's state.
ssl_surcharge_amount	5	Set In Terminal	N	Surcharge amount to apply to this transaction; configurable.
ssl_table_color	20	Set In Terminal	N	Any HTML color value.
ssl_test_mode	5	FALSE	N	Optional when set to true. Transactions will not be forwarded to the credit card processor, but instead will always return an “APPROVED” result.
ssl_text_color				Any HTML color value.
ssl_total_installments	2		Y	Number of payments. Numeric. Max 99
ssl_track_data	76		Y	Track 1 or 2 data as read from a magnetic stripe reader (MSR).

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
ssl_transaction_type	20		Y	**Credit Transactions** - Sale (ccsale) - Auth Only (ccauthonly) - Credit (cccredit) - Force (ccforce) - AVS Only (ccavsonly) - Balance Inquiry (ccbalinquiry) - Void (ccvoid) - Delete (ccdelete) - Signature (ccsignature) **Recurring Transactions** - Add Recurring (ccaddrecurring) - Update Recurring (ccupdaterecurring) - Delete Recurring (ccdeleterecurring) - Submit Recurring (ccrecurringsale) **Installment Transactions** - Add Installment (ccaddinstall) - Update Installment (ccupdateinstall) - Delete Installment (ccdeleteinstall) - Submit Installment (ccinstallsale) **Debit Transactions** - Debit Purchase (dbpurchase) - Debit Return (dbreturn) - Debit Inquiry (dbbainquiry) **EBT Transactions** - Food Stamp Purchase (fspurchase) - Food Stamp Return (fsreturn) - Food Stamp Inquiry (fsbainquiry) - Food Stamp Force Purchase (fsforcepurchase) - Food Stamp Force Return (fsforcereturn) - Cash Benefit Purchase (cbpurchase) - Cash Benefit Inquiry (cbbainquiry)

FIELD_NAME	LENGTH	DEFAULT	REQ	DESCRIPTION
				Gift Card Transactions - Activation (`egcactivation`) - Sale / Redemption (`egcsale`) - Card Refund (`egccardrefund`) - Replenishment / Reload (`egcreload`) - Card Balance Inquiry (`egcbalinquiry`) - Credit (`egccredit`) **Check Transactions (ECS)** - Electronic Check Purchase (`ecspurchase`) - Void (`ecsvoid`) **PinLess Debit Transactions** - PINLess Debit Purchase (`pldpurchase`)
`ssl_txn_id`	50		Y	The transaction identification number. This is a unique number used to identify the transaction. Required for void and delete transactions.
`ssl_user_id`	15		Y	VirtualMerchant User ID as configured on VirtualMerchant.
`ssl_voucher_number`			Y	The voucher number from an EBT sales slip. Used for Voucher Clear Food Stamp transactions.
`ssl_xid`			Y	Sent on 3D Secure authenticated transactions Only. Unique transaction identifier assigned by eMPI.

Chapter 9 Authorization Response Codes

This is a list of the values that may be returned during an authorization request in the `ssl_result_message` field.

Credit Card Response Codes

CODE	MESSAGE	DEFINITION
AA	APPROVAL	Approved
AP	PARTIAL APPROVAL	Approved for a Partial Amount
N7	DECLINE CVV2	Do Not Honor
N7	DECLINE CVV2	Declined due to CVV2 mismatch \ failure
NC	PICK UP CARD	Pick up card
ND	AMOUNT ERROR	Tran Amount Error
ND	AMT OVER SVC LMT	Amount is more than established service limit
ND	APPL TYPE ERROR	Call for Assistance
ND	CANNOT CONVERT	Check is ok, but cannot be converted. Do Not Honor
ND	DECLINED	Do Not Honor
ND	DECLINED T4	Do Not Honor. Failed negative check, unpaid items
ND	DECLINED-HELP 9999	System Error
ND	DUP CHECK NBR	Duplicate Check Number
ND	EXPIRED CARD	Expired Card
ND	INCORRECT PIN	Invalid PIN
ND	INVALID CARD	Invalid Card
ND	INVALID CAVV	Invalid Cardholder Authentication Verification Value
ND	INVALID TERM ID	Invalid Terminal ID
ND	INVLD R/T NBR	Invalid Routing/Transit Number
ND	INVLD TERM ID 1	Invalid Merchant Number

CODE	MESSAGE	DEFINITION
ND	INVLD TERM ID 2	Invalid SE Number **NOTE:** Amex Only
ND	INVLD VOID DATA	Invalid Data Submitted for Void Transaction
ND	MAX MONTHLY VOL	The maximum monthly volume has been reached
ND	MICR ERROR	MICR Read Error
ND	MUST SETTLE MMDD	Must settle, open batch is over 7 days old. **NOTE:** Best Practice is to settle within 24 hours. Batch will be Auto Settled after 10 days
ND	NETWORK ERROR	General System Error
ND	PLEASE RETRY	Please Retry/ Reenter Transaction
ND	RECORD NOT FOUND	Record not on the network
ND	REQ. EXCEEDS BAL.	Req. exceeds balance
ND	SEQ ERR PLS CALL	Call for Assistance
ND	SERV NOT ALLOWED	Invalid request
ND	TOO MANY CHECKS	Too Many Checks (Over Limit)
NR	CALL AUTH. CENTER	Refer to Issuer
N/A	SUCCESS	For successfully added, updated, deleted recurring or installment transactions
N/A	ERROR	For recurring or installment transactions that failed to be added, deleted or updated

Electronic Gift Card (EGC) Response Codes

This table is a list of the values that may be returned during an EGC authorization request.

Authorization Response Codes		
Code	**Message**	**Definition**
AA	APPROVAL	Approved
ND	SERV NOT ALLOWED	Invalid request
ND	INVLD TERM ID 1	Invalid Merchant Number
ND	SEQ ERR PLS CALL	Call for Assistance
ND	APPL TYPE ERROR	Call for Assistance
01	DECLINED-HELP 9999	Host Busy
02	INVALID CARD	Invalid Card
03	INVALID TERM ID	Invalid Terminal ID
04	AMOUNT ERROR	Tran Amount Error
05	ALREADY ACTIVE	Card already active
06	REQ. EXCEEDS BAL.	Request exceeds balance
07	MAX REACHED	Cannot load the amount specified
08	NON RELOADABLE	The card cannot be reloaded
09	TRAN NOT ALLOWED	Transaction type not allowed
10	INVLD TRAN TYPE	Transaction type not on server
11	EXPIRED CARD	Expired card or bad expiration date
12	CARD NOT ACTIVE	The Gift Card is not activated
13	DUPLICATE TRAN	Duplicate transaction
14	SEQ ERR PLS CALL	Call for Assistance
15	SEQ ERR PLS CALL	Sequence does not match previous response
16	INVALID BATCH ID	Batch ID is not on the server
17	INVALID TENDER	Tender types is not on the server
99	DECLINED-HELP 9999	General System Error

AVS Response Codes

An **AVS Response Code** is returned in Authorization Response Message when AVS information is present in the transaction authorization request.

AVS Response Codes	
Code	**Definition**
A	Address matches - ZIP Code does not match
B	Street address match, Postal code in wrong format (International issuer)
C	Street address and postal code in wrong formats
D	Street address and postal code match (international issuer)
E	AVS Error
F	Address does compare and five-digit ZIP code does compare (UK only)
G	Service not supported by non-US issuer
I	Address information not verified by international issuer.
M	Street Address and Postal code match (international issuer)
N	No Match on Address (Street) or ZIP
O	No Response sent
P	Postal codes match, Street address not verified due to incompatible formats
R	Retry, System unavailable or Timed out
S	Service not supported by issuer
U	Address information is unavailable
W	9-digit ZIP matches, Address (Street) does not match
X	Exact AVS Match
Y	Address (Street) and 5-digit ZIP match
Z	5-digit ZIP matches, Address (Street) does not match

CVV2/CVC2 Response Codes

The **CVV2 Response Codes** are returned in an Authorization Response Message when the CVV2 data is present in the transaction authorization request.

CVV2 Response Codes	
Code	**Definition**
M	CVV2 Match
N	CVV2 No match
P	Not Processed
S	Issuer indicates that CVV2 data should be present on the card, but the merchant has indicated that the CVV2 data is not resent on the card
U	Issuer has not certified for CVV2 or Issuer has not provided Visa with the CVV2 encryption keys

Error Codes

A **VirtualMerchant Error Code, Error Name** and **Error Message** are returned when the transaction fails to be authorized. This could be the result of a data or system error, or if the transaction is **Declined**. Note that error messages can be customized in the Virtual Terminal admin setup by the merchant.

CODE	ERROR NAME	DEFAULT MESSAGE
3000	Gateway not responding	Error, no response.
3001	Gateway generated error	#.
3002	Adapter generated error	#.
4000	VID Not Supplied	The VirtualMerchant ID was not supplied in the authorization request.
4001	VID, UID and PIN Invalid	The VirtualMerchant ID, User ID and/or PIN supplied in the authorization request are invalid.
4002	HTTP Trans Not Allowed	HTTP POST transactions are not allowed for this account.
4003	HTTP Referrer Invalid	HTTP POST transactions are not allowed for this HTTP Referrer.
4005	E-mail Address Invalid	The E-mail Address supplied in the authorization request appears to be invalid.
4006	CVV2 Not Requested With Data	The CVV2 indicator was not identified in the authorization request.
4007	CVV2 Requested But No Data	CVV2 check cannot be performed as no data was supplied in the authorization request.
4009	Required Field Not Supplied	A required field was not supplied in the authorization request.
4010	Invalid Transaction Type	An invalid Transaction Type was supplied in the authorization request.
4011	Receipt URL Missing	The Receipt URL supplied in the authorization request appears to be blank or invalid.
4012	VID/UID Invalid	The VirtualMerchant ID and/or User ID supplied in the authorization request is invalid.
4013	PIN Not Supplied	The PIN was not supplied in the authorization request.
4014	Not Permitted	This terminal or user ID is not permitted to process this transaction type.
4015	PIN Invalid	The PIN supplied in the authorization request is invalid.
4016	Permission Denied	This account does not have permission to process # transactions.

CODE	ERROR NAME	DEFAULT MESSAGE
4017	Time-Out	The request has timed out. The allotted time to complete the request has ended. Please try again.
4018	Settlement is in progress	Settlement is in progress, Void failed.
5000	Credit Card Number Invalid	The Credit Card Number supplied in the authorization request appears to be invalid.
5001	Exp Date Invalid	The Credit Card Expiration Date supplied in the authorization request appears to be invalid.
5002	Amount Invalid	The amount supplied in the authorization request appears to be invalid.
5003	Approval Code / No Force	A FORCE Approval Code was supplied for this transaction; however the transaction type is not FORCE.
5004	Invalid Approval Code	The FORCE Approval Code supplied in the authorization request appears to be invalid or blank. The FORCE Approval Code must be 6 or less alphanumeric characters.
5005	Field Character Limit Exceeded	The value for the # field is too long. # characters (maximum) are allowed. Your entry contains # characters. If you entered the value for this field, use the browser BACK button to return to the order form and modify the field value accordingly. Otherwise, contact Customer Service at #.
5006	Refund Amount Exceeds Limit	The refund amount for this transaction ($#) may not exceed $#.
5007	Sales Tax Invalid	The Sales Tax supplied in the authorization request appears to be invalid.
5008	Invalid Account Type	This PIN Less Debit Transaction contains invalid account type. Account type can be checking or saving.
5009	Invalid Surcharge Amount	Invalid Surcharge amount for the PIN less debit transaction.
5010	Invalid EGC Transaction type	An invalid EGC Transaction type has been supplied with this request.
5011	Invalid EGC Tender Type	An invalid EGC Tender type has been supplied with this request.
5012	Invalid Track Data	The track data sent appears to be invalid.
5013	Invalid Track 2 data	Transaction requires Track2 data to be sent.
5014	Missing Pin Data	Transaction requires a Pin entry or encrypted Pin device.
5015	Invalid Voucher Number	The value for the Voucher Number (`ssl_voucher_number`) field should be 15 digits in length. This value must be numeric.

CODE	ERROR NAME	DEFAULT MESSAGE
5016	Invalid MICR Data	The MICR Data sent appears to be invalid.
5017	MICR data and image mismatch	The image uploaded doesn't match the MICR data sent for that check.
5019	Minimum Length Error	Minimum Field Character Limit not reached.
5020	Invalid Field	The Field does not apply to this transaction type
5021	Invalid CVV2 Value	The value for the CVV2 (`ssl_cvv2cvc2`) field should be 3 or 4 digits in length. This value must be numeric.
5022	Invalid CVV2 Indicator Value.	The value for the CVV2 indicator (`ssl_cvv2cvc2_indicator`) field should be 1 Numeric Character only. Valid values are: 0, 1, 2, 9.
5023	Invalid card present indicator	Card present indicator sent is invalid.
5024	CashBack amount Invalid	The Cash back amount supplied in the authorization request appears to be invalid.
5025	Invalid Key pointer	The value for the key pointer (`ssl_key_pointer`)) field should be 1 Character only. Valid values is: T for Triple-DES DUKPT.
5030	Invalid Billing cycle	The billing cycle specified is not a valid entry.
5031	Invalid Payment date	The next payment date specified is not a valid entry.
5032	Invalid installment number	The installment number specified is invalid.
5033	Invalid recurring ID	The recurring ID is not valid.
5034	Invalid installment ID	The installment ID is not valid.
5035	Recurring Limit exceeded	The recurring batch has exceeded the 20,000 transactions limit.
5036	Installment payments completed	Installment payments completed.
5037	Invalid end of month value	Invalid end of month value.
5038	Invalid half of month value	Invalid half of month value.
5039	Half of month and next payment date combination mismatch	The half of the month value provided [value] doesn't correspond with the next payment date of [value].
5040	Invalid Transaction ID	The transaction ID does not exist in the database for a valid Credit-card, debit card, or E-check transaction.
5041	Exceeded the 10 minutes window	Unable to void transaction, exceeds the 10mn window.

CODE	ERROR NAME	DEFAULT MESSAGE
5042	Swipe data is not allowed for this market segment	Swipe data is not allowed for this market segment. Please rekey the card data.
5070	Signature already in System	The transaction ID sent has a signature associated to it in the system.
5071	Signature format Invalid	All signature images must be BASE64 encoded.
5072	Signature type Invalid	Signature image type valid values (JPG, GIF, TIF, or PNG).
5073	Signature image exceeds size limitation	Signature image exceeds the 5K size quota.
5080	values for ssl_3dsecure_value and ssl_xid are required	Values for ssl_3dsecure_value and ssl_xid are required.
5081	value for ssl_xid is required	Value for ssl_xid is required.
6001	Manual Transaction Declined	The transaction request was unable to be completed:
6002	Declined: Invalid Card	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6003	Declined: Pick up Card	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6004	Declined: Amount Error	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6005	Declined: Appl. Type Error	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6006	Declined	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6007	Declined: Help	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6008	Declined: Req. Exceeds Bal.	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.

CODE	ERROR NAME	DEFAULT MESSAGE
6009	Declined: Expired Card	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6010	Declined: Incorrect PIN	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6011	Declined: Invalid Term ID	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6012	Declined: Invalid Term ID 1	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6013	Declined: Invalid Term ID 2	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6014	Declined: Invalid Void Data	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6015	Declined: Must Settle MMDD	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6016	Declined: Not On File	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6017	Declined: Record Not Found	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6018	Declined: Serv Not Allowed	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6019	Declined: Seq Err Pls Call	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6020	Declined: Call Auth Center	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.

CODE	ERROR NAME	DEFAULT MESSAGE
6021	Declined: Call Ref.	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6022	Declined: CVV2	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6023	Declined: Please RetryXXXX	This transaction request has not been approved. You may elect to use another form of payment to complete this transaction or contact customer service for additional options.
6024	Card Already Active	The Gift Card is already active.
6025	Request Exceeds Balance	The transaction amount exceeds the Gift Card balance amount.
6026	Cannot Load The Amount Specified	Cannot Load The Amount Specified.
6027	Card Not Activated	The Gift Card Is Not Activated.
6028	Card Cannot Be Reloaded	The Gift Card Cannot Be Reloaded.
6029	Declined: Invalid Reg Key	Invalid Reg Key.
6030	Declined: Invalid Packet	Invalid Packet.
6031	Declined: Invalid LRC	Invalid LRC.
6032	Declined: Invalid Response	Invalid Response.
6033	Declined: Invalid LRC in Response	Invalid LRC in Response.
6034	Declined: Invalid Record Number in Response	Invalid Record Number in Response.
6038	System is Temporarily Unavailable	It appears that the system is temporarily unavailable. Please try your transaction again in a few minutes or contact the merchant you are trying to order from for further assistance. We apologize for this inconvenience.
6042	Invalid Request Format	XML request is not well-formed or request is incomplete.

Chapter 10 Transaction Security

Transaction security should be a key component of all merchant policies and practices related to payment acceptance and transaction processing. As customers seek out merchants that are reputable and reliable, they expect assurance that their account information is being guarded and their personal data is safe. By following a few recommendations and adhering to the latest Payment Card Industry (PCI) - Payment Application Data Security Standard (PA-DSS) guidelines, the VirtualMerchant integrators and merchants can keep cardholder data safe. Transaction security is everyone's responsibility.

This chapter covers the following topics:

- Common fraudulent activities
- Best practice tips
- PA-DSS guidelines
- External security resources

Common Fraudulent Activities

Fraud schemes are becoming more sophisticated, and so it becomes increasingly important to be vigilant and get familiar with the most common fraud activities and learn how to fight them.

Authorization testing – is the practice of submitting bulk generated credit card numbers to attempt to find valid accounts using stolen credentials. Once valid card numbers are identified, the auth tester either sells the data, or uses the information for other financial gains.

Phishing – is when the sender of an electronic communication tries to trick recipients into volunteering personal or credential-related information. That information can then be used to commit identity theft, or enter password-protected sites using your account. Phishing emails claim to be from legitimate sources such as Elavon, the IRS or a friend, and typically use two components:

1. An authentic-looking email, and
2. A real-looking, but fraudulent, Web page that asks you to supply personal information (name, address, financial information, passwords, etc.)

Phishing attempts may try to trick users to give away their VirtualMerchant login credentials.

Malware – is malicious software that consists of code, scripts or active contents that is designed to gather information that leads to loss of privacy and gain unauthorized access to systems.

Malware includes computer viruses, worms, Trojan horses, spyware, dishonest adware, scareware, crimeware, most rootkits, and other malicious and unwanted software or program.

Common treats:

- **Key logger**: This intercepts the user's keystrokes when entering a password, credit card number, or other information that may be exploited. This is then transmitted to the malware creator automatically, enabling credit card fraud and other theft.
- **Screen scrapers**: Programmatic collection of visual data or reading text data from a computer display terminal's screen.
- **Cache miners**: Stealing data left in memory and cache.
- **Session hijacking**: Using cookies to gain unauthorized access to information or services in a computer system.
- **Botnets**: Sophisticated malware that compromise Web sessions after the data has been decrypted, stealing account credentials as they are entered and transparently redirecting users to hostile sites.

Best Practice Tips

Developers and merchant administrators may find the information presented here valuable when writing and configuring applications and websites that will interface with VirtualMerchant. These best practices focus on ways to increase security and reduce the chance of fraudulent activities. There are measures contained that we would like to emphasize in order to help you protect yourself against fraudulent activities.

HTTP Referrer – Setting up HTTP referrers in the Administration website tells VirtualMerchant to only accept transactions from a pre-approved list of websites. While it requires more work to implement, this action helps to prevent fraudulent users from submitting transactions from their websites, claiming to be you.

Server Side Code – Your users can read HTML source code from your Web pages when they are downloaded to their Web browser. Although our simple examples in the document show this as a method for passing data to VirtualMerchant, we do not recommend this for your production website. All sensitive merchant data, including transaction amounts and your VirtualMerchant credentials, should be placed in server side code, rather than in hidden value fields on an HTML form. This will reduce the ability of malicious users to exploit client browser vulnerability to edit and use this data for their own fraudulent purposes. If you are not knowledgeable enough to implement this on your own, there are quite a few shopping cart providers that inherently provide this service and are compatible with VirtualMerchant.

Auto Pend – We recommend that you use this feature for any account that is set to **Auto-Settle**. This gives you the chance to review each transaction before it becomes finalized. This will help you to avoid settling fraudulent transactions or transactions that you are unable to fulfill.

Merchant Admin – The Merchant Admin account has full rights and access to each terminal in your system. We recommend that you use this account sparingly. We suggest that you create one or more separate accounts to manage day-to-day activities, including but not limited to: processing transactions from your website, processing Virtual Terminal transactions, reviewing transactions, and settling transactions. We recommend that you do this even in the case of an account with only one terminal.

Password Security – Do not set your password to be the same value (or a similar value) as any other data associated with your VirtualMerchant account. This includes your VirtualMerchant PIN used for submitting transactions to `process.do`. This PIN is not designed as a security feature. It is only used to ensure that transactions sent into VirtualMerchant are assigned to the correct account, user and terminal. Unlike the passwords, the PIN is not stored as encrypted data in our database. Your password is a highly confidential piece of data and is treated as such. Our administrators do not have access to your password data. You should make all passwords to your accounts as difficult to guess as possible.

Settings in Admin Site – We recommend that whenever possible, set terminal options in the Administrative site instead of setting equivalent parameters in code on your Web page. This will make it easier to maintain, and will reduce the amount of data that is passed across the Internet with each of your transactions.

Business Rules – Are a customizable set of tools that allow you to build constraints to match merchant business needs and control how transactions should be handled. This includes the ability to approve, decline or hold transactions for manual review. These can serve as important tools to help fight, manage, and prevent suspicious and costly fraudulent transactions. Transactions can be set to automatically decline or held for review at a later time. Business rules include:

- Ship To Postal Code
- Bill To Postal Code
- Tran Amount
- Return Amount
- Duplicate Checking
- AVS Response
- CVV response
- Settlement

3D Secure™ Authentication (Verified by Visa™ and MasterCard SecureCode™) – The number one reason shoppers do not make purchases online is because they have concerns about security. The biggest frustration for e-commerce businesses has been the risk of chargebacks, if a shopper were to tell their issuers that they did not authorize an Internet purchase with their credit card. By using VirtualMerchant 3D Secure capabilities, merchants get explicit evidence of authorized purchases (authentication data). The authentication data, together with an authorization approval gives you a transaction that is guaranteed against the most common types of chargebacks— "*cardholder not authorized*" and "*cardholder not recognized*" chargebacks.

3D Secure is a security tool that enables cardholders to authenticate their identity to their card issuer through the use of Visa's Verified by Visa™ and MasterCard's SecureCode™ services. 3D Secure adds another layer of security to cardholders by preventing fraudulent purchases in an e-commerce environment and reducing the number of unauthorized transactions. VirtualMerchant users processing transactions in an integrated e-commerce environment are able to take advantage of this functionality.

Cardholders who have Visa or MasterCard from a participating issuer will be presented an additional window hosted by the card issuer. If a cardholder has already established a password or private code for their credit card, they will be prompted to simply enter that identifier to authenticate before the transaction is submitted for authorization. If a cardholder has a participating credit card but has not yet established their password or private code, they will be prompted to do so.

To process 3D Secure, the terminal must be set as e-commerce and the 3D Secure option must be enabled in the Virtual Terminal under the **Terminal |Advanced | System Setup| Processing Options** section.

Custom Fields – The custom defined fields feature allows a merchant to create user defined fields that fit every business need. However, merchants should not use those fields to pass any sensitive data including but not limited to PAN data such as full card number, expiration date, track data, or CID/CVV2 data from a credit card. Furthermore, customer account numbers, social security numbers, and other private data should not be passed unmasked or unprotected.

Other Measures

CAPTCHA Verification – Secure the payment form on your site behind a user authentication system if at all possible by implementing CAPTCHA verification. This will make it more difficult for someone to write a tool to automate authorization testing using your website.

Velocity Check – Monitor your account traffic by implementing velocity check capability. This will control the number of authorization requests that can be made to your server in a given time period from one IP address. This will help you to identify abuse of your system and limit the damage if any other preventative measures prove ineffective.

Anti-Phishing – There are several different techniques to combat phishing. One strategy is to train people how to recognize phishing attempts, and to deal with them. Be suspicious of requests for personal information that come by emails or text messages, particularly requests for passwords, banking information, or wire transfers of money, even if the request seems to come from a good friend. Elavon will never request your password or other sensitive information by an email or text message.

Anti-Malware – As malware attacks become more frequent, attention has begun to shift from viruses and spyware protection, to malware protection, and programs have been developed specifically to combat them. Stay protected and install anti-malware programs and run scans periodically. Those types of programs can provide real time protection against the installation of malware software on a computer, and can be used to detect and remove malware software that has already been installed onto a computer. Restrict access to systems and user rights, and use the payment system for business purposes only.

PA-DSS Guidelines

PCI DSS, a set of comprehensive requirements for enhancing payment account data security, was developed by the founding payment brands of the PCI Security Standards Council that includes American Express, Discover Financial Services, JCB International, MasterCard Worldwide and Visa Inc. International. The purpose was to help to facilitate the broad adoption of consistent data security measures on a global basis. It is intended to help organizations proactively protect customer account data.

The core of the PCI DSS is a group of principles and accompanying requirements, around which the specific elements of the DSS are organized:

- Build and Maintain a Secure Network
- Protect Cardholder Data
- Maintain a Vulnerability Management Program
- Implement Strong Access Control Measures
- Regularly Monitor and Test Networks
- Maintain an Information Security Policy

The relationship between PCI DSS and PA-DSS

The requirements for the PA-DSS are derived from the PCI DSS Requirements. Elavon is fully committed to merchant and cardholder security. VirtualMerchant is PCI certified and merchants using VirtualMerchant are required to support and facilitate customers' PCI DSS compliance.

The goal of PA-DSS is to help software vendors and others to develop secure payment applications that do not store prohibited data, such as full magnetic stripe, CVV2 or PIN data, and ensure their payment applications support compliance with the PCI DSS. Payment applications that are sold, distributed or licensed to third parties are subject to the PA-DSS requirements.

Merchants and integrators are responsible to ensure that their application runs and adheres to the latest PA-DSS guidelines. They are also responsible to make sure that they keep current on those guidelines. The following are provided by Elavon for your convenience. For a complete description we **strongly** recommend that you use the additional resources provided in the External Security Resources section below:

1. Do not retain full magnetic stripe, card validation code or value (CAV2, CID, CVC2, CVV2), or PIN block data.
2. Protect stored cardholder data.
3. Provide secure authentication features.
4. Log payment application activity.
5. Develop secure payment applications.
6. Protect wireless transmissions.
7. Test payment applications to address vulnerabilities.
8. Facilitate secure network implementation.
9. Never store cardholder data on a server connected to the Internet.

10. Facilitate secure remote software updates.
11. Facilitate secure remote access to payment application.
12. Encrypt sensitive traffic over public networks.
13. Encrypt all non-console administrative access.
14. Maintain instructional documentation and training programs for customers, resellers and integrators.

External Security Resources

Elavon encourages integrators to use the resources below to ensure that all software applications and networks are within the PCI-DSS guidelines.

COMPANY AND RESOURCE	DESCRIPTION
Company: PCI Security Standards Council Website: https://www.pcisecuritystandards.org/	The PCI Security Standards Council is an open global forum for the ongoing development, enhancement, storage, dissemination and implementation of security standards for account data protection.
Resource: https://www.pcisecuritystandards.org/security_standards/ped/pedapprovallist.html	List of approved PIN Transaction Security (PTS) Devices.
Resource: https://www.pcisecuritystandards.org/security_standards/pa_dss.shtml	Payment Application (PA) Security Standards
Resource: https://www.pcisecuritystandards.org/pdfs/pci_pa_dss_program_guide.pdf	PCI / PA Programming Guide
Company: Visa	Visa is a global payments technology company that enables consumers, businesses, financial institutions and governments to use digital currency instead of cash and checks.
Resource: http://usa.visa.com/merchants/risk_management/cisp.html	Cardholder Information Security Program – Visa provides extensive information regarding the protection of cardholder data
Company: MasterCard Worldwide	
Resource: http://www.iian.ibeam.com/events/mast001/24008/	The PCI 360 Program is a complimentary initiative offered by MasterCard to raise awareness and promote the adoption of PCI. The program provides a holistic and informative platform for participants to increase their understanding of PCI DSS through the following sessions led by payment industry and data security experts.

COMPANY AND RESOURCE	DESCRIPTION
Company: Trustwave Information Security & Compliance	Trustwave is the leading provider of on-demand data security and payment card industry compliance management solutions to businesses and organizations throughout the world.
Company Website:https://www.trustwave.com/	
Company: Microsoft	
Resource: http://www.microsoft.com/security/default.aspx	Application Security Information for Developers, IT Professionals, Consumers, and Businesses
Additional Resource: PCI Compliance Guide	
Resource: http://www.pcicomplianceguide.org/pcifaqs.php	Comprehensive list of FAQs related to the PCI Standards

Chapter 11 Common Code Samples

Overview

All commonly used programming languages have the ability to emulate both HTTP and HTTPS-capable clients. When emulating a client using the HTTPS protocol, the messages will always have a header and conform to every request for comments (RFC) requirement. We have found that some of the most commonly used modules in some of the more frequently used languages include curl (php), LWP:UserAgent (perl), urllib, and urllib2 (python). These are the modules that are used in the code references, as well.

If you are using a server side script written in PHP, ASP, ASP.NET, JSP, etc., normally the browser and the server that is interpreting your scripting will handle all of this for you. However, if you find yourself needing to post data from the back end of your system, where no browsers or clients will be interpreting or managing the connection, you will have to ensure that all of your communications are valid under the RFC. This leads us to the most important part of the protocol, the headers.

Improperly formatted headers will cause you to either, get no response (timeout) or an HTTP server error response (404,302, etc.) instead of any usable response from VirtualMerchant.

Below is an example of a valid request header from a POST sent to `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do`. Your application's headers will be different from this, but some fields are required by the RFC. Consult the RFC to locate these fields.

You will notice from the user-agent field that the sample was created using Firefox 3.0 on windows 2000, but anything can be sent here if you are writing an application to make the post. We do not filter user-agents.

```
Begin Header


POST /VirtualMerchantDemo/process.do HTTP/1.1

Host: demo.myvirtualmerchant.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US;
rv:1.8.1.6) Gecko/20080725 Firefox/3.0

Accept:text/xml,application/xml,application/xhtml+xml,text/html;q
=0.9,text/plain;q=0.8,image/png,*/*;q=

0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gZIP,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Content-Type: application/x-www-form-urlencoded

Content-Length: 492

ssl_amount=1.00&ssl_merchant_id=xxxxxx&ssl_user_id=xxxxxxx&ssl_pi
n=xxxxxx&ssl_show_form=false&ssl_card_number=5000000000000003&ssl
_exp_date=1208&ssl_error_url=http%3A%2F

%2Fwww.url.com%2Fcgi-
bin%2Ftesttran.cgi&ssl_result_format=HTML&ssl_transaction_type=cc
sale&ssl_receipt_decl_method=REDG&ssl_receipt_decl_get_url=http%3
A%2F%2Fwww.url.com%2Fcgi-bin

%2Ftesttran.cgi&ssl_receipt_apprvl_method=REDG&ssl_receipt_apprvl
_get_url=http%3A%2F

%2Fwww.url.com%2Fcgi-bin%2Ftesttran.cgi


End Header
```

The previous POST message sends all the required information for VirtualMerchant to attempt to process a card for this merchant, and to define how this transaction should flow. This is covered in depth in the next section. Following is the server response and the outcome of the transaction.

Begin server response:

```
HTTP/1.x 200 OK

Date: Sat, 16 Aug 2008 15:09:37 GMT

Server: IBM_HTTP_Server/6.0.2.15 Apache/2.0.47 (Win32)

Set-Cookie: JSESSIONID=0000Ab-5nZbCEHx_3NQWJMUiBYC:11pe0mr91;
Path=/

Set-Cookie: JSESSIONID=0000N17B4UhHgxRYZ0zFSP4e9aQ:11pe0mr91;
Path=/

Keep-Alive: timeout=10, max=100

Connection: Keep-Alive

Transfer-Encoding: chunked

Content-Type: text/html;charset=UTF-8

Content-Language: en-US

End Server response

Begin redirect

GET /cgi-
bin/testtran.cgi?errorCode=4009&errorName=Required%20Field%20Not

%20Supplied&errorMsg=The%20field%20Test%20Forward%20(ssl_test_for
ward)%20required%20but%20not%20supplied%20in%20the%20authorizatio
n%20request. HTTP/1.1 Host: www.url.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US;
rv:1.8.1.6) Gecko/20070725 MSIE 6.0


Accept:

text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,te
xt/plain;q=0.8,image/png,*/*;q=0.5

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gZIP,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Proxy-Connection: keep-alive


End Redirect
```

There are some very basic methods of posting data to VirtualMerchant from different programming languages.

Code Samples

Perl Sample

Perl Input

```
#!/usr/bin/perl
use LWP::UserAgent;
$ua = LWP::UserAgent->new;
$ua->agent("$0/0.1 " . $ua->agent);
$ua->agent("Mozilla/8.0"); # pretend we are very capable browser
$req = HTTP::Request->new(GET =>
"https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do?
ssl_merchant_id=xxxxxx&ssl_user_id=xxxxxx&ssl_pin=xxxxxx&ssl_show_fo
rm=false&ssl_result_format=ascii&ssl_card_number=5000000000000003&ss
l_exp_date=1208&ssl_amount=1.02&ssl_transaction_type=ccsale&ssl_cvv2
cvc2_indicator=1&ssl_cvv2cvc2=123");# add all of the fields here for
link
variables
$req->header('Accept' => 'text/html'); # send request
$res = $ua->request($req); # check the outcome
if ($res->is_success) {
print $res->decoded_content;
} else {
print "Error: " . $res->status_line . "\n";};
```

Perl Output

This script as it is will net the following output to your console window

```
ssl_card_number=50********0003
ssl_exp_date=1208
ssl_amount=1.02
ssl_customer_code=
ssl_salestax=
ssl_invoice_number=
ssl_result=0
ssl_result_message=APPROVED
ssl_txn_id=1252E7696-A94F-9A37-4235-48A287CFEC68
ssl_approval_code=N15032
ssl_cvv2_response=
ssl_avs_response=
ssl_account_balance=0.00
ssl_txn_time=08/17/2008 10:15:59 AM
```

NOTE: This Perl post will require the `Crypt::SSLeay` module to connect using SSL. If you do not have it the perl interpreter will tell you. You can get it from cpan. We are using the LWP module for client emulation and are just sending a GET request and retrieving results in ASCII. You could improve on this greatly by making an array or a hash that contains all the expected responses, so that your script can parse through the response for you utilizing regular expressions such as:

```
if ($response->decoded_content =~m/approved/) {print "transaction
approved\n";}

elsif ($response->decoded_content =~m/declined/) {print "transaction
declined\n";}

else {print "There has been an error with your transaction\n";}
```

PHP Sample

The first PHP page will send the post to VirtualMerchant when a client requests it or when a different script calls it. Here, we use Curl to emulate a client. We are again just sending a GET string to VirtualMerchant. Also included are some basic PHP pages that will parse responses called `response.php` and `error.php`. You will notice in the POST, we direct VirtualMerchant to send our responses to our return scripts. You could combine all this into one file if you wish, by using functions and parameters. Once again you can use a regular expression to make decisions on what to show. You would also need to implement the entire table of known responses for your conditional statements to be effective. These are very basic examples that do not handle cookies or sessions. There are many elaborate ways this can be achieved.

virtualmerchant.php

```
<?php

//extract data from the post

extract($_POST);


//set POST variables

$url =
'https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do';

//Modify the values from xxx to your own account ID, user ID, and
PIN

//Additional fields can be added as necessary to support custom
fields or required fields configured in the Virtual Merchant
terminal

$fields = array(

   'ssl_merchant_id'=>'xxx',

   'ssl_user_id'=>'xxx',

   'ssl_pin'=>'xxx',

   'ssl_show_form'=>'false',

   ‘ssl_result_format'=>'html',

   'ssl_test_mode'=>'false',

   'ssl_receipt_apprvl_method'=>'redg',
```

```
    //modify the value below from xxx to the location of your error
    script
    'ssl_error_url'=>'xxx',
    //modify the value below from xxx to the location of your receipt
    script
    'ssl_receipt_apprvl_get_url'=>'xxx',
    'ssl_transaction_type'=>urlencode($ssl_transaction_type),
    'ssl_amount'=>urlencode($ssl_amount),
    'ssl_card_number'=>urlencode($ssl_card_number),
    'ssl_exp_date'=>urlencode($ssl_exp_date),
    'ssl_cvv2cvc2_indicator'=>urlencode($ssl_cvv2cvc2_indicator),
    'ssl_cvv2cvc2'=>urlencode($ssl_cvv2cvc2),
    'ssl_customer_code'=>urlencode($ssl_customer_code),
    'ssl_invoice_number'=>urlencode($ssl_invoice_number),
    );

//initialize the post string variable
$fields_string = '';

//build the post string
foreach($fields as $key=>$value) { $fields_string
.=$key.'='.$value.'&'; }
rtrim($fields_string, "&");

//open curl session
$ch = curl_init();

//begin seting curl options
//set URL
curl_setopt($ch, CURLOPT_URL, $url);
//set method
curl_setopt($ch, CURLOPT_POST, 1);
//set post data string
curl_setopt($ch, CURLOPT_POSTFIELDS, $fields_string);
//these two options are frequently necessary to avoid SSL errors
with PHP
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
```

```
//perform the curl post and store the result
$result = curl_exec($ch);

//close the curl session
curl_close($ch);

//a nice message to prevent people from seeing a blank screen
echo "Processing, please wait..."
```

error.php

```
<?php
$ssl_error=$_GET['errorCode'];
if ($ssl_error < 4000)
{echo "System error";}
else if ($ssl_error > 4000)
{echo "Authentication error , uid, vid, or pin";}
else
{echo "syntax error";}
?>
```

Response.php

```
<?php
$ssl_result=$_GET['ssl_result'];
if ($ssl_result == 0 )
{ echo "Transaction approved";}
else if ($ssl_result==1)
{ echo "Transaction Declined;}
?>
```

Python Sample

PYTHON INPUT

```
import sys, urllib2, urllib

url =
'https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do?

ssl_merchant_id=xxxxxx&ssl_user_id=xxxxxx&ssl_pin=xxxxxx&ssl_show_fo
rm=false&ssl_result_format=ascii&ssl_card_number=5000000000000003&ss
l_exp_date=1208&ssl_amount=1.02&ssl_transaction_type=ccsale&ssl_cvv2
cvc2_indicator=1&ssl_cvv2cvc2=123'

req = urllib2.Request(url)

fd = urllib2.urlopen(req)

while 1:

data = fd.read(1024)

if not len(data):

break

sys.stdout.write(data)
```

PYTHON OUTPUT

The above python script will net the following response from VirtualMerchant:

```
ssl_card_number=50********0003

ssl_exp_date=1208

ssl_amount=1.02

ssl_customer_code=

ssl_salestax=

ssl_invoice_number=

ssl_result=0

ssl_result_message=APPROVED

ssl_txn_id=138FA6E57-3FBE-BBE5-8EE2-FBAE43C782D9

ssl_approval_code=N20032

ssl_cvv2_response=

ssl_avs_response=

ssl_account_balance=0.00

ssl_txn_time=08/17/2008 10:20:27 AM
```

Once again you have to make sure that python was compiled with SSL support. If it does not have SSL installed it will give you a protocol error. You can code something that will loop through the results as well.

Simple PHP mySQL Configuration Script

The following PHP page assumes that a mySQL database named `CFG_DB` exists with a table titled `ELAVON_CFG`. This script pulls the merchant information from the database and displays the data to the screen. The purpose of this script is to demonstrate pulling merchant configuration data from a mySQL database and table creation only.

Table Creation SQL Script:

```
CREATE TABLE ` ELAVON_CFG ` (
 `MERCH_ID` text NOT NULL,
 `MERCH_USER` text NOT NULL,
 `MERCH_PIN` text NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Table Creation PHP Script:

```
$sql = 'CREATE TABLE ` ELAVON_CFG ` ('
    . ' `MERCH_ID` TEXT NOT NULL, '
    . ' `MERCH_USER` TEXT NOT NULL, '
    . ' `MERCH_PIN` TEXT NOT NULL'
    . ' )'
    . ' TYPE = myisam';
```

Fields existing in the table are as follows:

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	MERCH_ID	text	latin1_swedish_ci		No			
☐	MERCH_USER	text	latin1_swedish_ci		No			
☐	MERCH_PIN	text	latin1_swedish_ci		No			

Table Population Script

```
INSERT INTO `elavon_cfg` ( `MERCH_ID` , `MERCH_USER` , `MERCH_PIN` )
VALUES (
'XXXXXX', 'XXXXXX', 'XXXXXX'
);
```

NOTE: The Values section above containing the XXXXX should be replaced with your actual merchant ID, merchant user, and merchant PIN.

Below is the php source code for ELAVON_CFG.php:

```
<?php session_start();
?>
<html>
<head>
<title>mySQL PHP ELAVON_CFG</title>
<meta name="generator" http-equiv="content-type"
content="text/html">
<style type="text/css">
 body {
  background-color: #FFFFFF;
  color: #004080;
  font-family: Arial;
  font-size: 12px;
 }
 .bd {
  background-color: #FFFFFF;
  color: #004080;
  font-family: Arial;
  font-size: 12px;
 }
 .tbl {
  background-color: #FFFFFF;
 }
 a:link {
  background-color: #FFFFFF01;
  color: #FF0000;
  font-family: Arial;
  font-size: 12px;
 }
 a:active {
  color: #0000FF;
  font-family: Arial;
  font-size: 12px;
 }
 a:visited {
  color: #800080;
  font-family: Arial;
```

```
 font-size: 12px;
}
.hr {
 background-color: #336699;
 color: #FFFFFF;
 font-family: Arial;
 font-size: 12px;
}
a.hr:link {
 color: #FFFFFF;
 font-family: Arial;
 font-size: 12px;
}
a.hr:active {
 color: #FFFFFF;
 font-family: Arial;
 font-size: 12px;
}
a.hr:visited {
 color: #FFFFFF;
 font-family: Arial;
 font-size: 12px;
}
.dr {
 background-color: #FFFFFF;
 color: #000000;
 font-family: Arial;
 font-size: 12px;
}
.sr {
 background-color: #FFFFCF;
 color: #000000;
 font-family: Arial;
 font-size: 12px;
}
</style>
</head>
<body>
```

```
<?php
 $conn = connect();
 $showrecs = 2;
 $pagerange = 10;

 $page = @$_GET["page"];
 if (!isset($page)) $page = 1;
 select();
 mysql_close($conn);
?>
</body>
</html>

<?php function select()
 {
 global $a;
 global $showrecs;
 global $page;

 $res = sql_select();
 $count = sql_getrecordcount();
 if ($count % $showrecs != 0) {
  $pagecount = intval($count / $showrecs) + 1;
 }
 else {
  $pagecount = intval($count / $showrecs);
 }
 $startrec = $showrecs * ($page - 1);
 if ($startrec < $count) {mysql_data_seek($res, $startrec);}
 $reccount = min($showrecs * $page, $count);
?>
<table class="bd" border="0" cellspacing="1" cellpadding="4">
<tr><td>Table: ELAVON_CFG</td></tr>
<tr><td>Records shown <?php echo $startrec + 1 ?> - <?php echo
$reccount ?> of <?php echo $count ?></td></tr>
</table>
<hr size="1" noshade>
```

```
<?php showpagenav($page, $pagecount); ?>
<br>
<table class="tbl" border="0" cellspacing="1"
cellpadding="5"width="100%">
<tr>
<td class="hr"><?php echo "MERCH_ID" ?></td>
<td class="hr"><?php echo "MERCH_USER" ?></td>
<td class="hr"><?php echo "MERCH_PIN" ?></td>
</tr>
<?php
 for ($i = $startrec; $i < $reccount; $i++)
 {
  $row = mysql_fetch_assoc($res);
  $style = "dr";
  if ($i % 2 != 0) {
   $style = "sr";
  }
?>
<tr>
<td class="<?php echo $style ?>"><?php echo
htmlspecialchars($row["MERCH_ID"]) ?></td>
<td class="<?php echo $style ?>"><?php echo
htmlspecialchars($row["MERCH_USER"]) ?></td>
<td class="<?php echo $style ?>"><?php echo
htmlspecialchars($row["MERCH_PIN"]) ?></td>
</tr>
<?php
 }
 mysql_free_result($res);
?>
</table>
<br>
<?php showpagenav($page, $pagecount); ?>
<?php } ?>


<?php function showpagenav($page, $pagecount)
{
?>
<table class="bd" border="0" cellspacing="1" cellpadding="4">
```

```
<tr>
<?php if ($page > 1) { ?>
<td><a href="ELAVON_CFG.php?page=<?php echo $page - 1
?>">&lt;&lt; Prev</a> </td>
<?php } ?>
<?php
 global $pagerange;
 if ($pagecount > 1) {
 if ($pagecount % $pagerange != 0) {
  $rangecount = intval($pagecount / $pagerange) + 1;
 }
 else {
  $rangecount = intval($pagecount / $pagerange);
 }
 for ($i = 1; $i < $rangecount + 1; $i++) {
  $startpage = (($i - 1) * $pagerange) + 1;
  $count = min($i * $pagerange, $pagecount);

  if ((($page >= $startpage) && ($page <= ($i * $pagerange)))) {
   for ($j = $startpage; $j < $count + 1; $j++) {
    if ($j == $page) {
?>
<td><b><?php echo $j ?></b></td>
<?php } else { ?>
<td><a href="ELAVON_CFG.php?page=<?php echo $j ?>"><?php echo $j
?></a></td>
<?php } } } else { ?>
<td><a href="ELAVON_CFG.php?page=<?php echo $startpage ?>"><?php
echo $startpage ."..." .$count ?></a></td>
<?php } } } ?>
<?php if ($page < $pagecount) { ?>
<td> <a href="ELAVON_CFG.php?page=<?php echo $page + 1
?>">Next &gt;&gt;</a> </td>
<?php } ?>
</tr>
</table>
<?php } ?>
<?php function connect()
{
```

```
 $conn = mysql_connect("databasehost", "username", "password");
 mysql_select_db("CFG_DB");
 return $conn;
}
function sql_select()
{
 global $conn;
 $sql = "SELECT MERCH_ID, MERCH_USER, MERCH_PIN FROM
`ELAVON_CFG`";
 $res = mysql_query($sql, $conn) or die(mysql_error());
 return $res;
}
function sql_getrecordcount()
{
 global $conn;
 $sql = "SELECT COUNT(*) FROM `ELAVON_CFG`";
 $res = mysql_query($sql, $conn) or die(mysql_error());
 $row = mysql_fetch_assoc($res);
 reset($row);
 return current($row);
} ?>
```

Simple .Net DB Configuration Script

```
<%@ LANGUAGE="JScript"%>
<HTML>
<HEAD>
<TITLE> Simple ASP.Net DB Configuration Script </TITLE>
</HEAD>
<%
 // makes the connection to the Database (ACCESS DB FOR TESTING)
var recordSet = Server.CreateObject("ADODB.RecordSet");
recordSet.Open("select * from ELAVON_CFG;" , "DSN=localhost");

// YOU MAY PREFER TO USE THIS TYPE OF CONNECTION FOR MYSQL
// Dim Connection
// Dim recordSet
// Dim SQL
// SQL = "SELECT * FROM ELAVON_CFG"
```

```
// Set Connection = Server.CreateObject("ADODB.Connection")
// Set recordSet = Server.CreateObject("ADODB.Recordset")
// Connection.Open "DSN=dsn_name"
// recordSet.Open SQL,Connection


%>
<body>
<center>
 <%
 while(!recordSet.EOF)
 {
 %>
 <form action="
https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do
" method="post" name="Configuration Form">
<table border="1">
<tr>
<td><input type="text" name="MERCH_ID"
value="<%=recordSet("MERCH_ID")%>"></td>
<td><input type="text" name="MERCH_USER"
value="<%=recordSet("MERCH_USER")%>"></td>
<td><input type="text" name="MERCH_PIN"
value="<%=recordSet("MERCH_PIN")%>"></td>
</tr>
<%
recordSet.MoveNext();
}
recordSet.Close();
recordSet = null;


%>
</script>


</table>
<input type="submit">
</form>
</center>
</BODY>
</HTML>
```

Chapter 12 Summary

To summarize the contents covered in this integration guide, keep in mind a few facts about VirtualMerchant:

1. VirtualMerchant is merely an application that resides on a Web server and will act as such. As long as you are following RFC 2818 with your requests, as well as sending the designated key / value pairs, VirtualMerchant will respond as defined.
2. Regardless of your platform or programming language, VirtualMerchant has no idea what you are using, as long as it supports HTTPS.
3. Validate all forms. Test the output of your script if you are getting time outs. Normally the output is the problem. VirtualMerchant provides one you can use at: `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/test_tran.do` The source code for one is also in the Development Tool section.
4. Know your language and the client emulation module you are using. Should you need integration support, call 1-800-377-3962. Please have the error that you received as well as the ability to send us your source code, should it be requested by the representative. You can also get support by e-mailing internetproductsupport@elavon.com.

Glossary

Address Verification (AVS)
The process of verifying customer addresses with the issuing bank to minimize fraudulent transactions.

Authorization
The process of having credit card, gift card, and PINLess debit transactions approved by the issuing bank through communication with the network.

Auto-Pend Transaction
A transaction option that automatically "Pends" sale transactions submitted through the VirtualMerchant payment form.

Auto-Settle
An option that automatically settles all "unpended" transactions and transactions not “Set To Review” in the Unsettled Transaction batch at a specified time each day.

Card Verification Value (CVV)
The process of verifying the Card Verification Value with the issuing bank to minimize fraudulent transactions. The CVV2 value is a three to four digit value that is printed in reverse italics on the back side of the card. This additional value is not embossed upon the front of the card, nor is it contained upon the magnetic stripe on back.

Comma-separated Value
A text file format in which all data elements within the files are separated by a comma. This format is also referred to as a comma-delimited file or CSV file.

Filter
A function that allows you to enter specific parameters to narrow a search for transaction information in a particular file. You can search for a specific card number, within a specific date range, etc.

Force Transaction
A previously authorized transaction that needs to be entered in the current batch.

GBOK Number
A successful settlement batch with the network.

Merchant Admin

The default user account for the VirtualMerchant account. The Merchant Admin User ID (MA) is the same as the VirtualMerchant Account ID. This special user which cannot be deleted, always has all user rights and all terminal associations assigned to it.

Partial Approvals

Merchants can systemically conduct split-tender purchases by allowing debit and Pre-Paid card issuers to approve a portion of the original transaction amount in the authorization request when the transaction amount exceeds the funds available on the card. The merchant can then obtain the remainder of the purchase amount in another form of payment.

Partial Auth Capability

The POS application is capable of submitting one amount for authorization understanding that only part of the requested amount was approved. For example: $100.00 purchase, where $75.00 is approved on a credit card. The balance of $25.00 is understood to be still outstanding to complete the purchase.

Peer User

A user who shares the same supervisor as you.

Pend Transaction

A transaction status option that will not allow the transaction to be submitted for settlement. To allow the transaction to be submitted for settlement, the status of the transaction must be changed to “Unpended."

Refund Transaction

A transaction used to refund a previous purchase.

Reversals

A real-time transaction used to cancel an open authorization and restore the cardholders open to buy for the full amount previously authorized. This transaction is usually initiated when the cardholder decides that they do not want to proceed with the transaction. Reversals will free up cardholders’ open to buy amounts by reducing issuer holds on available balances when transactions are not completed; therefore reducing declines at the point of sale and the amount of cardholder complaints that are unpleasant for all parties involved.

Sale Transaction

A transaction in which an authorization is obtained and the transaction is entered into the unsettled batch.

Scope of User Rights

Virtual Terminal and Terminal Setup rights apply to your ability to do things in the context of any terminal in your Terminal Associations list. User Management rights apply to your ability to do things to your subordinates and to your peers’ subordinates. If you have the Edit Terminal Associations right, you may only add terminal associations that are assigned to you.

Settlement Process

The process of sending a batch of previously authorized transactions for settlement to the network.

Split Tender
Split Tender means that more than one form of tender (payment type) can be initially designated to be used to complete a single purchase. For example: $100.00 purchase, where $75.00 is paid in cash and $25.00 is paid by check, and the POS application knows that this was the two full amount tenders being used.

Subordinate
This is anyone who is directly below you in the user hierarchy or any of their subordinates.

Supervisor
This is the person directly above you in the user hierarchy.

Tab-delimited Value
A text file format in which all data elements within the file are separated by the Tab character.

Terminal Association
Where your user rights refer to a task you can do involving a terminal (i.e., make a sale or settle a transaction). Your user ID must be associated with that terminal and you must have selected that terminal context in VirtualMerchant. See the chapter on User Management for details on how to make or edit Terminal Associations in the VirtualMerchant User Guide.

Terminal Friendly Name
Terminals are referred to in VirtualMerchant by a Friendly Name configured by Elavon's Internet Product Support, for instance, "Website Terminal."

Terminal ID
A number used to identify the source of a transaction to the network. This corresponds to a physical credit card terminal in a traditional POS solution, but for VirtualMerchant, this is a virtual ID. You may have more than one terminal for use within your VirtualMerchant account. Each Terminal ID (TID) is associated with certain features as dictated by your merchant agreement. "Merchant Information" in Terminal Setup can be different for each terminal so that, for instance, the address printed on a receipt is correct for that location. See the chapter on Terminal Setup for details on configuring your terminal in the VirtualMerchant User Guide.

Unpend Transaction
A transaction status option that allows the transaction to be submitted for settlement. To prohibit the transaction from being submitted for settlement, the status must be set to "Pended."

User Account
The user you use to sign in to VirtualMerchant. The user ID is case sensitive.

User Rights
The tasks that the User Account can do in VirtualMerchant. There are three areas of User Rights: Virtual Terminal, User Management and Terminal Setup. See the section on User Management for details on how to make or edit User Rights in the VirtualMerchant User Guide.

VirtualMerchant Account
The VirtualMerchant Account your company has with Elavon.