

Converge

Developer Guide

Revision Date: September 2014

Copyright

Copyright © 2014 Elavon, Incorporated. All rights reserved. No part of this publication may be reproduced or distributed without the prior consent of Elavon, Inc., Two Concourse Parkway, Suite 800, Atlanta, GA 30328.

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 improvement and/or changes in the product(s) and/or programs(s) described in this publication at any time.

Trademarks

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

Preface

This document describes step by step procedures on how to use your developer guide including:

- Getting started
- Transaction security
- Transaction format
- Integration references
- Additional processing options
- Authorization response codes
- Code samples

This document is intended for all users of the Converge product and contains the information necessary for you to be able to use all the features of the product effectively.

Typographical Conventions

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

- **Bold text**

Indicates a menu option, a window title, buttons, and so on that you can use to identify a part of the user interface.

Examples:

Print or **Save As** dialog box

- **Menu selection sequences**

Indicates a series of menu options that you need to select in a particular sequence and listed in one step. Each menu option is separated by a pipe (|).

1. Choose **File | Save As | File Name** and enter the name of the document.

- **Courier text**

Indicate examples of software code. Usually this type of text is encapsulated in a code box as illustrated below.

```
Begin Header
  <head>
    <title>Batch Import</title>
  </head>
End Header
```

- **Bold courier text**

Indicates a command that you would type into a command prompt window as illustrated below.

```
cd c:\users\
```

- **Italicized text**

Indicates that the word or phrase is:

- A reference to another document as illustrated below.

Refer to the *Elavon User Guide*.

- Emphasized for clarification as illustrated below.

You *do not* need to select **Apply**.

- The word is replacement text, such as a variable for a piece of code that you need to enter the appropriate value for your implementation as illustrated below.

```
<xml>
  <country_code>Country Code</country_code>
</xml>
```

Related Documentation

The following documents are available related to the Converge product.

- Converge Getting Started guide
- Converge System Administration Guide
- Converge Transaction Processing Guide

Revision History

The following table provides a description of the changes made to this document from its origination to the current release.

| Revision | Date | Revision Notes |
|----------|----------|--|
| A | SEP-2014 | Original release of the Converge Developer Guide |

Table of Contents

| | |
|--|----|
| Chapter 1: Introduction | 1 |
| Converge Features | 1 |
| API Overview | 2 |
| Implementation Guidelines | 3 |
| Integration Checklist | 4 |
| Chapter 2: Getting Started..... | 5 |
| Getting a Unique Test Account | 5 |
| URLS | 6 |
| Demo URLs..... | 6 |
| Production URLs | 6 |
| Communicating with Converge..... | 7 |
| Authentication | 8 |
| Payment Forms | 9 |
| Merchant Payment Form | 9 |
| Converge Payment Form | 12 |
| Receipt Forms | 16 |
| Merchant Receipt..... | 16 |
| Converge Receipt | 18 |
| Export Scripts | 19 |
| Chapter 3: Transaction Security..... | 23 |
| Common Fraudulent Activities | 23 |
| Best Practices and Compliance Checklist..... | 24 |
| PCI DSS Guidelines | 29 |
| External Security Resources..... | 30 |
| Chapter 4: Transaction Format..... | 32 |
| Credit Card Transactions..... | 34 |
| Credit Card Sale (ccsale)..... | 36 |

| | |
|---|-----|
| Credit Card Auth Only (ccaonly) | 58 |
| Credit Card AVS Only (ccavonly) | 67 |
| Credit Card Verification (ccverify) | 72 |
| Credit Card Return/Credit (ccccredit) | 76 |
| Credit Card Force (ccforce) | 81 |
| Credit Card Balance Inquiry (ccbalinquiry) | 87 |
| Credit Card Generate Token (ccgettoken) | 89 |
| Credit Card Return (ccreturn) | 92 |
| Credit Card Void (ccvoid) | 94 |
| Credit Card Completion (cccomplete) | 99 |
| Credit Card Delete (ccdelete) | 103 |
| Credit Card Update Tip (ccupdatetip) | 104 |
| Credit Card Signature (ccsignature) | 107 |
| Recurring Transactions | 110 |
| Credit Card Add Recurring Transaction (ccaddrecurring) | 110 |
| Credit Card Update Recurring Transaction (ccupdaterecurring) | 119 |
| Credit Card Delete Recurring Transaction (ccdeleterecurring) | 122 |
| Credit Card Submit Recurring Payment (ccrecurringsale) | 123 |
| Installment Transactions | 125 |
| Credit Card Add Installment Transactions (ccaddinstall) | 126 |
| Credit Card Update Installment Transactions (ccupdateinstall) | 132 |
| Credit Card Delete Installment Transactions (ccdeleteinstall) | 135 |
| Credit Card Submit Installment Payment (ccinstallsale) | 136 |
| Batch Import Transactions | 138 |
| Credit Card Batch Import (ccimport) | 139 |
| Credit Card Information Batch Import (cctokenimport) | 146 |
| Credit Card Recurring Batch Import (ccrecimport) | 151 |
| Debit Card Transactions | 158 |
| Debit Purchase (dbpurchase) | 159 |
| Debit Return (dbreturn) | 162 |
| Debit Inquiry (dbbainquiry) | 163 |

| | |
|---|-----|
| EBT Transactions | 165 |
| Food Stamp Purchase (fspurchase)..... | 165 |
| Food Stamp Return (fsreturn)..... | 167 |
| Food Stamp Inquiry (fsbainquiry) | 169 |
| Food Stamp Force Purchase (fsforcepurchase) | 170 |
| Food Stamp Force Return (fsforcereturn) | 172 |
| Cash Benefit Purchase (cbpurchase)..... | 173 |
| Cash Benefit Inquiry (cbbainquiry) | 175 |
| Gift Card Transactions..... | 177 |
| Gift Card Activation (egcactivation)..... | 177 |
| Gift Card Sale/Redemption (egcsale)..... | 179 |
| Gift Card Refund (egccardrefund)..... | 182 |
| Gift Card Replenishment/Reload (egcreload)..... | 184 |
| Gift Card Balance Inquiry (egcbainquiry) | 186 |
| Gift Card Credit (egccredit) | 188 |
| Gift Card Generate Token (egcgettoken)..... | 191 |
| Electronic Check Transactions | 192 |
| Electronic Check Purchase (ecspurchase)..... | 192 |
| Electronic Check Void (ecsvoid)..... | 196 |
| PINLess Debit Transactions..... | 197 |
| PINLess Debit Purchase (pldpurchase) | 197 |
| Cash Tender Transactions | 198 |
| Cash Sale (cashsale) | 199 |
| Cash Return/Credit (cashcredit) | 201 |
| Card Manager Transactions | 203 |
| Token Query (ccquerytoken) | 203 |
| Token Update (ccupdatetoken) | 205 |
| Token Delete (ccdeletetoken)..... | 207 |
| End of Day Transactions..... | 209 |
| Transaction Query (txnquery)..... | 209 |
| Transaction Email (txnemail) | 217 |

| | |
|--|-----|
| Settle (settle)..... | 218 |
| Account Admin Transactions | 222 |
| Terminal Setup (terminalsetup)..... | 222 |
| Payment Field (fieldsetup)..... | 228 |
| Printer Setup (printersetup) | 232 |
| Chapter 5: Integration Reference | 235 |
| Supported Transaction Input Fields..... | 235 |
| Fraud Prevention Matrix..... | 250 |
| ISO Country Codes | 251 |
| ISO Currency Codes..... | 255 |
| Chapter 6: Additional Processing Options | 257 |
| 3D Secure | 257 |
| Transaction Flow..... | 258 |
| Transaction Format..... | 267 |
| Transaction Examples | 270 |
| MasterPass..... | 273 |
| Transaction Flow..... | 274 |
| Transaction Examples | 279 |
| Dynamic Currency Conversion (DCC)..... | 281 |
| Transaction Flow..... | 282 |
| Transaction Examples | 283 |
| Receipt Requirements..... | 288 |
| Multi-Currency Conversion (MCC)..... | 291 |
| Transaction Flow..... | 292 |
| Transaction Examples | 294 |
| Tokenization..... | 295 |
| Generating Tokens..... | 297 |
| Processing Transactions Using Tokens..... | 301 |
| Managing Stored Tokens | 303 |
| Transaction Flow..... | 303 |
| Transaction Examples | 312 |

| | |
|---|-----|
| Encryption | 322 |
| Electronic Check ACH ECheck | 324 |
| ACH Types | 325 |
| Transaction Flow | 327 |
| Transaction Examples | 329 |
| Best Practices | 331 |
| Chapter 7: Authorization Response Codes | 333 |
| Credit Card Response Codes | 333 |
| Electronic Gift Card (EGC) Response Codes | 334 |
| AVS Response Codes | 335 |
| CVV2/CVC2 Response Codes | 336 |
| Error Codes | 336 |
| Chapter 8: Code Samples | 345 |
| Perl Sample | 345 |
| PHP Sample | 347 |
| Python Sample | 351 |
| HTML Sample | 352 |
| Glossary | 363 |

Chapter 1: Introduction

The Converge application is a secure, server-based system that supports transaction processing (authorization and settlement) in real-time. The Converge API uses a pseudo-XML or key value pairs implementation.

You can submit transactions to the Converge application using one of the following methods:

- Virtual Terminal (Converge Web interface or Converge Mobile application)
- Converge API through the use of an integrated application

This guide provides the information necessary to complete a successful Converge integration including:

- Transaction security
- Transaction format
- Transaction reference
- Additional processing options
- Authorization response codes
- Code samples

Converge Features

The Virtual Terminal enables you to use a standard Web browser to process transactions as a cost-effective payment solution. Using this application, you can:

- Manage your payment account
- Submit transactions
- Monitor and review unsettled transactions
- Search for and view settled transactions
- Configure account settings

For help with the Merchant Interface features and settings, refer to the *Converge System Administration Guide*.

API Overview

The Converge API enables you to write a point-of-sale application (website, software application, shopping cart, and so on) that interfaces with the Converge payment gateway to process the following:

- Full range of payment types including:
 - Credit card
 - Debit card
 - Food stamp
 - Cash benefit
 - Electronic check
 - Gift card
 - Cash tender
- Recurring and installment transactions
- Batch file transactions (a group of individual transaction requests combined into a file and sent in a single request)
- Batch management transactions (including query and settlement)
- Admin transactions (account setups)

The integration to the Converge gateway supports transactions for both card present and non-present environments, including the following market segments:

- E-Commerce
- Mail order
- Retail
- Service

Typically, customers use a variety of methods to integrate to the Converge gateway. For example, you could:

- Submit as little as four pieces of data from your application. By using the settings that have been configured in the Converge administration section, you can gather the rest of the necessary information directly from the customer.
- Use Converge as a backend feature to your integrated application, completely transparent to your customers. You can write 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.

Most merchants fall somewhere in the middle of these scenarios. This combination of methods involves gathering some data from the customer before referring them to the Converge application. This application gathers more information from them and displays the receipt after the transaction has been approved.

This guide focuses on the processes and settings available to you to integrate to the Converge payment gateway.

Implementation Guidelines

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

- Converge is 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 XML or key value pairs, Converge will respond as defined.
- The programming language used for processing using Converge is inconsequential, provided your chosen language supports post over HTTPS.
- Choose the integration type that makes sense for your programming capabilities and business needs, for instance a website that wants to have Converge collect the information should be using `process.do` with form set to true in order to call the Converge payment page.
- Select a payment form. You will need to decide if you want to use Host payment data form or built in payment form to collect card data.
- Validate all forms and test the output of your scripts. If you are experiencing issues, try sending your transaction to the following link which will help us identify the root cause of the issue: techsupp@elavon.com
- Converge does not expect a fully validated XML document but an XML formatted request. An XML request is simply a data string formatted in XML syntax. Only the Converge specific elements for the transaction itself are supported, as defined in this developer's guide and no encoding statement, additionally the XML data provided is assigned to a variable called `xmldata`.
- Know your language and the client emulation module you are using. Should you need integration support, call 1-800-377-3962, option 2 then option 2 (in Canada you are asked to choose either English or French for you language). 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 emailing internetproductsupport@merchantconnect.com.

Integration Checklist

1. Demo account received from Software Technical Support (techsupp@elavon.com)
2. Work with an Elavon sales representative to begin the process of establishing an Elavon merchant account
3. Decide whether or not you wish to use the Converge hosted payment form, or host the page that collects the card data on your application
4. Develop the integration using HTTPS POST requests to the Converge demo site:
 - `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do` for key value pairs single transaction request
 - `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/processBatch.do` for key value pairs batch import request
 - `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/processxml.do` for xml single transaction request
 - `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/accountxml.do` for xml Admin transaction request
5. Receive production Converge processing credentials
6. Demo account does not automatically map to production. Configure production account payment fields, business rules, and other fields to match demo account setup.
7. Replace `ssl_merchant_id`, `ssl_user_id`, and `ssl_pin`, with production information, and change POST target to:
 - `https://www.myvirtualmerchant.com/VirtualMerchant/process.do` for key value pairs single transaction request
 - `https://www.myvirtualmerchant.com/VirtualMerchant/processBatch.do` for key value pairs batch import request
 - `https://www.myvirtualmerchant.com/VirtualMerchant/processxml.do` for xml
 - `https://www.myvirtualmerchant.com/VirtualMerchant/accountxml.do` for xml Admin transaction request
8. Test production integration using live card (normal processing fees apply)
9. Work with sales rep to make sure all underwriting requirements are met to get hold removed from funding.
10. Go live

Chapter 2: Getting Started

This chapter provides some basic information that you may need before you are able to use Converge.

Topics include:

- Getting a unique test account
- URLs
- Communicating with Converge
- Authentication
- Payment forms
- Receipt forms

Note: 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](#) chapter.

Getting a Unique Test Account

Prior to beginning Converge integration, integrators must request a unique test account with the Enable HTTPS Transaction option enabled, to be able to perform transactions from an integrated solution. In addition, the Enable HTTPS Batch Import option must be enabled in order to process batch files.

Contact Elavon Internet product support group at techsupp@elavon.com or 1-800-377-3962, option 2, option 2 (in Canada you are asked to choose either English or French for you language) to submit your request for a test account.

The following information must be provided to the support group:

- Company name
- Primary contact name
- Primary contact phone
- Primary email address

Notes:

- The Virtual Terminal can be accessed by logging to <https://demo.myvirtualmerchant.com/VirtualMerchantDemo/login.do> while testing, once integration testing has been completed and you are ready to begin processing production transactions.
 - You must login to the production environment to retrieve your production credentials at <https://www.myvirtualmerchant.com/VirtualMerchant/login.do>.
 - Username and Passwords are case sensitive (the system differentiates between upper- and lower-case characters).
-

URLS

All reference of URLs [Insert URL Here] in the samples must be replaced with the following:

Demo URLs

- `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do` for key value pairs formatted single request.
- `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/processBatch.do` for key value pairs formatted batch request.
- `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/processxml.do` for XML formatted single transactions.
- `https://demo.myvirtualmerchant.com/VirtualMerchantDemo/accountxml.do` for single Admin request.

Production URLs

Once integration testing has been completed and you are ready to begin processing production transactions, you must ensure that your integrated solution is pointed to the production environment and pass your unique production credentials posting to the following URLs:

- `https://www.myvirtualmerchant.com/VirtualMerchant/process.do` for key value pairs formatted single request.
- `https://www.myvirtualmerchant.com/VirtualMerchant/processBatch.do` for key value pairs formatted batch request.
- `https://www.myvirtualmerchant.com/VirtualMerchant/processxml.do` for XML formatted single transactions.
- `https://www.myvirtualmerchant.com/VirtualMerchant/accountxml.do` for XML formatted single Admin requests.

Communicating with Converge

Converge accepts information sent using Hypertext Transfer Protocol Secure (HTTPS) over POST method. The data you send, along with Converge settings, will determine:

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

Converge currently supports two different ways to integrate:

- Key value pairs formatted request using `process.do` (for a single transaction) or `processBatch.do` (for a batch file) with the following syntax: `ssl_name_of_field = value of field` (example: `ssl_amount = 1.00`)

Or

- XML formatted request using `processxml.do` (for a single transaction) or `accountxml.do` (for a Admin request), the transaction data formatted in XML syntax must include all supported transaction elements nested between one beginning and ending element `<txn>`, the data is contained within the `xmldata` variable.

The following is an example of a fully formatted XML request:

```
xmldata=<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_tra
nsaction_type><ssl_card_number>00*****0000</ssl_card_number><ssl_exp
date>1215</ssl_exp_date><ssl_amount>10.00</ssl_amount><ssl_cvv2cvc2_indic
ator>1</ssl_cvv2cvc2_indicator><ssl_cvv2cvc2>123</ssl_cvv2cvc2><ssl_first
_name>Test</ssl_first_name></txn>
```

Notes:

- 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 Converge. Virtual Terminal 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 key value pairs or XML input fields, refer to the [Supported Transaction Input Fields](#) section.
-

Authentication

The API fields that comprise the sensitive processing credentials and are required to be passed for each transaction are:

| Field Name | Description |
|-----------------|---|
| ssl_merchant_id | Converge ID as provided by Elavon |
| ssl_user_id | Converge user ID as configured on Converge (case sensitive) |
| ssl_pin | Converge PIN as generated within Converge (case sensitive) |

Important:

- The Merchant Admin (MA) user ID cannot be used.
 - It is *strongly* recommended that you create a user ID specifically for the API. This allows more accurate tracking of how transactions occur and who is submitting them, as well as protects you in the event of a security compromise by limiting what transaction types the user ID can process.
 - It is *strongly* recommended that your user PIN is 32 or 64 characters long.
-

Each merchant account has one Merchant Admin user called the MA user, which is identical to your Converge ID (VID) and can also have multiple standard users. When specifying a user ID in the transaction request, make sure that the PIN matches the user ID that you are passing for the desired terminal you wish to process transactions.

Each VID can have multiple UIDs, each UID will have a unique PIN per terminal assigned (hierarchy).

The `ssl_user_id` cannot be omitted and should be passed along with a correct PIN for all transactions even if that user ID is the Merchant Admin user. The application will validate that the correct VID (`ssl_merchant_id`), user ID (`ssl_user_id`), and PIN (`ssl_pin`) combination has been passed for each transaction.

When an account has more than one terminal, it is the combination of `ssl_merchant_id`, `ssl_pin`, and `ssl_user_id` that Converge uses to determine which terminal the transaction is processed under.

It is best that the user ID used for the API is a separate user from the one used to login to the Converge application user interface. It is best that you never use your API user to login to the application.

All sensitive data, specifically your Converge credentials, must be placed in server side code rather than placing hidden value fields on an HTML form. This will limit the ability of malicious users to edit and use this data for their own fraudulent purposes. 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. This 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 using server-side scripting with Converge integration is the ability to hide the sensitive processing credentials from the browser.

Payment Forms

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 Converge 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 Converge for `process.do` and `processxml.do`.

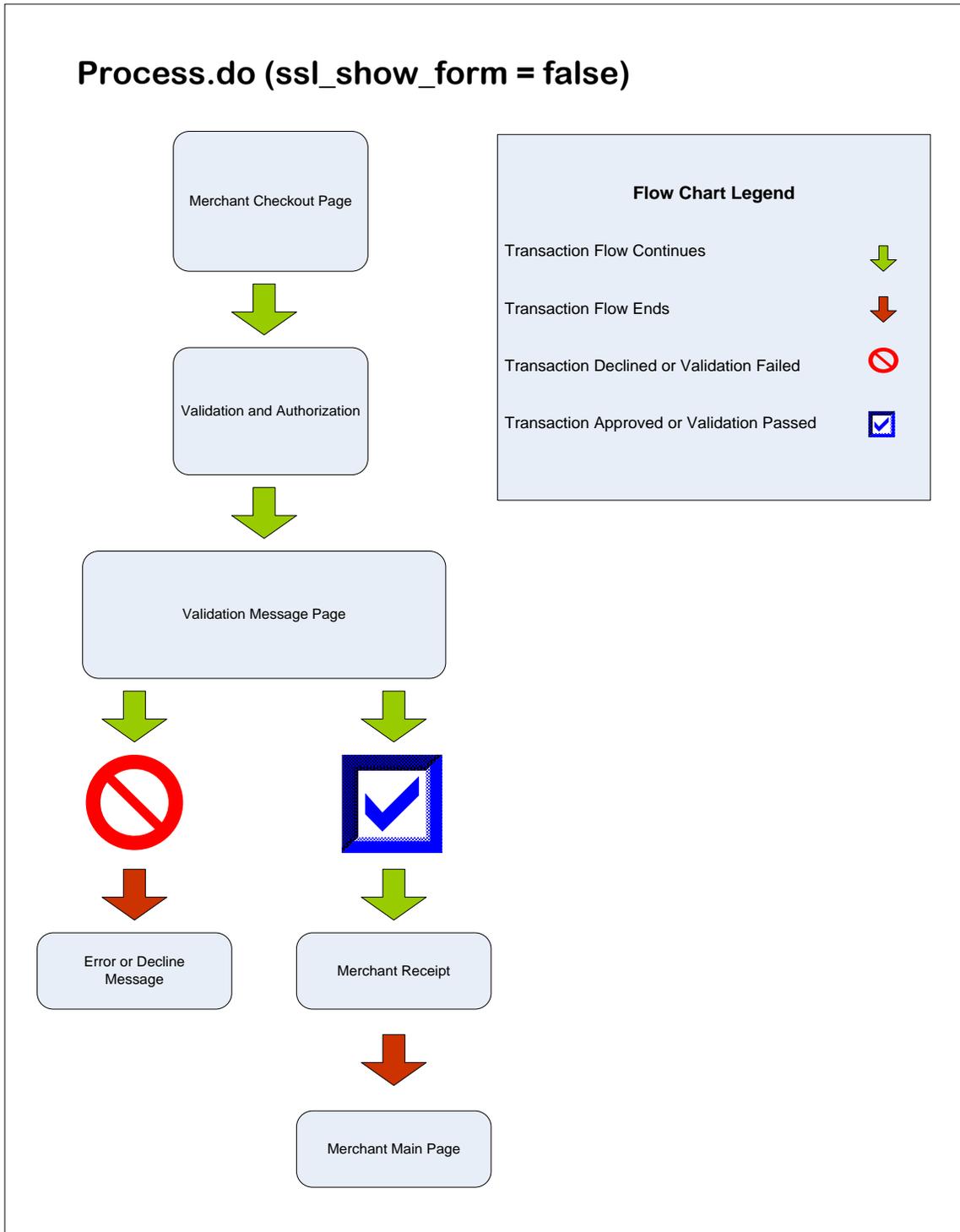
- **Converge:**

If Converge 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. The Converge payment form is applicable to `process.do` only.

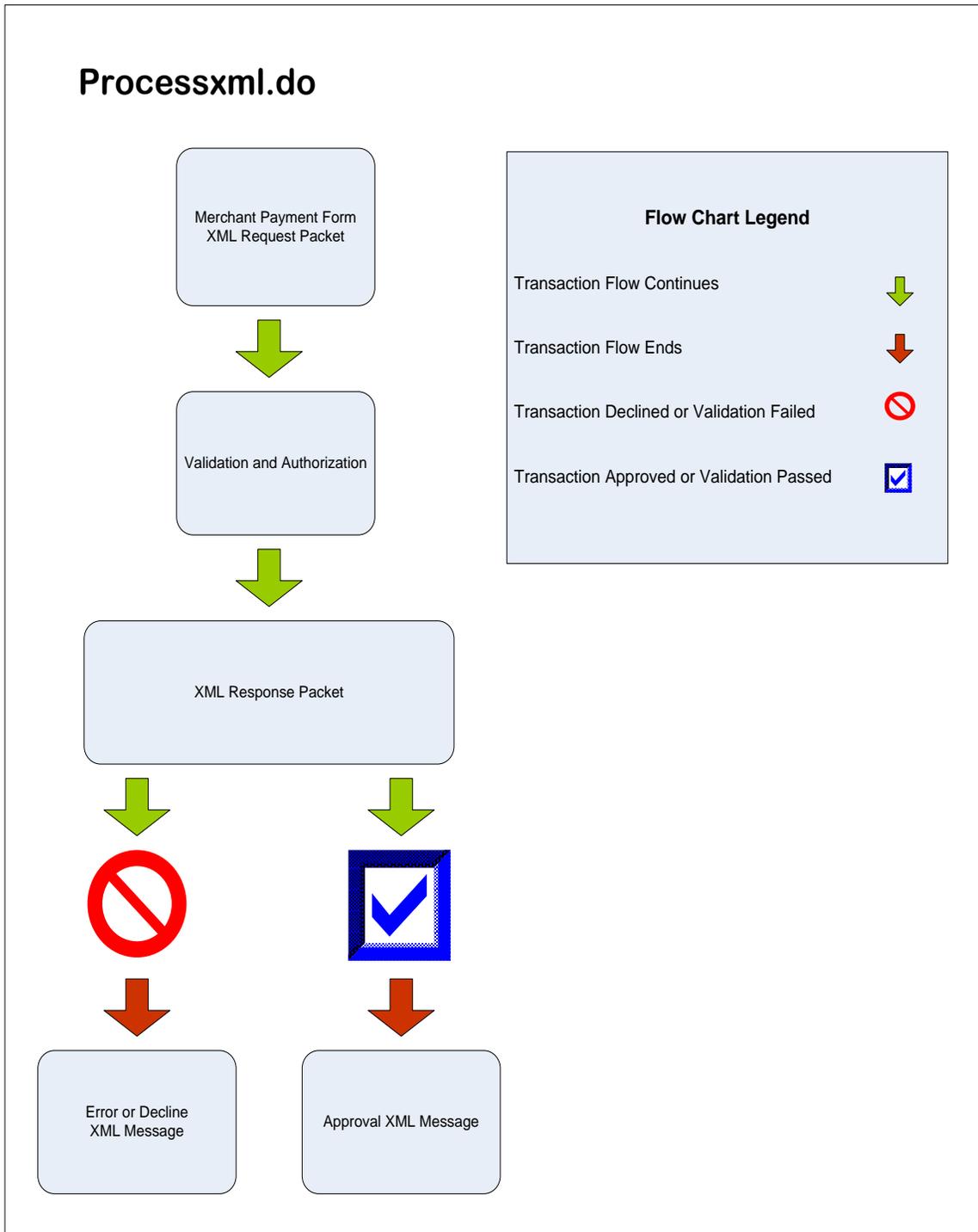
Merchant Payment Form

This section explains how to send information for Converge 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 Converge 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** when using `process.do`.



When using `processxml.do`, `processBatch.do`, and `accountxml.do`, the `ssl_show_form` property does not apply.



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

| Field Name | Description |
|----------------------|---|
| ssl_merchant_id | Converge ID as provided by Elavon |
| ssl_user_id | Converge user ID as configured on Converge (case sensitive) |
| ssl_pin | Converge PIN as generated within Converge (case sensitive) |
| ssl_transaction_type | Transaction type |
| ssl_show_form | Set to false for <code>process.do</code> |

Important: The Merchant Admin (MA) user ID cannot be used.

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

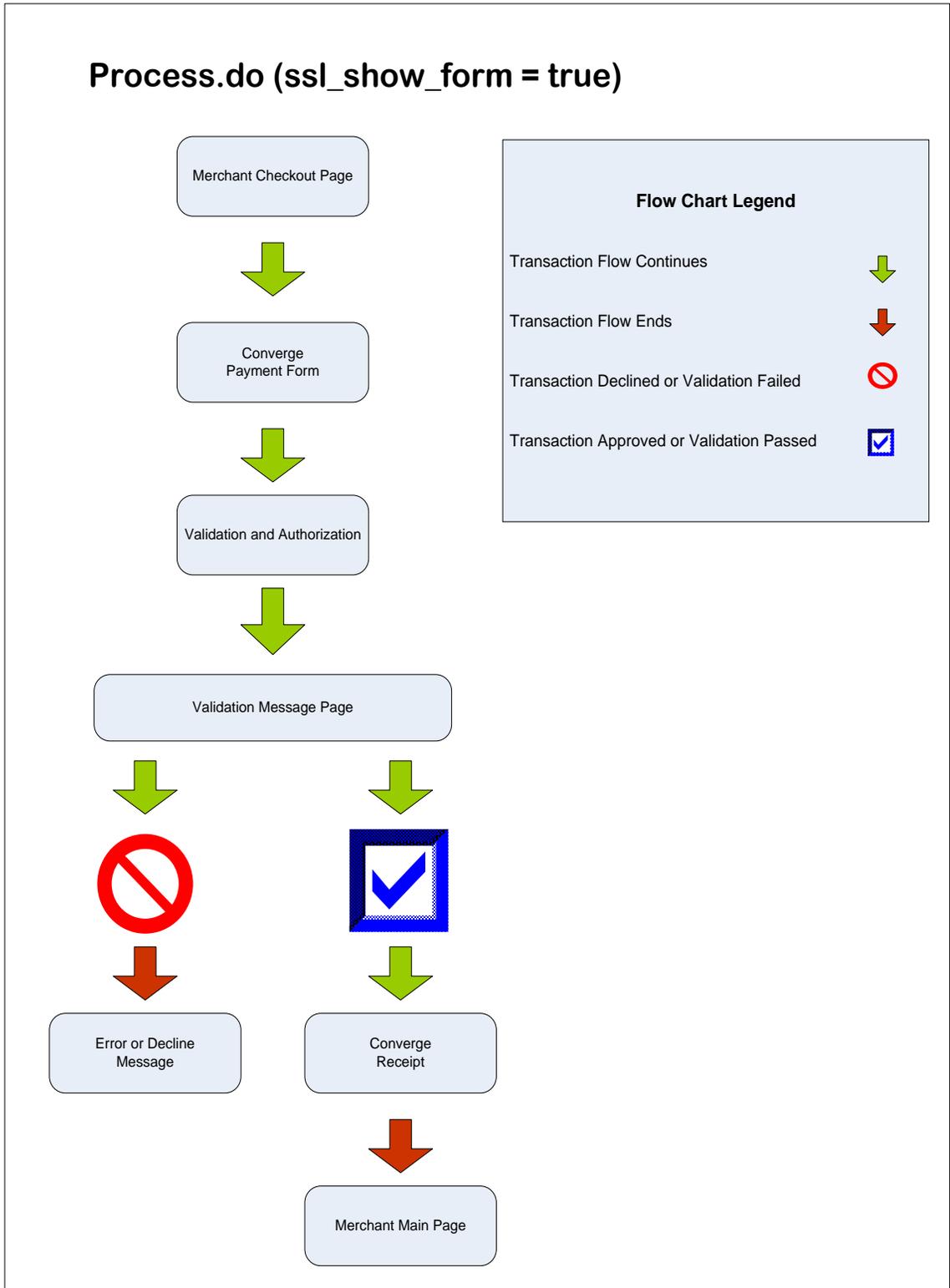
| Field Name | Description |
|-----------------|--|
| ssl_track_data | Track data (required for swiped transactions) |
| ssl_card_number | Card number (required for hand-keyed transactions where the track data is not present) |
| ssl_exp_date | Expiration date (required to be used with card number on hand-keyed transactions) |
| ssl_amount | Transaction amount |

There are also conditional fields that should be supplied based on Converge 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.

Converge Payment Form

This section explains how to send information to have Converge 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 Converge account.

The `ssl_show_form` property must be set to **true** and it is only available through `process.do`.



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

| Field Name | Description |
|-----------------------------------|---|
| <code>ssl_merchant_id</code> | Converge ID as provided by Elavon |
| <code>ssl_user_id</code> | Converge user ID as configured on Converge (case sensitive) |
| <code>ssl_pin</code> | Converge PIN as generated within Converge (case sensitive) |
| <code>ssl_transaction_type</code> | Transaction type |
| <code>ssl_show_form</code> | Set to true for <code>process.do</code> |

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, Converge 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 Converge account.

If you want to integrate Converge 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 a script on your server that will send a POST request to the `process.do` URL using cURL or an equivalent. Refer to the [URLS](#) section to set your URL for either the demo or production environment.
3. Collect as much or as little transactional data on the website as needed and pass the values through the POST to your server-side script. Example: Pass an amount to your script, but allow the customer to fill out the Converge hosted payment form with their contact and shipping information and credit card data.
4. On the server-side script, collect the information from the POST request (typically using `$_POST` variables or an equivalent) and include the `ssl_merchant_id`, `ssl_user_id`, and `ssl_pin`.

Note: Do not include your `ssl_merchant_id`, `ssl_user_id`, and `ssl_pin` in hidden fields on the website.

5. Set the transaction type you wish to perform `ssl_transaction_type` to `ccsale` to perform a sale or `ccauthonly` for an authorization.

6. Call Converge's `process.do` through cURL or the equivalent and Converge will return the source code for your payment form, or the response as indicated by the `ssl_show_form` value to the customer's browser.
7. Set the value of the `ssl_amount` so that it is unable to be changed on the payment form (unless you are accepting donations).
8. Add a **Submit** button on your website.

Note: Once integration testing has been completed and you are ready to begin processing production transactions, you must ensure that your integrated solution is pointed to the production environment (<https://www.myvirtualmerchant.com/VirtualMerchant/process.do>) and passing your unique production credentials.

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

The image shows a web form titled "SALE" with three main sections: "Order Section", "Billing Address", and "Shipping Address".

- Order Section:** Contains fields for Account Data (50*****3003), Expiration Date (MMYY: 1209), Amount (5.00), CVC2, Customer Code, Sales Tax, and Invoice Number. Red asterisks indicate required fields.
- Billing Address:** Contains fields for Company, First Name, Last name, Address1, Address2, City, State/Province, Postal Code, Country, Phone, and Email Address.
- Shipping Address:** Contains fields for Ship to Company, Ship to First Name, Ship to Last name, Ship to Address1, Ship to Address2, Ship to City, Ship to State/Province, Ship to Postal Code, Ship to Country, and Ship to Phone.

A "Process" button is located at the bottom center of the form.

Receipt Forms

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

- **Merchant Receipt:**

If you draw your own receipt, your form must handle the data received from Converge to correctly communicate to your customers the outcome of their transactions.

- **Converge Receipt:**

If Converge draws the receipt for you, you do not need to include logic to parse through the Converge result. However, your customer might not return to your website when the transaction is complete.

Additionally the application will allow you to specify an alternative destination where the response is sent to:

- **Export Script:**

Converge will send the receipt/ response via an export script to the destination of your choice a back end process used mainly when adjusting inventory in real time, it is transparent to your consumer. You do not need to include logic to parse through the Converge result. The script is fired as a backup to the response. When Converge sends the receipt or response to your integrated application, it will also fire an export script of that response to specified URL.

Merchant Receipt

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

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 Converge.

Using the various `ssl_receipt_link_url` variables, Converge 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`

Converge Receipt

This section shows you how to have Converge 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 Converge website. Refer to the *Converge System Administration Guide* for more information about how to use the Converge 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 (RE-Direct 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 Converge 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 Converge 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.

Export Scripts

Some merchants request that the results of their payment transactions be returned to their website for inventory purposes, sales analysis, or customer database maintenance. Converge offers the ability to do so through an Export script.

Export scripts are an asynchronous authorization response method, allowing you to designate an alternate destination for the transaction response. In addition to returning a response to you, Converge will also send that response to the export destination of your choice. Converge will send an Export script and then posts the transaction results to your system after completing each transaction in Real-time.

Export script generation uses default encoding ISO-8859-1. Converge will send an Export script and then posts the transaction results to your system after completing each transaction in Real-time. If you setup a website without username and password, Converge will just post the response to the destination, but if you have setup a secure Export script (website that require authentication), Converge first will authenticate the script using the username and password from the setup then post the transaction response. If Converge doesn't provide the correct credentials, it is not allowed to post the script to your system. The export script that gets sent back to you is the same whether you are doing authentication or not. The only difference between the Export script with authentication and non-authentication is the handshake. Once the handshake is established the export script is then sent.

The Export script if set up will be run following completion of a transaction (approvals, declines, errors). It returns results of a payment to your web server using a standard web protocol HTTP to *call* a page on your server just as a browser calls any web page. The Export script dumps data about the transaction response to the web page using a form POST.

Export Script Setup:

To initiate export scripts on your payments, log into the Converge Virtual Terminal and select the **Terminal | Advanced | System Setup** option under the **Export Options** section.

Note: Export scripts are permission based feature.

You can enter any of three URLs:

- One to specify where the approvals should be sent
- One to specify where the declines should be sent
- One to specify where the errors should be sent

These URLs can all be the same, or they can be different. You should be using a secure server protected by a security certificate. Simply enter the URL (for example, `https://www.ismerchant.com`).

If you wish, the web page used can be secured by regular web page authentication. To do this, simply specify the username and password required to access the page.

Fill in an approval URL entry if you wish information to be exported only for those transactions for which the system received an approval. Otherwise a script will fire for both approved and declined transactions.

We have built in an additional verification process to allow you to submit a confirmation string that our system will look for in each Export script response. If our system does not detect this string, it will then issue an Alert Email to advise you that the export script has failed. If you wish to use a confirmation string, enter it where indicated. If you wish to provide a confirmation string, it must be used in conjunction with the username and password.

When you have completed all fields, you must click **Update** to save your settings.

Once you have set up the export script with the correct web page, it will start sending the transaction data to that page after each applicable transaction has been processed.

Export Script Examples:

This is an example of the script that gets posted back to the merchant destination
(<https://www.merchantinventory.com/postscriptshere>)

```
POST https://www.merchantinventory.com/postscriptshere HTTP/1.1Content-
Length:803Host:www. merchantinventory:-1Content-Type: application/x-www-
form-urlencodeduser-agent:Apache-HttpClient/4.2.2 (java
1.5)authorization:{}ssl_email=&ssl_cvv2_response=M&Custom3=&ssl_ship_to_p
hone=&ssl_last_name=&Custom1=&Custom2=&ssl_ship_to_country=&ssl_ship_to_s
tate=&ssl_account_balance=4.00&ssl_ship_to_zip=&ssl_company=&ssl_result_m
essage=APPROVAL&ssl_country=&ssl_city=&ssl_phone=&ssl_invoice_number=&ssl
_ship_to_address2=&ssl_ship_to_address1=&ssl_transaction_currency=USD&ssl
_txn_id=AA4843B-8464D502-E195-46B7-838C-
160E218558CF&ssl_result=0&ssl_ship_to_company=&ssl_avs_response=&ssl_tran
saction_type=SALE&ssl_approval_code=CVI463&ssl_ship_to_last_name=&ssl_avs
_zip=&ssl_ship_to_city=&ssl_dynamic_dba=Sch.Med*&ssl_exp_date=1215&ssl_sh
ip_to_first_name=&ssl_avs_address=&ssl_salestax=&ssl_description=&ssl_add
ress2=&ssl_first_name=&ssl_amount=4.00&ssl_state=&ssl_card_number=00*****
*****0000&ssl_txn_time=01/03/2014 04:18:42 PM
```

This is an example of a similar transaction but authenticated (username here is USER) so when Converge reaches a destination it must provide the USER and the password in order to be allowed to post the data in green:

```
POST https://demo.myvirtualmerchant.com/VirtualMerchantDemo/trans.do
HTTP/1.1Content-Length:1060Host:demo.myvirtualmerchant.com:-lContent-
Type: application/x-www-form-urlencodeduser-agent:Apache-
HttpClient/4.2.2 (java 1.5)authorization:{BASIC =[principal: USER]}

ssl_email=&ssl_cvv2_response=M&Custom3=&ssl_ship_to_phone=&ssl_last_name
=&Custom1=&Custom2=&ssl_ship_to_country=&ssl_ship_to_state=&ssl_account_
balance=4.00&ssl_ship_to_zip=&ssl_company=&ssl_result_message=APPROVAL&s
sl_country=&ssl_city=&ssl_phone=&ssl_invoice_number=&ssl_ship_to_address
2=&ssl_ship_to_address1=&ssl_transaction_currency=USD&ssl_txn_id=AA4983B
-8464D602-E195-46B7-838C-
160E218876CF&ssl_result=0&ssl_ship_to_company=&ssl_avs_response=&ssl_tra
nsaction_type=SALE&ssl_approval_code=CVI463&ssl_ship_to_last_name=&ssl_a
vs_zip=&ssl_ship_to_city=&ssl_dynamic_dba=Sch.Med*&ssl_exp_date=1215&ssl
_ship_to_first_name=&ssl_avs_address=&ssl_salestax=&ssl_description=&ssl
_address2=&ssl_first_name=&ssl_amount=4.00&ssl_state=&ssl_card_number=00
*****0000&ssl_txn_time=01/03/2014 04:22:11 PM
```

Chapter 3: 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 Converge 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**
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**
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:
 - An authentic-looking email
 - 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 Converge login credentials.

- **Malware**

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, crimeware, most rootkits, and other malicious and unwanted software or program.

Common threats:

- **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 Practices and Compliance Checklist

Developers and merchant administrators may find the information presented here valuable when writing and configuring applications and websites that will interface with Converge. 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 Converge 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 Converge, we do not recommend this for your production website. All sensitive merchant data, including transaction amounts and your Converge 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 Converge.

- **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 User**

The Merchant Admin account has full rights and access to each terminal in your system. Each merchant account has one Merchant Admin user also called the MA user, which is identical to your Converge ID or VID. 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. To use the Merchant Admin (MA) user account to process transactions through gateway integration is not allowed, you must create a User ID specifically for this purpose. This allows more accurate tracking of how transactions occur and who is submitting them, as well as protecting you in the event of a security compromise by limiting what transaction types the User ID can process.

- **User requirements**

Due to updated security requirements, the Converge API requires that the User ID field be passed with a valid value for all transaction requests. It is best that the User ID used for the API is a separate user from the one used to login to the Converge application user interface. It is also important that you never use your API user to login to the application. When specifying a User ID in the transaction request, make sure that the PIN matches the User ID that you are passing and that those match the terminal on which you wish to process transactions. When an account has more than one Terminal, it is the combination of the account or merchant ID, PIN, and User ID that Converge uses to determine which terminal the transaction is processed under.

- **Password Security**

Do not set your password to be the same value (or a similar value) as any other data associated with your Converge account. This includes your Converge 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 Converge 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**

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

- **Fraud Prevention Rules**

A set of flexible customizable filters that help by minimizing and preventing suspicious and potentially costly fraudulent transactions, maximizing the acceptance of legitimate transactions. They also provide flexibility by allowing merchants to customize filter settings according to their unique business needs and improving intelligence by restricting transaction activity from specific Internet Protocol (IP) addresses. It is strongly recommended that you get familiar with the capabilities of each of these filters to determine which ones work best for your business needs and use them as fraud fighting tools. The available filters are as follows:

- Auto Pend
 - Merchant IP Address Filter
 - IP Address Filter
 - Individual/Ranges Filter
 - Country IP Address Filter
 - Country Filter
 - Billing Country Filter
 - Shipping Country Filter
 - IP Address & Country Mismatch Filter
 - IP Address & Billing Country Mismatch Filter
 - IP Address & Shipping Country Mismatch Filter
 - Email Address Filter
 - Card Number Filter
 - Email Domain Filter
 - Transaction Timeout Filter
- **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 Converge 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. Converge 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.

PCI 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. Converge is PCI certified and merchants using Converge 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 | 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 | Description |
|--|--|
| 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 | MasterCard is a technology company and payments industry leader. |
| Resource: https://www.eiseverywhere.com/ehome/index.php?eventid=8231&tabid=17737 | 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: 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 4: Transaction Format

The following chapter reviews the transaction processes and provide implementation guidelines and examples to process, format, and send transactions using `process.do`, `processxml.do`, `processBatch.do` and `accountxml.do`.

Topics include:

- Credit card transactions
- Recurring transactions
- Installment transactions
- Batch import transactions
- Debit card transactions
- EBT transactions
- Gift card transactions
- Electronic check transactions
- PINLess debit transactions
- Cash tender transactions
- Card Manager Transactions
- End-of-Day transactions
- Account Admin transactions

Important:

- All code samples provided in this document are examples and should not be used for live transactions.
- 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 Converge. 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.

- Converge allows you to set up custom defined fields and define them as required or optional. You need to consider the following when using any custom defined fields:
 - Only 25 fields can be set up for any given terminal, each field can be up to 999 alphanumeric characters long, and no special characters should be used.
 - You are not allowed to pass any sensitive data, including but not limited to PAN data such as full card number, expiration date, social security numbers, track data, or CID/CVV2 data from a credit card into a custom field.
 - The integrated application should not store or print the track data, CVV2, CVC2, or CID data from the back or front of credit cards
 - Use `process.do` for key value pairs formatted request or `processxml.do` for XML formatted request in the following transactions:
 - Credit card transactions
 - Recurring transactions
 - Installment transactions
 - Debit card transactions
 - EBT transactions
 - Gift card transactions
 - Electronic check transactions
 - PINLess debit transactions
 - Cash tender transactions
 - Card manager transactions
 - End of Day transactions
 - Account Admin transactions
 - Use `processBatch.do` for key value pairs formatted request in the following transactions:
 - Batch import transactions
 - Use `accountxml.do` for XML formatted request in the following transactions:
 - Admin 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.

Important Notes:

- **Credentials**

A unique API user different from the Merchant Admin (MA) user ID must be used.

- **Card data**

You must pass one of the following fields:

- Track data in the `ssl_track_data` for swiped transactions.
- The encrypted track data in the `ssl_encrypted_track1_data` and/or `ssl_encrypted__track2_data` fields extracted from an encrypting device. Refer to the [Encryption](#) section for more information.
- The card number in the `ssl_card_number` for hand keyed transactions.
- The token in the `ssl_token` from a previously tokenized card number.

- **AVS**

An integrated application should pass the Address Verification Service (AVS) data on hand key transactions 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 [AVS Response Codes](#) section for complete list of AVS response codes.

- **CVV**

An integrated application should pass the card verification value (CVV) on hand key transactions for fraud protection. The CVV field can be set as optional or required based on the merchant's business needs. When CVV data is passed, it is compared to the cardholder's CVV data that the card issuer has on file, a CVV Response Code is then returned. Refer to the [CVV2/CVC2 Response Codes](#) section for complete list of CVV response codes.

- If the CVV value is set to required:
 - A valid CVV value must be passed along with a **CVV** indicator of present (1) at the time of the authorization
- Or
- The Converge application will default the indicator to present (1) if no indicator is passed and a CVV value is present

- If the CVV value is set to optional:
 - A CVV value along with the appropriate **CVV** indicator may be passed to indicate if the CVV is present, bypassed, illegible or not present

Or

 - The Converge application will set the indicator to present (1) if the CVV value is passed or Bypassed (0) if the CVV value is not passed at the time of the authorization
- **Level 2 Data**

Commercial cards and purchasing cards require additional data to receive lower processing fees, it is recommended to pass Tax amount and customer code.
- **Tip Processing**

You can pass tips on *Service* market segment at the time of the authorization Cashier-based processing or after authorization Server-based processing. Optionally a server ID and shift ID can be passed as well.

On the initial transaction, the tip amount must be sent along with the transaction amount. The authorization amount is now the total amount, which is recalculated using the authorization amount originally sent and the new tip provided. The settlement amount will reflect the new authorized amount.

 - $\text{Authorization Amount} = \text{Amount} + \text{Tip Amount}$
 - $\text{Settlement Amount} = \text{Authorization Amount}$

After the initial transaction, the tip amount must be sent without the transaction amount. The authorization amount will not change. However, the total amount is recalculated using the original authorized amount and the new tip provided. The settlement amount will reflect the new total amount.

 - $\text{Settlement Amount} = \text{Authorization Amount} + \text{Tip Amount}$
 - The tip amount can be sent in the merchant or the cardholder currency
- **DBA**

The Dynamic Doing Business As (DBA) provides merchants the ability to control the descriptor on their customer's credit card statements. It allows the merchant to specify, on a per-transaction basis, wording that may be more recognizable or more service-specific to the customer than their usual business name preventing chargebacks. As an example, if the merchant sells magazine subscriptions for multiple publications, they may prefer to include the name of the publication in the authorization: MANYMAG*BAKERS MONTHLY Or MANYMAG*CAR DIGEST.

DBA is up to 25 characters constructed in the following format:

- Constant/prefix (3, 7, or 12 fixed alphanumeric value) configured in the terminal
- Asterisks (*) delimiter
- DBA name passed by the application The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17, or 12 based on the length setup for the constant.
- **Travel Data**
Converge supports the ability to capture and send travel information regarding departure and completion dates. The travel information dates are sent to Merchant Airline Risk Monitoring System (MARMS). MARMS is a tool used to monitor risk associated with merchants that collect or accept payments for future services or bookings made well in advance of the actual date of use. Both dates must be sent together and cannot be in past.

Credit Card 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.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Sale (<code>ccsale</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), otherwise set to false. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_track_data | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Number as it appears on the credit card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> <p>Note: Do not send an expiration date with a token that is stored in the Card Manager.</p> |
| ssl_amount | Y | <p>Transaction Sale Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax.</p> <ul style="list-style-type: none"> The authorized amount passed on request should not contain the tip amount, tips must be passed separately. The total authorized amount will be calculated during the authorization if tip is provided. For example: 1.00. For those terminals processing <i>Multi-Currency</i>, be sure to submit the correct number of decimal places for the transaction as some currencies have no exponents and some can have three. |

| Input Field Name | Req? | Description |
|------------------------|------|---|
| ssl_card_present | N | Recommended for hand-keyed transactions. 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 and service environments. |
| ssl_avs_zip | N | Recommended for hand-keyed transactions. Cardholder ZIP code. |
| ssl_avs_address | N | Recommended for hand-keyed transactions. Cardholder Address. |
| ssl_cvv2cvc2 | N | Recommended for hand-keyed transactions. The credit card security code is a three-digit or four-digit number, printed either on the back or the front of the card. When CVV data is passed, it is compared to the cardholder's CVV data that the card issuer has on file, a CVV Response Code is then returned. |
| ssl_cvv2cvc2_indicator | N | Recommended for hand-keyed transactions. The CVV2/CVC/CID indicator is one numeric value that should be passed with the CVV value (<code>ssl_cvv2cvc2</code>) to indicate if the CVV is present in the request. Valid values are: <ul style="list-style-type: none"> • 0 for Bypassed • 1 for Present • 2 for Illegible • 9 for Not Present |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_customer_code | N | Recommended for purchasing cards. The Customer Code or PO Number that appears on the cardholder's credit card billing statement. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |
| ssl_tip_amount | N | Use only in a <i>Service</i> market segment Tip or gratuity amount to be added, must be 2 decimal places, can be 0.00 to reset or remove the original tip from a transaction. Example: 1.00. |

| Input Field Name | Req? | Description |
|----------------------------|------|--|
| ssl_server | N | <p>Use only in a <i>Service</i> market segment</p> <p>Server ID, this is the clerk, cashier, and waiter or waitress identification number.</p> <p>Alphanumeric, Example: Jack.</p> |
| ssl_shift | N | <p>Use only in a <i>Service</i> market segment.</p> <p>Shift, can refer to or be used to identify time period, course or type of service.</p> <p>Alphanumeric, Example: Lunch.</p> |
| ssl_dynamic_dba | N | <p>User only with a terminal that is setup with <i>DBA</i>.</p> <p>DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup.</p> |
| ssl_partial_auth_indicator | N | <p>The partial indicator flag must be sent to indicate that the application supports partial approval.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 0 – Partial Auth not supported • 1 – Partial Auth supported |
| ssl_recurring_flag | N | <p>Use only when maintaining your own <i>recurring</i> and <i>installment</i> database.</p> <p>The recurring flag must be sent to indicate if a credit card sale transaction is a recurring or an installment payment.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 1=Recurring • 2=Installment <p>Note: When the flag is indicated as Installment, the payment number and payment count must be passed.</p> |
| ssl_payment_number | N | <p>Use only when maintaining your own <i>recurring</i> and <i>installment</i> database.</p> <p>Installment Sequence Number (Payment Number).</p> |
| ssl_payment_count | N | <p>Use only when maintaining your own <i>recurring</i> and <i>installment</i> database.</p> <p>Installment Count (total number of payments).</p> |

| Input Field Name | Req? | Description |
|--------------------------|------|--|
| ssl_departure_Date | N | <p>Use only with a terminal that is setup with <i>Travel Data</i>.</p> <p>Date of the departure. Format MM/DD/YYYY. This field will be sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with option for MARMS.</p> |
| ssl_completion_Date | N | <p>Use only with a terminal that is setup with <i>Travel Data</i>.</p> <p>Date of the completion of travel. Format MM/DD/YYYY. This field is sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with Travel Data option for MARMS.</p> |
| ssl_transaction_currency | N | <p>Use only with a terminal that is setup with <i>Multi-Currency</i>.</p> <p>Transaction currency alphanumeric code must be included in the request to indicate the currency in which you wish to process. If omitted, the terminal default currency is assumed (for example: USD, CAD, JPY). More than 94 currencies are supported. Refer to the ISO Currency Codes section for an extensive list of available currencies.</p> |
| ssl_get_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid value: Y (generate a token), N (do not generate token). Defaulted to N.</p> <p>Once generated, the token number can be stored and used as a substitute for a card number at later time.</p> |
| ssl_add_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Add to Card Manager indicator, used to indicate if you wish to generate a token and store it in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N</p> <p>To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent <i>alone</i> and will be used as a substitute for the stored information.</p> |

Response

| Output Field Name | Description |
|-------------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | 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 | 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_approval_code | Transaction approval code. |
| ssl_amount | The total transaction authorized or approved amount. This amount will include tip if tip has been provided in the request. Returned based on merchant setup. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. Returned in a <i>Service</i> market segment. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. Returned in a <i>Service</i> market segment. |
| ssl_requested_amount | The amount originally requested on partial approvals only. |
| ssl_balance_due | Remaining balance due in the <code>ssl_balance_due</code> 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_account_balance | The balance left in card, which is always 0.00 for a partially authorized transaction. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_invoice_number | The invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| ssl_cardholder_currency | Only returned on DCC transactions. |

| Output Field Name | Description |
|----------------------------|---|
| ssl_cardholder_amount | Total amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_base_amount | Base amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_tip_amount | Tip amount in cardholder currency, only returned on DCC transactions with <i>Service</i> market segment. |
| ssl_server | Server ID submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_shift | Shift information submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_eci_ind | Only returned on 3D Secure transactions. Valid values: <ul style="list-style-type: none"> Fully Authenticated Authentication Attempted |
| ssl_transaction_currency | Transaction currency. Returned only if terminal is setup for <i>Multi-Currency</i> . |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that is setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, or Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, or FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_promo_list | The promo list will contain a list of all promo products; the data for each promo product will be nested and embedded between beginning and ending elements <ssl_promo_product> up to 5 promo products. Each promo product ssl_promo_product will contain ssl_promo_code, ssl_promo_code_name, ssl_promo_code_description, and ssl_promo_code_issue_points. |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Examples

Important:

- In all of these examples, you will have to change the data values, such as `my_virtualmerchant_id`, `my_user_id`, `my_pin`, and transaction data to match your Converge account and meet the needs of your website.
- Code samples provided are for demonstration only and should not be used for live transactions. All sensitive merchant data, including transaction amounts and your Converge credentials, should be placed in server side code.

Example 1: `process.do (true)`

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

```
ssl_merchant_id=my_vid_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=true
ssl_amount=1.00
ssl_result_format=HTML
ssl_transaction_type=ccsale
```

Example 2: process.do (false)

The following example demonstrates the key value pairs from the header by themselves for a credit card sale transaction where the merchant collects all the data

```
ssl_merchant_id=my_vid_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_card_number=0000000000000000
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
```

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=00*****0000
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
```

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
```

Example 3: process.do (false)with AVS and CVV2/CVC2

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

```
ssl_merchant_id=my_vid_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_transaction_type=ccsale
ssl_amount=12.77
ssl_card_number=0000000000000000
ssl_exp_date=1208
ssl_cvv2cvc2_indicator=1
ssl_cvv2cvc2=123
ssl_avs_address=123 Main Street
ssl_avs_zip=99999
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-
```

Example 4: process.do with Receipt

The following example 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:

```
ssl_merchant_id=my_vid_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_transaction_type=ccsale
ssl_amount=5.00
ssl_card_number=0000000000000000
ssl_exp_date=1208
ssl_cvv2cvc2_indicator=1
ssl_cvv2cvc2=123
ssl_avs_address=123 Main Street
ssl_avs_zip=99999
ssl_invoice_number=123-ABC
ssl_email=test@test.com
ssl_result_format=HTML
ssl_receipt_decl_method=POST
ssl_receipt_decl_post_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=GET
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
ssl_receipt_link_text=Continue
```

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="M">
<input type="hidden" name="ssl_avs_response" value="A">
<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>
```

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: process.do (form) with ASCII

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

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_transaction_type=ccsale
ssl_amount=5.00
ssl_invoice_number=123-ABC
ssl_email=test@test.com
ssl_card_number= 0000000000000000
ssl_exp_date=1215
ssl_result_format=ASCII
```

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=M
ssl_avs_response=A
ssl_invoice_number=123-ABC
ssl_amount=5.00
ssl_card_number=00*****0000
ssl_exp_date=0000
ssl_email=test@test.com
```

A declined transaction will generate a receipt that includes the following key value pairs:

```
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: process.do (false) Redirect

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

```
ssl_amount=5.00
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccsale
ssl_show_form=false
ssl_invoice_number=123-ABC
ssl_email=test@test.com
ssl_card_number=0000000000000000
ssl_exp_date=1214
ssl_result_format=HTML
ssl_receipt_decl_method="REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

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. Transaction data will be passed along as GET variables in the query string of the URL.

Example 7: process.do (false) Partially Approved Transaction

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

```
ssl_amount=10.10
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccsale
ssl_partial_auth_indicator=1
ssl_show_form=true
ssl_invoice_number" value="123-ABC
ssl_email=test@test.com
ssl_card_number=0000000000000000
ssl_exp_date=1214
ssl_result_format=HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

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>
```

Example 8: processxml.do

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

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

```
xmldata=<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>00*****0000
</ssl_card_number><ssl_exp_date>1215</ssl_exp_date><ssl_amount>1.00
</ssl_amount>
</txn>
```

The Response generated will look similar to the following:

```
xmldata=<txn>
<ssl_card_number>00*****0000</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>
```

Example 9: processxml.do Partial Approved Transaction

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.

The Initial Request will show the following:

```

xmldata=<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>0000000000000000</ssl_card_number>
  <ssl_exp_date>1212</ssl_exp_date>
  <ssl_amount>10.10</ssl_amount>
  <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
  <ssl_first_name>Test</ssl_first_name>
  <ssl_partial_auth_indicator>1</ssl_partial_auth_indicator>
</txn>

```

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.

```

xmldata=<txn>
  <ssl_card_number>00*****0000</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_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.

```
xmldata=<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>
```

Example 10: processxml.do Sale with Tip Transaction

The following XML code example demonstrates the initiation of a Sale transaction with tip in the *Service* market segment.

Notes:

Provide the following information in order to add a tip or gratuity amount at the time of the transaction:

- The amount to be authorized in the `ssl_amount` field.
 - The tip amount you wish to add to the amount to be authorized in the `ssl_tip_amount` field.
 - The Server ID in the `ssl_server` field.
 - The Shift in the `ssl_shift` field.
-

In this example the Sale amount is 10.00, the tip amount is 2.00. Please do not provide the total amount in the request.

```
xmldata=<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>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>10.00</ssl_amount>
  <ssl_tip_amount>2.00</ssl_tip_amount>
  <ssl_shift>dinner</ssl_shift>
  <ssl_server>Joel23</ssl_server>
</txn>
```

The Sale amount is then added to the tip amount and sent for authorization, an approval will be returned to the integrated application. The following information is returned in the response:

- The total amount that has been authorized in the `ssl_amount` field.
- The original Sale amount in the `ssl_base_amount` field.
- The tip amount provided in the in the `ssl_tip_amount` field.
- The authorization data (response message, approval code).

```
xmldata=<txn>
  <ssl_result>0</ssl_result>
  <ssl_approval_code>N25032</ssl_approval_code>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_txn_id>TEST43B-D0638677-26EB-40F5-B2F9-3AF2545DE144</ssl_txn_id>
  <ssl_txn_time>10/03/2012 10:25:03 PM</ssl_txn_time>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_base_amount>10.00</ssl_base_amount>
  <ssl_amount>12.00</ssl_amount>
  <ssl_tip_amount>2.00</ssl_tip_amount>
  <ssl_shift>dinner</ssl_shift>
  <ssl_server>Joel</ssl_server>
</txn>
```

Example 11: processxml.do Sale with currency

The following XML code example demonstrates the initiation of a Sale transaction with currency:

```
xmldata=<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>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>200</ssl_amount>
  <ssl_transaction_currency>JPY</ssl_transaction_currency>
  <ssl_invoice_number>XMLXSSJPYNVISA20-200</ssl_invoice_number>
```

Notes:

- When specifying amounts, be sure to submit the correct number of decimal places for the transaction currency.
- For Japanese Yen, there are no currency exponents after the decimal place. Any numbers included after the decimal place will be dropped.
- For those currencies with 3 possible digits, for example: Bahraini Dinar, we will automatically round the transaction to two decimals. (for example: 1.235 will round to 1.23).

Converge will process and settle this transaction in JPY. The following information is returned in the response:

```
<txn>
<ssl_approval_code>CVI032</ssl_approval_code>
<ssl_cvv2_response />
<ssl_exp_date>1222</ssl_exp_date>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_invoice_number>XMLXSSJYPYNVISA20-200</ssl_invoice_number>
<ssl_amount>200</ssl_amount>
<ssl_transaction_currency>JPY</ssl_transaction_currency>
<ssl_result>0</ssl_result>
<ssl_txn_id>AA47AE-BD32FF54-BE12-4046-98E9-
F78E9D75212D</ssl_txn_id>
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_txn_time>11/19/2013 12:02:29 AM</ssl_txn_time>
```

Credit Card Auth Only (ccauthonly)

The `ccauthonly` is used to obtain a real-time authorization for a credit card transaction. This transaction will guarantee that the funds are available on the card and reduce the cardholder's limit to buy for only a predetermined amount of time. This is usually 7-10 days based on the credit card's issuing bank, however it will not place the authorization in the batch for settlement.

To place the transaction in the open batch, it must be converted to Sale using `cccomplete`, or reversed using `ccdelete` to restore the funds back to the card.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Auth Only (<code>ccauthonly</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_track_data | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | C | <p>Required for hand-keyed transactions.</p> <p>Credit card number as it appears on the credit card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> <p>Note: Do not send an expiration date with a token that is stored in the Card Manager.</p> |
| ssl_amount | Y | <p>Transaction Auth Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax.</p> <p>For those terminals processing <i>Multi-Currency</i>, be sure to submit the correct number of decimal places for the transaction as some currencies have no exponents and some can have three.</p> |
| ssl_card_present | N | <p>Recommended for hand-keyed transactions.</p> <p>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 and service environments.</p> |
| ssl_avs_zip | N | <p>Recommended for hand-keyed transactions.</p> <p>Cardholder ZIP code.</p> |

| Input Field Name | Req? | Description |
|----------------------------|------|---|
| ssl_cvv2cvs2 | N | Recommended for hand-keyed transactions. Cardholder Address. |
| ssl_cvv2cvs2_indicator | N | Recommended for hand-keyed transactions. The CVV2/CVC/CID indicator is one numeric value that should be passed with the CVV value (<i>ssl_cvv2cvc2</i>) to indicate if the CVV is present in the request. Valid values are: <ul style="list-style-type: none"> • 0 for Bypassed • 1 for Present • 2 for Illegible • 9 for Not Present |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_customer_code | N | Recommended for purchasing cards. The Customer Code or PO Number that appears on the cardholder's credit card billing statement. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |
| ssl_dynamic_dba | N | Use only with a terminal that is setup with <i>DBA</i> . DBA name provided by the merchant with each transaction. The maximum allowable length of DBA Name variable provided by the merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup. |
| ssl_partial_auth_indicator | N | The partial indicator flag must be sent to indicate that the application supports partial approval. Valid value: 1. |
| ssl_departure_date | N | User only with a terminal that is setup with <i>Travel Data</i> . Date of the departure. Format MM/DD/YYYY. This field will be sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with option for MARMS. |
| ssl_completion_date | N | Used only with a terminal that is setup with <i>Travel Data</i> . Date of the completion of travel. Format MM/DD/YYYY. Optional. This field is sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with <i>Travel Data</i> option for MARMS. |

| Input Field Name | Req? | Description |
|--------------------------|------|---|
| ssl_transaction_currency | N | Use only with a terminal that is setup with <i>Multi-Currency</i> . Transaction currency alphanumeric code must be included in the request to indicate the currency in which you wish to process. If omitted the terminal default currency is assumed (for example: USD, CAD, JPY). More than 94 currencies are supported. Refer to the ISO Currency Codes section for an extensive list of available currencies. |
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |
| ssl_add_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to generate a token and store it in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> 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 | 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 | Transaction identification number. This is a unique number used to identify the transaction. This value can be used to complete the Auth Only 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. |

| Output Field Name | Description |
|--------------------------|--|
| ssl_approval_code | Transaction approval code. |
| ssl_amount | Transaction authorized or approved amount. Returned based on merchant setup. |
| ssl_requested_amount | The amount originally requested on partial approvals only. |
| ssl_balance_due | Remaining balance due in the <code>ssl_balance_due</code> 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_account_balance | The balance left on the card, which is always 0.00 for a partially authorized transaction. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_invoice_number | The invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| ssl_eci_ind | Only returned on 3D Secure transactions. Valid values: <ul style="list-style-type: none"> Fully Authenticated Authentication Attempted |
| ssl_transaction_currency | Transaction currency. Returned only if terminal is setup for <i>Multi-Currency</i> . |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that is setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, or Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, or FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Examples

Important:

- In all of these examples, you will have to change the data values, such as `my_virtualmerchant_id`, `my_user_id`, `my_pin`, and transaction data to match your Converge account and meet the needs of your website.
 - Code samples provided are for demonstration only and should not be used for live transactions. All sensitive merchant data, including transaction amounts and your Converge credentials, should be placed in server side code, rather than in hidden value fields on an HTML form.
-

Example 1: process.do (false)

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

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_card_number=0000000000000000
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=cacauthonly
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
```

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=00*****0000
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
```

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
```

Example 2: `process.do(true)`

This example demonstrates the initiation of a minimal Auth Only transaction in which Converge payment form gathers the entire customer's billing information.

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccsale
ssl_show_form=true
ssl_amount=5.00
```

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 5.00 to values that match your Converge account and meet the needs of your websites.

Example 3: processxml.do

The following XML code example demonstrates the initiation of a basic Auth Only transaction. In the case of XML, no additional information can be collected by Converge when the request is sent. All required data must be sent in the transaction request to process the transaction:

```
<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_description>Auth for 3.00</ssl_description>
  <ssl_transaction_type>CCAUTHONLY</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>3.00</ssl_amount>
  <ssl_salestax>0.01</ssl_salestax>
  <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
  <ssl_cvv2cvc2>123</ssl_cvv2cvc2>
  <ssl_customer_code>CORP</ssl_customer_code>
  <ssl_invoice_number>1234</ssl_invoice_number>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
  <ssl_phone>9999999999</ssl_phone>
  <ssl_ship_to_state>GA</ssl_ship_to_state>
  <ssl_ship_to_zip>99999</ssl_ship_to_zip>
  <ssl_ship_to_address1>999 Main</ssl_ship_to_address1>
  <ssl_ship_to_company>Ship Company</ssl_ship_to_company>
  <ssl_ship_to_last_name>Doe</ssl_ship_to_last_name>
  <ssl_ship_to_city>Any City</ssl_ship_to_city>
  <ssl_ship_to_first_name>Jane</ssl_ship_to_first_name>
</txn>
```

(continued)

```

<txn>
  <ssl_approval_code>CMC720</ssl_approval_code>
  <ssl_cvv2_response>M</ssl_cvv2_response>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_account_balance>3.00</ssl_account_balance>
  <ssl_company />
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_country>USA</ssl_country>
  <ssl_city>Atlanta</ssl_city>
  <ssl_phone>9999999999</ssl_phone>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_invoice_number>1234</ssl_invoice_number>
  <ssl_address2 />
  <ssl_first_name>John</ssl_first_name>
  <ssl_amount>3.00</ssl_amount>
  <ssl_state>GA</ssl_state>
  <ssl_txn_id>AA4843A-1416AF5F-9A4C-4C10-AA00-
537ADF2B3E0E</ssl_txn_id>
  <ssl_result>0</ssl_result>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_txn_time>01/28/2014 05:14:49 PM</ssl_txn_time>
  <ssl_avs_response>Z</ssl_avs_response>
</txn>

```

(end)

Credit Card 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.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Credit Card AVS Only (ccavsonly). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| ssl_track_data | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | C | <p>Required for hand-keyed transactions.</p> <p>Credit card number as it appears on the credit card.</p> |
| ssl_exp_date | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> |
| ssl_avs_address | Y | Customer's address used to process AVS. |
| ssl_avs_zip | Y | Customer's ZIP code used to process AVS. |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Examples

Important:

- In all of these examples, you will have to change the data values, such as `my_virtualmerchant_id`, `my_user_id`, `my_pin`, and transaction data to match your Converge account and meet the needs of your website.
- Code samples provided are for demonstration only and should not be used for live transactions. All sensitive merchant data, including transaction amounts and your Converge credentials, should be placed in server side code, rather than in hidden value fields on an HTML form.

Example 1: process.do (false)

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

```
ssl_merchant_id=xxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxx
ssl_show_form=false
ssl_transaction_type=ccavonly
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_avs_zip=99999
ssl_avs_address=123 Main
ssl_result_format= ASCII
```

The following values are returned for the approved transaction indicating that the Address matches, but ZIP Code does not match

```
ssl_approval_code=CMC020
ssl_avs_address=123 Main
ssl_avs_zip=99999
ssl_txn_id=AA4843A-444ECA99-EB04-4A8D-8339-B2A578F92EF6
ssl_result=0
ssl_card_number=00*****0000
ssl_account_balance=0.00
ssl_txn_time=01/28/2014 05:25:48 PM
ssl_result_message=APPROVAL
ssl_avs_response=A
```

Example 2: `process.do(true)`

This example demonstrates the initiation of a minimal AVS Only transaction in which Converge payment form gathers the entire card data.

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccavsonly
ssl_show_form=true
```

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 5.00 to values that match your Converge account and meet the needs of your websites.

Example 3: processxml.do

The following XML code example demonstrates the initiation of a basic AVS Only transaction. In the case of XML, no additional information can be collected by Converge when the request is sent. All required data must be sent in the transaction request to process the transaction:

```
<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_transaction_type>CCAVSONLY</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
</txn>
<txn>
  <ssl_approval_code>CMC720</ssl_approval_code>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_account_balance>3.00</ssl_account_balance>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_country>USA</ssl_country>
  <ssl_city>Atlanta</ssl_city>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_first_name>John</ssl_first_name>
  <ssl_state>GA</ssl_state>
  <ssl_txn_id>AA3456VC-1416AF5F-9A4C-4C10-AA00-
537ADF2B3E0E</ssl_txn_id>
  <ssl_result>0</ssl_result>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_txn_time>01/28/2014 05:33:40 PM</ssl_txn_time>
  <ssl_avs_response>Z</ssl_avs_response>
</txn>
```

Credit Card Verification (ccverify)

The `ccverify` transaction is used to verify the credit card account for AVS and CVV data. AVS and CVV codes are returned to indicate if the AVS and CVV data passed originally were correct and matched the cardholder statement billing address and the CVV value located on the card.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Verification (<code>ccverify</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| <code>ssl_track_data</code> | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| <code>ssl_card_number</code> | C | <p>Required for hand-keyed transactions.</p> <p>Credit card number as it appears on the credit card.</p> |
| <code>ssl_exp_date</code> | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> |
| <code>ssl_avs_zip</code> | Y | <p>Recommended for hand-keyed transactions.</p> <p>Cardholder ZIP code.</p> |
| <code>ssl_avs_address</code> | Y | <p>Recommended for hand-keyed transactions.</p> <p>Cardholder Address.</p> |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_cvv2cvc2 | Y | <p>Recommended for hand-keyed transactions.</p> <p>The credit card security code is a three-digit or four-digit number, printed either on the back or the front of the card.</p> <p>When CVV data is passed, it is compared to the cardholder's CVV data that the card issuer has on file, a CVV Response Code is then returned.</p> |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example**Important:**

- In all of these examples, you will have to change the data values, such as `my_virtualmerchant_id`, `my_user_id`, `my_pin`, and transaction data to match your Converge account and meet the needs of your website.
- Code samples provided are for demonstration only and should not be used for live transactions. All sensitive merchant data, including transaction amounts and your Converge credentials, should be placed in server side code, rather than in hidden value fields on an HTML form.

Example 1: process.do (false)

Shown below are the key value pairs from the header by themselves for a verification transaction:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_transaction_type=ccverify
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_cvv2cvc2=123
ssl_avs_zip=99999
ssl_avs_address=123 Main
ssl_result format= ASCII
```

The following values are returned for the approved transaction indicating that the Address and CVV value match – but ZIP Code does not match

```
ssl_approval_code=CMC020
ssl_avs_address=123 Main
ssl_avs_zip=99999
ssl_txn_id=AA4843A-B73B4A08-FA09-43E1-B858-072923E2E387
ssl_result=0
ssl_card_number=00*****0000
ssl_account_balance=0.00
ssl_txn_time=01/28/2014 05:25:48 PM
ssl_result_message=APPROVAL
ssl_avs_response=A
ssl_cvv2_response=M
```

Example 2: process.do(true)

This example demonstrates the initiation of a minimal verification in which Converge payment form gathers the entire card data.

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccverify
ssl_show_form=true
```

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 5.00 to values that match your Converge account and meet the needs of your websites.

Example 3: processxml.do

The following XML code example demonstrates the initiation of a basic verification. In the case of XML, no additional information can be collected by Converge when the request is sent. All required data must be sent in the transaction request to process the transaction:

```
<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_transaction_type>CCVERIFY</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_cvv2cvc2>123</ssl_cvv2cvc2>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
</txn>
```

(continued)

```

<txn>
  <ssl_approval_code>CMC720</ssl_approval_code>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_zip>99999</ssl_avs_zip>
  <ssl_account_balance>0.00</ssl_account_balance>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_country>USA</ssl_country>
  <ssl_city>Atlanta</ssl_city>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_first_name>John</ssl_first_name>
  <ssl_state>GA</ssl_state>
  <ssl_txn_id> AA4843A-750B6205-7F8F-42B4-838A-
  79E0BA322D56</ssl_txn_id>
  <ssl_result>0</ssl_result>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_txn_time>01/28/2014 05:36:40 PM</ssl_txn_time>
  <ssl_avs_response>Z</ssl_avs_response>
  <ssl_cvv2_response>M</ssl_cvv2_response>
</txn>
    
```

(end)

Credit Card Return/Credit (cccredit)

The cccredit transaction is used to issue a return (refund) to a cardholder’s credit card using full card number.

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Credit Card Return/Credit (cccredit). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (available only for process.do), set to false otherwise. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_track_data | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Number as it appears on the credit card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> <p>Note: Do not send an expiration date with a token that is stored in Card Manager.</p> |
| ssl_amount | Y | <p>Transaction Amount to be refunded. Positive Number with 2 decimal places. Example: 2.00</p> <p>For those terminals processing <i>Multi-Currency</i>, be sure to submit the correct number of decimal places for the transaction as some currencies have no exponents and some can have three.</p> |
| ssl_card_present | N | <p>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 and service environments.</p> |
| ssl_dynamic_dba | N | <p>Use only with a terminal that is setup with <i>DBA</i>.</p> <p>DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup.</p> |

| Input Field Name | Req? | Description |
|--------------------------|------|---|
| ssl_departure_Date | N | <p>User only with a terminal that is setup with <i>Travel Data</i>.</p> <p>Date of the departure. Format MM/DD/YYYY. This field will be sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with option for MARMS.</p> |
| ssl_completion_Date | N | <p>Used only with a terminal that is setup with <i>Travel Data</i>.</p> <p>Date of the completion of travel. Format MM/DD/YYYY. Optional. This field is sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with <i>Travel Data</i> option for MARMS.</p> |
| ssl_transaction_currency | N | <p>Use only with a terminal that is setup with <i>Multi-Currency</i>.</p> <p>Transaction currency alphanumeric code must be included in the request to indicate the currency in which you wish to process. If omitted the terminal default currency is assumed (for example: USD, CAD, JPY). More than 94 currencies are supported. Refer to the ISO Currency Codes section for an extensive list of available currencies.</p> |
| ssl_get_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid values: Y (generate a token), N (do not generate token).</p> <p>Defaulted to N</p> <p>Once generated, the token number can be stored and used as a substitute for a card number at later time.</p> |
| ssl_add_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Add to Card Manager indicator, used to indicate if you wish to store the token generated in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N</p> <p>To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information.</p> |

Response

| Output Field Name | Description |
|--------------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_amount | Transaction amount. Returned based on merchant setup. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_cardholder_amount | Only returned on DCC transactions. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| 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_transaction_currency | Transaction currency. Returned only if terminal is setup for <i>Multi-Currency</i> |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that is setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, or Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, or FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

Example: process.do(false)

The following example demonstrates the key value pairs from the header by themselves for a credit card return transaction for \$10.00 where the merchant collects all the data from the consumer:

```
ssl_merchant_id=my_vid_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_amount=10.00
ssl_invoice_number=1234
ssl_error_url=http://www.url.com/cgi-bin/testtran.cgi
ssl_result_format=HTML
ssl_transaction_type=cccredit
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
```

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=00*****0000
ssl_amount=10.00
ssl_result=0
ssl_result_message=APPROVAL
ssl_txn_id= AA4843A-335FBE74-9608-45A1-B30A-658A3ABB5584
ssl_account_balance=1.00
ssl_invoice_number=1234
ssl_txn_time= 01/28/2014 05:46:22 PM
```

Credit Card Force (ccforce)

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

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Credit Card Force (<code>ccforce</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| <code>ssl_track_data</code> | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information.</p> |

| Input Field Name | Req? | Description |
|-------------------|------|---|
| ssl_card_number | C | Required for hand-keyed transactions. Credit Card Number as it appears on the credit card. |
| ssl_token | C | Required for hand-keyed transaction if card number is not sent. Use only with a terminal that is setup with <i>Tokenization</i> . Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number. |
| ssl_exp_date | C | Required for hand-keyed transactions. Credit Card Expiry Date as it appears on credit card formatted as MMY. Note: Do not send an expiration date with a token that is stored in Card Manager. |
| ssl_approval_code | Y | Previously received Authorization Approval Code. |
| ssl_amount | Y | Transaction Sale Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. <ul style="list-style-type: none"> The authorized amount passed on request should not contain the tip amount, tips must be passed separately. The total authorized amount will be calculated during the authorization if tip is provided. For example: 1.00. For those terminals processing <i>Multi-Currency</i>, be sure to submit the correct number of decimal places for the transaction as some currencies have no exponents and some can have three. |
| ssl_card_present | N | Recommended for hand-keyed transactions. 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 and service environments. |
| ssl_avs_zip | N | Recommended for hand-keyed transactions. Cardholder ZIP code. |
| ssl_avs_address | N | Recommended for hand-keyed transactions. Cardholder Address. |
| ssl_cvv2cvc2 | N | Recommended for hand-keyed transactions. The credit card security code is a three-digit or four-digit number, printed either on the back or the front of the card. When CVV data is passed, it is compared to the cardholder's CVV data that the card issuer has on file, a CVV Response Code is then returned. |

| Input Field Name | Req? | Description |
|------------------------|------|--|
| ssl_cvv2cvc2_indicator | N | Recommended for hand-keyed transaction. The CVV2/CVC/CID indicator is one numeric value that should be passed with the CVV value (<code>ssl_cvv2cvc2</code>) to indicate if the CVV is present in the request. Valid values are: <ul style="list-style-type: none"> • 0 for Bypassed • 1 for Present • 2 for Illegible • 9 for Not Present |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_customer_code | N | Recommended for purchasing cards. The Customer Code or PO Number that appears on the cardholder's credit card billing statement. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |
| ssl_tip_amount | N | Use only in a <i>Service</i> market segment Tip or gratuity amount to be added, must be 2 decimal places, can be 0.00 to reset or remove the original tip from a transaction. Example: 1.00. |
| ssl_server | N | Use only in a <i>Service</i> market segment Server ID, this is the clerk, cashier, and waiter or waitress identification number. Alphanumeric, Example: Jack. |
| ssl_shift | N | Use only in a <i>Service</i> market segment. Shift, can refer to or be used to identify time period, course or type of service. Alphanumeric, Example: Lunch. |
| ssl_dynamic_dba | N | Use only with a terminal that is setup with <i>DBA</i> . DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup. |
| ssl_departure_Date | N | User only with a terminal that is setup with <i>Travel Data</i> . Date of the departure. Format MM/DD/YYYY. This field will be sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with option for MARMS. |

| Input Field Name | Req? | Description |
|--------------------------|------|--|
| ssl_completion_Date | N | Used only with a terminal that is setup with <i>Travel Data</i> . Date of the completion of travel. Format MM/DD/YYYY. Optional. This field is sent to the MARMS system to monitor risk associated with advanced booking. Examples include airlines and travel or tour agencies. Only used when setup with Travel Data option for MARMS. |
| ssl_transaction_currency | N | Use only with a terminal that is setup with <i>Multi-Currency</i> . Transaction currency alphanumeric code must be included in the request to indicate the currency in which you wish to process. If omitted the terminal default currency is assumed (for example: USD, CAD, JPY). More than 94 currencies are supported. Refer to the ISO Currency Codes section for an extensive list of available currencies. |
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |
| ssl_add_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to generate a token and store it in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> 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 | Transaction result message. Example: APPROVAL. Refer to Authorization Response Codes section for more information. |

| Output Field Name | Description |
|----------------------------|--|
| ssl_txn_id | 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_approval_code | Transaction approval code. |
| ssl_amount | The total transaction authorized or approved amount. This amount will include tip if tip has been provided in the request. Returned based on merchant setup. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. Returned in the <i>Service</i> market segment. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. Returned in the <i>Service</i> market segment. |
| ssl_requested_amount | The amount originally requested on partial approvals only. |
| ssl_balance_due | Remaining balance due in the <code>ssl_balance_due</code> 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_account_balance | The balance left in card, which is always 0.00 for a partially authorized transaction. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_invoice_number | The invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| ssl_cardholder_currency | Only returned on DCC transactions. |
| ssl_cardholder_amount | Total amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_base_amount | Base amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_tip_amount | Tip amount in cardholder currency, only returned on DCC transactions with <i>Service</i> market segment. |

| Output Field Name | Description |
|--------------------------|--|
| ssl_server | Server Id submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_shift | Shift information submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_eci_ind | Only returned on 3D Secure transactions. Valid values: <ul style="list-style-type: none"> Fully Authenticated Authentication Attempted |
| ssl_transaction_currency | Transaction currency. Returned only if terminal is setup for <i>Multi-Currency</i> |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that is setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, or Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, or FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card 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 hand keyed pre-paid card.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Balance Inquiry (<code>ccbalinquiry</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| <code>ssl_track_data</code> | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| <code>ssl_card_number</code> | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Number as it appears on the credit card.</p> |
| <code>ssl_token</code> | C | <p>Required for hand-keyed transaction if card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number.</p> |
| <code>ssl_exp_date</code> | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Expiry Date as it appears on credit card formatted as MMY.</p> <p>Note: Do not send an expiration date with a token that is stored in Card Manager.</p> |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |
| ssl_add_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to store the token generated in Card Manager. Valid value: Y (add token), N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |

Response

| Output Field Name | Description |
|--------------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | The account balance. Number with two decimal places. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_transaction_currency | Transaction currency. Returned only if terminal is setup for <i>Multi-Currency</i> . |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that is setup for <i>Tokenization</i> . |

| Output Field Name | Description |
|------------------------|--|
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, or Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, or FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card Generate Token (ccgettoken)

The `ccgettoken` is a transaction that generates a token from a card number or an exiting recurring/installment in the recurring batch. The token generated can be used in place of a credit card number in any subsequent transactions. Additionally, you can request that the token generated is added to the Card Manager. This transaction type is supported only when a terminal is setup for tokenization; refer to the [Tokenization](#) section for more information.

To perform a `ccgettoken`, you must submit either:

- Card number, expiration date for generating a token, in addition first and last name are required to add the token to card manager.
- Or
- Track data in the `ssl_track_data` for swiped transactions.
- Or
- Only the Recurring ID or installment ID of an existing recurring/installment transaction in the `ssl_recurring_id` field for generating a token. In order to add the token to the card manager, the first and last names must be previously stored with the recurring records or sent along with the transaction.

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Credit Card Generate Token (ccgettoken). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| ssl_track_data | C | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | C | <p>Required for hand-keyed transactions.</p> <p>Credit Card Number as it appears on the credit card.</p> |
| ssl_exp_date | C | <p>Required for hand-keyed transactions.</p> <p>Expiration date as it appears on the credit card.</p> |
| ssl_recurring_id | C | <p>Required for recurring or installment transactions.</p> <p>The ID number of the recurring record or installment record to be submitted for payment. This value was returned in the recurring ID when the original recurring record was added for recurring or populate with installment ID when the original installment was added for installment. Alphanumeric.</p> |
| ssl_verify | N | Account Verify indicator to indicate that account verification is needed prior to generating a token. Valid values: Y, N. |
| ssl_avs_zip | C | <p>Recommended with Account Verify indicator, required with Add to Card Manager indicator.</p> <p>Cardholder ZIP code.</p> |
| ssl_avs_address | C | <p>Recommended with Account Verify indicator, required with Add to Card Manager indicator.</p> <p>Cardholder Address.</p> |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_cvv2cvc2 | N | Recommended with Account Verify indicator. The credit card security code is a three-digit or four-digit number, printed either on the back or the front of the card. When CVV data is passed, it is compared to the cardholder's CVV data that the card issuer has on file, a CVV Response Code is then returned. |
| ssl_add_token | N | Add to Card Manager indicator, used to indicate if you wish to store the token generated in Card Manager. Valid value: Y (add token), N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required or you can send a recurring ID that has those data previously stored. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |
| ssl_first_name | C | Required with Add to Card Manager indicator when generating a token from card number. Cardholder first name. |
| ssl_last_name | C | Required with Add to Card Manager indicator when generating a token from card number. Cardholder last name. |

Response

| Output Field Name | Description |
|------------------------|---|
| ssl_result | Generate Token Result code. A result of 0 indicates that the token was generated; 1 indicates that the token was not generated. |
| ssl_result_message | The account verification transaction result message returned only when ccgettoken is requested with Account verification flag. Example: APPROVAL, DECLINE. Refer to the Authorization Response Codes section for an extensive list of possible returned messages. |
| ssl_token | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. |
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, FAILURE - First Name - is required. Returned only if storing token is requested in a terminal that setup for <i>Tokenization</i> . |

| Output Field Name | Description |
|-------------------|---|
| ssl_txn_id | Transaction identification number, returned only when <code>ccgettoken</code> is requested with Account verification flag. This is a unique number used to identify the transaction. This value can be used to complete the Auth Only transaction. |
| ssl_txn_time | Date and time when the transaction was processed, returned only when <code>ccgettoken</code> is requested with Account verification flag. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). Returned only when <code>ccgettoken</code> is requested with Account verification flag. |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). Returned only when <code>ccgettoken</code> is requested with Account verification flag. |
| ssl_approval_code | Transaction approval code, returned only when <code>ccgettoken</code> is requested with Account verification flag. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card Return (ccreturn)

The `ccreturn` transaction is used to issue a partial or a full return (refund) to a cardholder's credit card using the transaction ID of the original sale or force transaction. This will guarantee that the same credit card used previously for the purchase is the one being refunded.

Users may choose to generate a partial return by passing the original transaction ID of the sale or force transaction and an amount that is less than the original amount, or a full return by passing the original transaction ID only without the amount. Enhanced credits for an amount higher than the original sale/force amount are not allowed.

Notes:

- Converge will continue to allow merchants to refund credit card transactions using full card number/track data per current functionality using ccredit. We strongly advise however, to use the enhanced credits in order to minimize risks associated with refund abuse.
 - The `ssl_show_form` property does not apply on Void transactions.
-

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Enhanced Return/Credit (<code>creturn</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_txn_id</code> | Y | Unique identifier returned on the original transaction. |
| <code>ssl_amount</code> | N | Amount to be refunded in full or partial. Number with two decimal places. Must be less or equal to the original purchase, if not supplied original full amount is refunded. For example: 1.00. |

Response

| Output Field Name | Description |
|---------------------------------|--|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| <code>ssl_result_message</code> | Transaction result message. Example: APPROVAL. |
| <code>ssl_txn_id</code> | Transaction identification number. This is a unique number used to identify the transaction. |
| <code>ssl_txn_time</code> | 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. |
| <code>ssl_amount</code> | Transaction amount. Returned based on merchant setup. |
| <code>ssl_approval_code</code> | Transaction approval code. |
| <code>ssl_avs_response</code> | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |

| Output Field Name | Description |
|-----------------------|---|
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_cardholder_amount | Only returned on DCC transactions. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card 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.

Note: The `ssl_show_form` property does not apply on Void transactions.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Credit Void (<code>ccvoid</code>). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_txn_id | Y | Unique identifier returned on the original transaction. |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | 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. Example: 03/18/2010 10:34:10 AM. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |
| ssl_approval_code | 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |
| errorMessage | Detailed explanation of the error returned <i>only</i> 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. |

Examples

Example 1: process.do(false)

To void the transaction submit a transaction request using HTTPS, either by the HTTP GET or POST with the Transaction ID associated to the transaction you wish to modify. The card number or track data should not be sent. Shown below are the key value pairs from the header by themselves for a credit card void transaction:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_transaction_type=ccvoid
ssl_txn_id =1016413275E60BB4EC-B4C6-FD4D-A878-F70C3372C986
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
```

By specifying the `http://www.url.com/cgi-bin/testtran.cgi` URL in the `ssl_receipt_apprvl_get_url` field for the redirect for the transaction above, the following values are an example of a response returned for the approved modified transaction:

```
ssl_card_number=00*****0000
ssl_amount=1.00
ssl_result=0
ssl_result_message=APPROVAL
ssl_txn_id=1016413275E60BB4EC-B4C6-FD4D-A878-F70C3372C986
ssl_approval_code=CVI368
ssl_txn_time=10/05/2008 10:50:55 AM
```

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 Transaction ID is invalid) are returned if the request is invalid:

```
errorCode=5040
errorName= Invalid Transaction ID
errorMessage= The transaction ID is invalid for this transaction
type
```

Example 2: processxml.do

This XML example demonstrates how to void a sale transaction. This is the response associated to a sale transaction:

```
xmldata=<txn>
<ssl_result>0</ssl_result>
<ssl_approval_code>N07032</ssl_approval_code>
<ssl_account_balance>0.00</ssl_account_balance>
<ssl_amount>1.00</ssl_amount>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_invoice_number>P0456</ssl_invoice_number>
<ssl_description>Purchase</ssl_description>
<ssl_txn_id>AA49315-F494FC02-661E-4426-A410-
A87F6249F26B</ssl_txn_id>
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_txn_time>07/21/2013 03:07:41 PM</ssl_txn_time>
<ssl_avs_response/>A</ssl_avs_response>
<ssl_cvv2_response>M</ssl_cvv2_response>
```

This is an example of a request and response to void the previous sale transaction:

Request

```
xmldata=<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>ccvoid</ssl_transaction_type>
<ssl_txn_id>AA49315-F494FC02-661E-4426-A410-
A87F6249F26B</ssl_txn_id>
```

Response

```
<txn>
  <ssl_result>0</ssl_result>
  <ssl_approval_code>N07033</ssl_approval_code>
  <ssl_account_balance>0.00</ssl_account_balance>
  <ssl_amount>1.00</ssl_amount>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_invoice_number>PO456</ssl_invoice_number>
  <ssl_description>Purchase</ssl_description>
  <ssl_txn_id>AA47AD-8B8D1518-FAD3-46AD-A0F4-
  790652B5617E</ssl_txn_id>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_txn_time>07/21/2013 03:15:31 PM</ssl_txn_time>
  <ssl_avs_response/>A</ssl_avs_response>
  <ssl cvv2 response>M</ssl cvv2 response>
```

Notes:

- 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 transaction ID obtained from the original transaction to void.
- Transaction update returns a new Transaction ID value.
- Transaction update returns the original data such as billing, shipping and custom values in the response.

Credit Card Completion (cccomplete)

A transaction type of `cccomplete` places an approved Auth Only transaction into the open batch. This transaction is considered the second half of a sale. An Auth Only transaction is converted to a sale when a transaction type of `cccomplete` is sent with a transaction ID that belongs to an Auth Only transaction. All transactions converted to Sale will be placed in the open batch and are handled the same way as Sale transactions.

An Auth Only transaction cannot be settled until a completion transaction is performed.

Users may choose to generate a partial sale by passing the original transaction ID for the Auth Only transaction for an amount that is less than the original amount. They can also generate a full completion by passing the original transaction ID without the amount. A completion request for an amount higher than the original Auth Only is not allowed.

Notes:

- Transaction status will not change. A pended Auth Only will be converted to a pended sale. Users have to login to the application in order to unpend a transaction. Transactions set to pend will not settle until they are set to unpend
- The `ssl_show_form` property does not apply on Void transactions.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Credit Card Completion (<code>cccomplete</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_txn_id</code> | Y | Unique identifier returned on the Auth Only transaction to be converted to Sale. |
| <code>ssl_amount</code> | N | Amount to be converted in full or partial. Number with two decimal places. Must be less or equal to the original purchase, If not supplied full amount will be converted. For example: 1.00. |

Response

| Output Field Name | Description |
|-----------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_card_number | Masked card number. Returned based on merchant setup. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_invoice_number | Invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| ssl_cardholder_amount | Only returned on DCC transactions. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

This example demonstrates how to complete an Auth Only transaction. This is the response associated to an Auth Only transaction:

```
<txn>
  <ssl_result>0</ssl_result>
  <ssl_approval_code>N19032</ssl_approval_code>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_txn_id>AA4843B-EEBFF0FE-0BDC-47A3-B4F6-
5D752E2FBE42</ssl_txn_id>
  <ssl_txn_time>07/21/2013 05:27:42 PM</ssl_txn_time>
  <ssl_amount>3.00</ssl_amount>
  <ssl_salestax>0.01</ssl_salestax>
  <ssl_account_balance>0.00</ssl_account_balance>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_cvv2_response>M</ssl_cvv2_response>
  <ssl_avs_response>A</ssl_avs_response>
  <ssl_invoice_number>1234</ssl_invoice_number>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_state>GA</ssl_state>
  <ssl_country>USA</ssl_country>
  <ssl_dynamic_dba>123456789012*ABCCORP</ssl_dynamic_dba>
</txn>
```

This is an example of a request and response to complete the previous auth transaction for the full amount:

Request

```
xmldata=<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_description>Keyed Sale API</ssl_description>
  <ssl_transaction_type>cccomplete</ssl_transaction_type>
  <ssl_txn_id>AA4843B-EEBFF0FE-0BDC-47A3-B4F6-
  5D752E2FBE42</ssl_txn_id>
</txn>
```

Response

```
<txn>
  <ssl_result>0</ssl_result>
  <ssl_approval_code>N19032</ssl_approval_code>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_txn_id>AA4843B-EEBFF0FE-0BDC-47A3-B4F6-
  5D752E2FBE42</ssl_txn_id>
  <ssl_txn_time>07/21/2013 05:19:38 PM</ssl_txn_time>
  <ssl_amount>3.00</ssl_amount>
  <ssl_salestax>0.01</ssl_salestax>
  <ssl_account_balance>0.00</ssl_account_balance>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_cv2_response>M</ssl_cv2_response>
  <ssl_avs_response>A</ssl_avs_response>
  <ssl_invoice_number>1234</ssl_invoice_number>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_state>GA</ssl_state>
  <ssl_country>USA</ssl_country>
  <ssl_dynamic_dba>123456789012*ABCCORP</ssl_dynamic_dba>
</txn>
```

Credit Card Delete (ccdelete)

The `ccdelete` is a transaction that deletes and attempts a reversal on a Sale or Auth Only Credit transaction. A transaction that has been deleted from the batch cannot be recovered. This transaction type is typically 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.

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. To perform a `ccdelete`, you must submit the transaction ID received from the original transaction.

Note: The `ssl_show_form` property does not apply on Delete transactions.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Credit Card Delete (<code>ccdelete</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_txn_id</code> | Y | Unique identifier returned on the original transaction. |

Response

| Output Field Name | Description |
|---------------------------------|--|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| <code>ssl_result_message</code> | Transaction result message. Example: APPROVAL. |
| <code>ssl_txn_id</code> | Transaction identification number. This is a unique number used to identify the transaction. |
| <code>ssl_approval_code</code> | Transaction approval code. |
| <code>ssl_email</code> | Returned based on merchant setup. |
| <code>ssl_invoice_number</code> | Invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| <code>ssl_txn_time</code> | Date and time when the transaction was processed, Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 A.M. |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card Update Tip (ccupdatetip)

The `ccupdatetip` transaction is used to add, modify or reset a tip (gratuity) on an open approved credit card sale or force transactions using the original transaction ID. This transaction type is supported in the *Service* market segment. Tips are updated or added after the transaction has been processed, typically at the end of the day prior to settlement. The most current tip amount sent will reflect in the total amount of that transaction. Error 5040 is returned if adding a tip is attempted on an invalid transaction.

To perform a `ccupdatetip`, you must submit the transaction ID received from the original transaction along with the desired tip amount. This also will override the tip amount if present with the latest tip amount provided.

Notes:

- The `ssl_show_form` property does not apply on update tips.
- A tip amount of 0.00 removes or resets an existing tip on a transaction
- Tip amount can be sent in the cardholder amount
- You may send the Shift and Server ID to update this information on an existing transaction
- This transaction may be sent several times prior to settlement if needed. The last and most current tip sent will be processed as the tip amount

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Update Tip (<code>ccupdatetip</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |

| Input Field Name | Req? | Description |
|---------------------------|------|---|
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_txn_id | Y | Unique identifier returned on the original transaction, must be either a credit sale or credit force. |
| ssl_tip_amount | Y | Tip or gratuity amount to be added or updated, must be 2 decimal places, can be 0.00 to reset or remove the original tip from a transaction. Example: 1.00. |
| ssl_cardholder_tip_amount | C | Use only for DCC transaction. Tip or gratuity amount to be added or updated in the cardholder currency if the original transaction was processed as a DCC transaction and the amount authorized was in the cardholder currency, must be two decimal places, can be 0.00 to reset or remove the original tip from a DCC transaction. Example: 1.00. |
| ssl_server | N | Server ID, this is the clerk, cashier, waiter or waitress identification number, can be used for reporting purposes. Alphanumeric, Example: Jack. |
| ssl_shift | N | Shift, can refer to or be used to identify time period, course or type of service, and can be used for reporting purposes. Alphanumeric, Example: Lunch. |

Response

| Output Field Name | Description |
|-------------------------|--|
| ssl_result_message | Tip update result message: a result of SUCCESS indicates the tip was updated or added successfully. ERROR indicates the tip was not added or updated successfully. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |
| ssl_amount | Transaction total amount including tip. Returned based on merchant setup. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. |
| ssl_server | Server identification number or name sent on request. Returned based on merchant setup. |
| ssl_shift | Shift sent on request, Returned based on merchant setup. |
| ssl_conversion_rate | Only returned on DCC transactions. |
| ssl_cardholder_currency | Only returned on DCC transactions. |

| Output Field Name | Description |
|----------------------------|---|
| ssl_cardholder_amount | Total amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_base_amount | Base amount in cardholder currency, only returned on DCC transactions. |
| ssl_cardholder_tip_amount | Tip amount in cardholder currency, only returned on DCC transactions. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

The following XML code example demonstrates how to update the tip on a previously approved transaction. The previously approved transaction may or may not have a tip. This will override the tip amount with the latest tip amount provided. This request can be performed several times if needed on a single transaction.

In this example the tip amount that was previously added to the transaction (2.00) will be updated to 5.00, the shift and server values have been updated as well.

Initial Request

```

xmldata=<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>ccupdatetip</ssl_transaction_type>
  <ssl_txn_id>TEST43B-D0638677-26EB-40F5-B2F9-
  3AF2545DE144</ssl_txn_id>
  <ssl_tip_amount>5.00</ssl_tip_amount>
  <ssl_shift>lunch</ssl_shift>
  <ssl_server>Jane</ssl_server>

```

Response Receipt

```
xmldata=<txn>
  <ssl_result_message>SUCCESS</ssl_result_message>
  <ssl_txn_id>101641221593ACBA6-BAFD-76B7-4948-
  B3DE68CFD0CC</ssl_txn_id>
  <ssl_amount>15.00</ssl_amount>
  <ssl_base_amount>10.00</ssl_base_amount>
  <ssl_tip_amount>5.00</ssl_tip_amount>
  <ssl_shift>lunch</ssl_shift>
  <ssl_server>Jane</ssl_server>
</txn>
```

Credit Card 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 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 Credit

Notes:

- The `ssl_show_form` property does not apply on adding a signature.
- All signature images must be BASE64 encoded in the following supported formats:
- Signature is not allowed for the e-Commerce market segment.
- Signature can only be added to an approved transaction
- Signature can only be added to a transaction that has no signature

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Credit Signature (<i>ccsignature</i>). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_image_type | Y | Image format. Possible values, must be capital: <ul style="list-style-type: none"> • GIF • TIF • JPG • PNG |
| ssl_signature_image | Y | BASE 64 Encoded version of an IMAGE. |
| ssl_txn_id | Y | Unique identifier returned on the original transaction. |

Response

| Output Field Name | Description |
|--------------------|---|
| 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 or added successfully. ERROR indicates the image was not added or imported successfully. |
| ssl_user_id | Converge User ID. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

The following XML code example demonstrates how to add a signature to a previously approved sale transaction. The original transaction ID obtained from the approved Sale must be passed along with the signature data:

Initial Sale Request

```

xmldata=<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_amount>1.00</ssl_amount>
  <ssl_track_data>%B0000000000000000^NOVA
CORPORATION^2212101543213961456?;
0000000000000000=2212101543213961456?</ssl_track_data>
</txn>

```

Sale Response

```

<txn>
  <ssl_approval_code>N21032</ssl_approval_code>
  <ssl_cvv2_response />
  <ssl_exp_date>1222</ssl_exp_date>
  <ssl_departure_date />
  <ssl_account_balance>0.00</ssl_account_balance>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_invoice_number />
  <ssl_amount>1.00</ssl_amount>
  <ssl_result>0</ssl_result>
  <ssl_txn_id>AA49315-DCA463E6-F924-4CA8-9FBF-
01220FBACA19</ssl_txn_id>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_completion_date />
  <ssl_txn_time>01/28/2014 10:21:31 PM</ssl_txn_time>
  <ssl_avs_response />
</txn>

```

Adding Signature Request

```
xmldata=<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_description>Keyed Sale API</ssl_description>
  <ssl_transaction_type>ccsignature</ssl_transaction_type>
  <ssl_txn_id>AA49315-DCA463E6-F924-4CA8-9FBF-
  01220FBACA19</ssl_txn_id>
</txn>
```

Adding Signature Response

```
xmldata=<txn>
  <ssl_transaction_type>CCSIGNATURE</ssl_transaction_type>
  <ssl_user_id>kmintz</ssl_user_id>
  <ssl_result>0</ssl_result>
  <ssl_result_message>SUCCESS</ssl_result_message>
</txn>
```

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.

Notes:

- No track, track 1 or track 2 magnetic stripe reads are allowed.
- CVV data cannot be passed, as it should not be stored in the system.
- The Merchant Admin (MA) user ID cannot be used

Credit Card Add Recurring Transaction (ccaddrecurring)

The `ccaddrecurring` is a transaction that adds a recurring record to Converge 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.

Request

| Input Field Name | Req? | Description |
|-------------------------|-------------|--|
| ssl_transaction_type | Y | Add Recurring (ccaddrecurring). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (available only for process.do), set to false otherwise. |
| ssl_card_number | Y | Credit Card Number as it appears on the credit card. |
| ssl_exp_date | Y | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| ssl_amount | Y | Transaction Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |
| ssl_next_payment_date | Y | Next payment date. Format MM/DD/YYYY. |
| ssl_billing_cycle | Y | Billing cycle. Valid returned values, all caps and no hyphens: <ul style="list-style-type: none"> • DAILY • WEEKLY • BIWEEKLY • SEMIMONTHLY • MONTHLY • BIMONTHLY • QUARTERLY • SEMESTER • SEMIANNUALLY • ANNUALLY • SUSPENDED |
| ssl_bill_on_half | C | Half of the month or Semimonthly indicator. Valid values are 1 and 2: <ul style="list-style-type: none"> • 1 = the 1st and the 15th of the month • 2 = the 15th and the last day of the month |

| Input Field Name | Req? | Description |
|--------------------|------|--|
| ssl_end_of_month | C | <p>End of month indicator. Valid values Y or N</p> <p>You must indicate if the recurring transaction will be processed on the last day of the month for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 30-Apr • 30-June • 30-Sept • 30-Nov • 28-Feb (non-leap year) • 29-Feb (leap year) <p>You also need to provide the end of the month indicator for the annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 28-Feb (non-leap year) • 29-Feb (leap year) |
| ssl_skip_payment | N | Skip Payment field. Valid values: Y for yes or N for no. Defaulted to N. |
| ssl_first_name | N | Cardholder first name |
| ssl_last_name | N | Cardholder last name |
| ssl_avs_zip | N | Cardholder ZIP code. |
| ssl_avs_address | N | Cardholder Address. |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_dynamic_dba | N | User only with a terminal that is setup with DBA. DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup. |
| ssl_customer_code | N | Recommended for purchasing cards. The Customer Code or PO Number that appears on the cardholder's credit card billing statement. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |

Response

| Output Field Name | Description |
|---------------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents a successful recurring transaction. |
| ssl_result_message | Transaction result message. A result of SUCCESS indicates the recurring was successfully added. |
| ssl_user_id | Converge User ID as configured on Converge. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| 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_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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example**Example 1: process.do(false)**

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

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccaddrecurring  
ssl_card_number=0000000000000000  
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
```

Converge then returns a response to the POST 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.

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=00*****0000
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
```

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.

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
```

Example 2: process.do(true)

This example demonstrates the initiation of a minimal recurring transaction in which Converge 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 Converge Gateway will automatically add this customer to the recurring Billing database and run the payment every month.

Send a ccaddrecurring request

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccaddrecurring
ssl_show_form=true
ssl_amount=9.95
ssl_billing=MONTHLY
ssl_result_format=HTML
```

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="00*****0000">
<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 3: processxml.do

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

ccaddrecurring request

```

xmldata=<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>ccaddrecurring</ssl_transaction_type>
  <ssl_card_number>0000000000000000</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>
</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>00*****00000</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>

```

Credit Card Update Recurring Transaction (ccupdaterecurring)

The `ccupdaterecurring` is a transaction that updates a recurring record in Converge. To perform a `ccupdaterecurring`, you must submit the recurring ID received from the original recurring transaction.

Note: The `ssl_show_form` property does not apply on Update transactions.

Request

| Input Field Name | Req? | Description |
|------------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Update Recurring (<code>ccupdaterecurring</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_recurring_id</code> | Y | The ID number of the recurring record to be updated. This value, was returned when the original recurring record was added. Alphanumeric. |
| <code>ssl_card_number</code> | N | Credit Card Number as it appears on the credit card. |
| <code>ssl_exp_date</code> | N | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| <code>ssl_amount</code> | N | Transaction Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |
| <code>ssl_next_payment_date</code> | N | Next payment date. Format MM/DD/YYYY. |
| <code>ssl_billing_cycle</code> | N | Billing cycle. Valid returned values, all caps and no hyphens: <ul style="list-style-type: none"> • DAILY • WEEKLY • BIWEEKLY • SEMIMONTHLY • MONTHLY • BIMONTHLY • QUARTERLY • SEMESTER • SEMIANNUALLY • ANNUALLY • SUSPENDED |

| Input Field Name | Req? | Description |
|--------------------|------|--|
| ssl_bill_on_half | C | <p>Half of the month or Semimonthly indicator.</p> <p>Valid values are 1 and 2:</p> <ul style="list-style-type: none"> • 1 = The 1st and the 15th of the month • 2 = The 15th and the last day of the month |
| ssl_end_of_month | C | <p>End of month indicator. Valid values Y or N</p> <p>You must indicate if the recurring transaction will be processed on the last day of the month for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 30-Apr • 30-June • 30-Sept • 30-Nov • 28-Feb (non-leap year) • 29-Feb (leap year) <p>You also need to provide the end of the month indicator for the annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 28-Feb (non-leap year) • 29-Feb (leap year) |
| ssl_skip_payment | N | <p>Skip Payment field.</p> <p>Valid values: Y for yes or N for no. Defaulted to N.</p> |
| ssl_first_name | N | Cardholder first name |
| ssl_last_name | N | Cardholder last name |
| ssl_avs_zip | N | Cardholder ZIP code. |
| ssl_avs_address | N | Cardholder Address. |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_dynamic_dba | N | <p>User only with a terminal that is setup with DBA.</p> <p>DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup.</p> |
| ssl_customer_code | N | <p>Recommended for purchasing cards.</p> <p>The Customer Code or PO Number that appears on the cardholder's credit card billing statement.</p> |
| ssl_salestax | N | <p>Recommended for purchasing cards.</p> <p>Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction.</p> |

Response

| Output Field Name | Description |
|---------------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents a successful transaction. |
| ssl_result_message | Transaction result message. A result of SUCCESS indicates the recurring was successfully updated. ERROR indicates the recurring was not updated successfully. |
| ssl_amount | Recurring amount. |
| ssl_billing_cycle | Billing cycle. |
| ssl_card_number | Card number. Returned in masked format. |
| ssl_next_payment_date | Next payment due date. |
| ssl_number_of_payments | Current number of payments run so far. Numeric. Returned by Converge. 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_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. |
| errorCode | Error code returned <i>only</i> if an error occurred. Typically, when the transaction failed validation or the request is incorrect. This prevents 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

The following example demonstrates the key value pairs needed to pass 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.

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccupdaterecurring
ssl_show_form=false
ssl_recurring_id=AA484C3-B08B6F1B-4765-A1FF-C0BC-5722F21A0EB6
ssl_billing=SUSPENDEED
```

Credit Card Delete Recurring Transaction (ccdeleterecurring)

The `ccdeleterecurring` is a transaction that deletes a recurring record in Converge. To perform a `ccdeleterecurring`, you must submit the recurring ID received from the original recurring transaction.

Note: The `ssl_show_form` property does not apply on Delete transactions.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Delete Recurring (<code>ccdeleterecurring</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_recurring_id</code> | Y | The ID number of the recurring record to be updated. This value was returned when the original recurring record was added. Alphanumeric. |

Response

| Output Field Name | Description |
|---------------------------------|---|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents a successful transaction. |
| <code>ssl_result_message</code> | Transaction result message. A result of SUCCESS indicates the recurring was successfully deleted. ERROR indicates the recurring was not deleted successfully. |

| 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card 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. To perform a `ccrecurringsale`, you must submit the recurring ID received from the original recurring transaction.

Note: The `ssl_show_form` property does not apply when submitting payments.

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Submit Recurring Payment (<code>ccrecurringsale</code>). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_recurring_id | Y | The ID number of the recurring record to be submitted for payment. This value was returned when the original recurring record was added. Alphanumeric. |

| Input Field Name | Req? | Description |
|------------------|------|---|
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token after processing the recurring sale. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |
| ssl_add_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to generate a token and store it in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |

Response

| Output Field Name | Description |
|------------------------|---|
| ssl_result | Outcome of a transaction. A response that contains an <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | 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 | 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. |
| ssl_amount | Transaction authorized or approved amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_billing_cycle | Billing cycle. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_invoice_number | 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 Converge. This number represents the number of payments run on the system. |

| Output Field Name | Description |
|-------------------|---|
| ssl_recurring_id | ID number of the recurring record. Alphanumeric. Returned on SUCCESS only. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

The following example demonstrates the key value pairs needed to pass the minimum required data to send a recurring sale outside of the billing cycle. The recurring ID obtained from the original transaction must be passed. This transaction will increase the total payments.

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccrecurringsale
ssl_show_form=false
ssl_recurring_id=AA484C3-B08B6F1B-4765-A1FF-C0BC-5722F21A0EB6
```

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 reads are allowed. Also, CVV data cannot be passed, as it should not be stored in the system.

Notes:

- No track, track 1 or track 2 magnetic stripe reads are allowed.
- CVV data cannot be passed, as it should not be stored in the system.
- The Merchant Admin (MA) user ID cannot be used

Credit Card Add Installment Transactions (ccaddinstall)

The `ccaddinstall` is a transaction that adds an installment record to Converge 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.

Request

| Input Field Name | Req? | Description |
|-------------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Add Installment (<code>ccaddinstall</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | Y | Set value to true to show the Converge Payment Form (available only for <code>process.do</code>), set to false otherwise. |
| <code>ssl_card_number</code> | Y | Credit Card Number as it appears on the credit card. |
| <code>ssl_exp_date</code> | Y | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| <code>ssl_amount</code> | Y | Transaction Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |
| <code>ssl_total_installments</code> | Y | Number of payments, Numeric. |
| <code>ssl_next_payment_date</code> | Y | Next payment date. Format MM/DD/YYYY. |
| <code>ssl_billing_cycle</code> | Y | Billing cycle. Valid returned values, all caps and no hyphens: <ul style="list-style-type: none"> • DAILY • WEEKLY • BIWEEKLY • SEMIMONTHLY • MONTHLY • BIMONTHLY • QUARTERLY • SEMESTER • SEMIANNUALLY • ANNUALLY • SUSPENDED |
| <code>ssl_bill_on_half</code> | C | Half of the month or Semimonthly indicator. Valid values are 1 and 2: <ul style="list-style-type: none"> • 1 = The 1st and the 15th of the month • 2 = The 15th and the last day of the month |

| Input Field Name | Req? | Description |
|--------------------|------|--|
| ssl_end_of_month | C | <p>End of month indicator. Valid values Y or N</p> <p>You must indicate if the recurring transaction will be processed on the last day of the month for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 30-Apr • 30-June • 30-Sept • 30-Nov • 28-Feb (non-leap year) • 29-Feb (leap year) <p>You also need to provide the end of the month indicator for the annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • 28-Feb (non-leap year) • 29-Feb (leap year) |
| ssl_skip_payment | N | <p>Skip Payment field.</p> <p>Valid values: Y for yes or N for no. Defaulted to N.</p> |
| ssl_first_name | N | Cardholder first name |
| ssl_last_name | N | Cardholder last name |
| ssl_avs_zip | N | Cardholder ZIP code. |
| ssl_avs_address | N | Cardholder Address. |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_dynamic_dba | N | <p>User only with a terminal that is setup with DBA.</p> <p>DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup.</p> |
| ssl_customer_code | N | <p>Recommended for purchasing cards.</p> <p>The Customer Code or PO Number that appears on the cardholder's credit card billing statement.</p> |
| ssl_salestax | N | <p>Recommended for purchasing cards.</p> <p>Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction</p> |

Response

| Output Field Name | Description |
|---------------------------|---|
| ssl_result | Outcome of a transaction. A response that contains a <code>ssl_result</code> of 0 represents a successful transaction. |
| ssl_result_message | Transaction result message. A result of SUCCESS indicates the installment was successfully added. ERROR indicates the installment was not added successfully. |
| ssl_amount | Recurring amount. |
| ssl_billing_cycle | Billing cycle. |
| ssl_card_number | Card number. Returned in 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_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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example**Example 1: process.do (true):**

The following example demonstrates a basic request 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 will call the payment form that displays the customer's payment and asks for their credit card number and expiration date. After the user enters the information and clicks the process button, Converge 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. Shown below are the key value pairs from the header by themselves for this transaction:

```
ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccaddinstall
ssl_show_form=true
ssl_billing=SEMIMONTHLY
ssl_next_payment_date=01/01/2012
ssl_bill_on_half=1
ssl_amount=75.00
ssl_total_installments=10
```

Example 2 processxml.do:**Request**

```
xmldata=<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_tr <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>5.00</ssl_amount>
  <ssl_transaction_type>ccaddinstall</ssl_transaction_type>
  <ssl_customer_code>FF1234</ssl_customer_code>
  <ssl_billing_cycle>WEEKLY</ssl_billing_cycle>
  <ssl_next_payment_date>01/30/2014</ssl_next_payment_date>
  <ssl_total_installments>10</ssl_total_installments>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
</txn>
```

Response

```
<txn>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_result_message>SUCCESS</ssl_result_message>
  <ssl_country>USA</ssl_country>
  <ssl_city>Atlanta</ssl_city>
  <ssl_number_of_payments>0</ssl_number_of_payments>
  <ssl_billing_cycle>WEEKLY</ssl_billing_cycle>
  <ssl_start_payment_date>01/30/2014</ssl_start_payment_date>
  <ssl_transaction_type>CCADDINSTALL</ssl_transaction_type>
  <ssl_total_installments>10</ssl_total_installments>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_salestax>0</ssl_salestax>
  <ssl_next_payment_date>01/30/2014</ssl_next_payment_date>
  <ssl_address2/>
  <ssl_skip_payment>N</ssl_skip_payment>
  <ssl_first_name>John</ssl_first_name>
  <ssl_recurring_batch_count>23</ssl_recurring_batch_count>
  <ssl_amount>5.00</ssl_amount>
  <ssl_state>GA</ssl_state>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_installment_id>AA49315-AB9B8587-ECE6-445A-A2A4-
  3B00A9985C0A</ssl_installment_id>
</txn>
```

Credit Card Update Installment Transactions (ccupdateinstall)

The `ccupdateinstall` is a transaction that updates an installment record in Converge. To perform a `ccupdateinstall`, you must submit the installment ID received from the original installment transaction.

Note: The `ssl_show_form` property does not apply on Update transactions.

Request

| Input Field Name | Req? | Description |
|-------------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Update installment (<code>ccupdateinstall</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_installment_id</code> | Y | The ID number of the installment record to be updated. Alphanumeric. |
| <code>ssl_card_number</code> | Y | Credit Card Number as it appears on the credit card. |
| <code>ssl_exp_date</code> | Y | Credit Card Expiry Date as it appears on credit card formatted as MMYYY. |
| <code>ssl_amount</code> | Y | Transaction Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |
| <code>ssl_total_installments</code> | N | Number of payments, Numeric. |
| <code>ssl_next_payment_date</code> | N | Next payment date. Format MM/DD/YYYY. |
| <code>ssl_billing_cycle</code> | N | Billing cycle. Valid returned values, all caps and no hyphens: <ul style="list-style-type: none"> • DAILY • WEEKLY • BIWEEKLY • SEMIMONTHLY • MONTHLY • BIMONTHLY • QUARTERLY • SEMESTER • SEMIANNUALLY • ANNUALLY • SUSPENDED |

| Input Field Name | Req? | Description |
|--------------------|------|--|
| ssl_bill_on_half | C | <p>Half of the month or Semimonthly indicator.</p> <p>Valid values are 1 and 2:</p> <ul style="list-style-type: none"> 1 = The 1st and the 15th of the month 2 = The 15th and the last day of the month |
| ssl_end_of_month | C | <p>End of month indicator. Valid values Y or N</p> <p>You must indicate if the recurring transaction will be processed on the last day of the month for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> 30-Apr 30-June 30-Sept 30-Nov 28-Feb (non-leap year) 29-Feb (leap year) <p>You also need to provide the end of the month indicator for the annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> 28-Feb (non-leap year) 29-Feb (leap year) |
| ssl_skip_payment | N | <p>Skip Payment field.</p> <p>Valid values: Y for yes or N for no. Defaulted to N.</p> |
| ssl_first_name | N | Cardholder first name |
| ssl_last_name | N | Cardholder last name |
| ssl_avs_zip | N | Cardholder ZIP code. |
| ssl_avs_address | N | Cardholder Address. |
| ssl_invoice_number | N | The invoice or ticket number. |
| ssl_dynamic_dba | N | <p>User only with a terminal that is setup with DBA.</p> <p>DBA Name provided by the merchant with each transaction. The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup.</p> |
| ssl_customer_code | N | <p>Recommended for purchasing cards.</p> <p>The Customer Code or PO Number that appears on the cardholder's credit card billing statement.</p> |
| ssl_salestax | N | <p>Recommended for purchasing cards.</p> <p>Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction.</p> |

Response

| Output Field Name | Description |
|---------------------------|---|
| ssl_result | Outcome of a transaction. A response that contains a <code>ssl_result</code> of 0 represents a successful transaction. |
| ssl_result_message | Transaction result message. A result of SUCCESS indicates the installment was successfully updated. ERROR indicates the installment was not updated successfully. |
| ssl_installment_id | ID number of the installment record updated. Returned on SUCCESS only. |
| ssl_amount | Recurring amount. |
| ssl_billing_cycle | Billing cycle. |
| ssl_card_number | Card number. Returned in masked format. |
| ssl_next_payment_date | Next payment due date. |
| ssl_number_of_payments | Current number of payments run so far. Numeric. Returned by Converge. 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_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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card Delete Installment Transactions (ccdeleteinstall)

The `ccdeleteinstall` is a transaction that deletes an installment record in Converge. To perform a `ccdeleteinstall`, you must submit the installment ID received from the original installment transaction.

Note: The `ssl_show_form` property does not apply on Delete transactions.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Delete Installment (<code>ccdeleteinstall</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_installment_id</code> | Y | ID number of the recurring record to be deleted. Alphanumeric. |

Response

| Output Field Name | Description |
|--|---|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents a successful transaction. |
| <code>ssl_result_message</code> | Transaction result message. A result of SUCCESS indicates the installment was successfully deleted. ERROR indicates the installment was not deleted successfully. |
| <code>ssl_installment_id</code> | ID number of the installment record updated. Returned on SUCCESS, only if the installment record was deleted successfully. |
| <code>ssl_recurring_batch_count</code> | Current number of transactions sitting in the recurring batch after the installment transaction has been deleted. |
| <code>errorCode</code> | Error code returned <i>only</i> 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. |
| <code>errorMessage</code> | Detailed explanation of the error returned <i>only</i> 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. |
| <code>errorName</code> | Error name or reason for the error returned <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card 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. To perform a `ccinstallsale`, you must submit the installment ID received from the original installment transaction.

Note: The `ssl_show_form` property does not apply when submitting payments.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Submit Installment Payment (<code>ccinstallsale</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_installment_id</code> | Y | ID number of the recurring record to be authorized. Alphanumeric. |
| <code>ssl_get_token</code> | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |
| <code>ssl_add_token</code> | N | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to generate a token and store it in Card Manager. Valid value: Y (add token), N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |

Response

| Output Field Name | Description |
|---------------------------------|--|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| <code>ssl_result_message</code> | Transaction result message. Example: APPROVAL, PARTIAL APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages. |

| Output Field Name | Description |
|------------------------|---|
| ssl_amount | Transaction authorized or approved amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_billing_cycle | Billing cycle. |
| ssl_card_number | 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 | 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 Converge. |
| ssl_start_payment_date | Date when the first payment started. If recently added, start date is same as next payment. |
| ssl_total_installments | Total number of payments. Numeric. |
| ssl_txn_id | 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Batch Import Transactions

This message format is for importing and processing batch files of credit card or recurring or installment transactions.

The ability to import batch files is only possible through `processBatch.do` through an HTTP POST using one of the following transaction types:

- **ccimport**
To process a file of Credit Cards transactions.
- **cctokenimport**
To process a file of Credit Card information for token generation.
- **ccrecimport**
to process a file of Credit Card Recurring transactions.

Every batch import HTTP request must include the `enctype` attribute of `multipart/form-data`, a properly formatted CSV or XML batch file along with the individual key value pairs formatted file. The POST request does not support XML formatted data at this time.

Batch files must meet the following criteria:

- File must be binary and the extension must be CSV or XML.
- Only one file can be imported at any given time for any terminal.
- There is a limit of five files per day per terminal.
- The same file cannot be imported in the same terminal within a 24-hour period.
- There is a limit of 500 transactions per file.
- The CSV file must be formatted properly as follows:
 - The first line in the CSV file will contain the proper header with elements. The elements in the header and what they mean are specified in the table below.
 - Remaining lines in the CSV file will contain the transaction data. Data for a transaction should be in a single line. Data for each new transaction must be in a new line.
 - Data that pertains to an element in a transaction will be separated by a comma delimiter.
 - Order of the values provided must follow the order of fields in the header. If no value is present to accommodate the element in the header, comma and double quotes without a value must be provided as placeholder.
 - Use the double quotes to enclose each field and value.

- XML Files must be formatted properly as follows:
 - XML files must contain the root xml beginning and ending element `<txnimport>` for Credit Batch Import or `<txnrecimport>` for Recurring Batch Import.
 - Data for each transaction will be nested and embedded between beginning and ending elements `<txn>`.
 - A transaction data that resides between beginning `<txn>` and ending `</txn>` will contain all needed XML elements and values to process a single transaction for that type of transaction as specified in the table below.
 - The order of the tags in the XML format is not important.

Credit Card Batch Import (ccimport)

The `ccimport` is a request that allows you to import and process a batch file of credit card transactions. This request should include the batch file to import in XML or CSV format. Once the file is uploaded, Converge will validate the file and return a response via `processBatch.do`. This response is simply to indicate if the file upload was successful or not. Transactions in the file will be sent for authorization and are shown in the appropriate batches in the User Interface. The import response files can be viewed and downloaded from the User Interface.

Request

| Input Field Name | Req? | Description |
|------------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Credit Card Batch Import (<code>ccimport</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_import_file</code> | Y | Path and location and the name of the import file. The entire entry can have a maximum size of 255, the file name can only be 30 characters (including the extension) in length, and the extension can only be CSV or XML. The path must be sent along with the file name. Example: C:\Program Files\Elavon\VirtualMerchant\Import\ImportFile.csv. |
| <code>ssl_response_file</code> | Y | Response file friendly name, cannot be more than 30 characters and should not contain any of the following characters (<code>\ / : * ? " < > </code>). |
| <code>ssl_do_merchant_email</code> | N | Determine if the merchant would like an email when the import file is completed, possible values: T to send an email or F not to send. |
| <code>ssl_merchant_email</code> | N | Merchant's email address, if email is not specified and <code>ssl_do_merchant_email</code> is set to T then an email will be sent to the terminal email. |

| Input Field Name | Req? | Description |
|-------------------------|------|---|
| ssl_result_format | N | Valid values: HTML, XML, ASCII. When set to ASCII Virtual Merchant will generate a plain text key-value document. Default is HTML if not sent. |
| ssl_receipt_link_method | N | Only valid option is REDG so no receipt is displayed and data redirected to the link defined in request in the ssl_receipt_link_url or terminal settings. |
| ssl_receipt_link_url | N | Target of the Redirect link for the batch import results. Ignored when the ssl_result_format = ASCII and ssl_receipt_link_method = REDG. |
| ssl_error_url | N | If present, the error will be redirected to the URL specified including the errorCode, errorName and errorMessage fields. |

Response

| Output Field Name | Description |
|----------------------|---|
| ssl_result | Batch Import Result code: a result of 0 indicates the file was successfully imported. A result of 1 indicates that the file import failed. |
| ssl_result_message | Batch Import Result message: a result of File upload successful indicates the file was imported successfully. File upload failed indicates the file was not imported successfully. |
| ssl_transaction_type | Transaction type (ccimport). |
| ssl_user_id | Converge User ID. |
| ssl_number_trans | Number of transactions imported in the import file. |
| ssl_response_file | Response file name. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Batch Import File Format

A Credit Batch Import file is a properly formatted CSV or XML batch file of credit card transactions. This file must be uploaded with transaction type `ccimport`.

The first line in a CSV file is the header which contains the field names and values as outlined in the table. Each field in the header must be enclosed within double quotes and separated by a comma. Remaining lines in the file are for transaction data. Each transaction must be in a new line. The data in the transaction line must be placed in the same order as its corresponding field value in the header, enclosed within double quotes and separated by a comma.

The XML file contains one single `<txnimport>` element and multiple transaction `<txn>` nested elements, every transaction is a set of XML tags and its corresponding values as shown in the table. The number of `<txn>` corresponds to the number of transactions in the file. All elements are closed in the order that they were opened.

Notes:

- CVV2/CVC/CID values and Track Data should never be stored, and therefore cannot be passed to Converge on the file.
 - You must provide either the card number `ssl_card_number` or the token `ssl_token` from a previously tokenized card number in each record.
 - A single file can have a mix of tokens and card numbers.
-

The file must be formatted as follows:

| Header Field Name | Req? | Length | Description |
|---------------------------------------|------|--------|---|
| <code>ssl_card_number</code> | Y | 19 | Credit card number. |
| <code>ssl_token</code> | C | 19 | Use only with a terminal that is setup with <i>Tokenization</i> . Credit Card Token, previously generated from a card number. A token can be stored and used as a substitute for a card number. |
| <code>ssl_exp_date</code> | Y | 4 | In MMY format. |
| <code>ssl_amount</code> | Y | 13 | Must be in decimal form, no dollar sign. |
| <code>ssl_transaction_currency</code> | N | 3 | Use only with a terminal that is setup with <i>Multi-Currency</i> . Transaction currency alphanumeric code must be included in the request to indicate the currency in which you wish to process. If omitted the terminal default currency is assumed. For example: USD, CAD, JPY. More than 94 currencies are supported. Refer to the ISO Currency Codes section for an extensive list of available currencies. |

| Header Field Name | Req? | Length | Description |
|------------------------|------|--------|---|
| ssl_transaction_type | Y | 20 | Available transaction types: <ul style="list-style-type: none"> ccsale (Sale) ccauthonly (Auth Only) ccavsonly (AVS Only) ccverify (Verification) cccredit (Credit) ccforce (Force) |
| ssl_customer_code | N | 17 | Customer code for purchasing card transactions. |
| ssl_salestax | N | 10 | Sales tax for purchasing card transactions. |
| ssl_invoice_number | N | 25 | Invoice number. |
| ssl_approval_code | C | 6 | Required for force transactions. |
| ssl_description | N | 255 | Transaction description. |
| ssl_company | N | 50 | Customer's company name. |
| ssl_first_name | N | 20 | Customer's first name. |
| ssl_last_name | N | 30 | Customer's last name. |
| ssl_avs_address | N | 30 | Cardholder's street address. |
| ssl_address2 | N | 30 | Customer's address line 2. |
| ssl_city | N | 30 | Customer's city. |
| ssl_state | N | 30 | Customer's state. |
| ssl_avs_zip | N | 9 | Customer's ZIP code used to process AVS. |
| ssl_country | N | 50 | Customer's country. |
| ssl_phone | N | 20 | Customer's phone number. |
| ssl_email | N | 100 | Customer's email address. |
| ssl_ship_to_company | N | 50 | Ship To company name. |
| ssl_ship_to_first_name | N | 20 | Ship To first name. |
| ssl_ship_to_last_name | N | 30 | Ship To last name. |
| ssl_ship_to_address1 | N | 30 | Ship To address line 1. |
| ssl_ship_to_address2 | N | 30 | Ship To address line 2. |
| ssl_ship_to_city | N | 30 | Ship To city. |
| ssl_ship_to_state | N | 30 | Ship To state. |
| ssl_ship_to_zip | N | 10 | Ship To ZIP code. |
| ssl_ship_to_country | N | 50 | Ship To country. |
| ssl_ship_to_phone | N | 20 | Ship To phone number. |

| Header Field Name | Req? | Length | Description |
|--------------------|------|--------|---|
| User Defined Field | N | | User defined, user should be able to pass up to 25 user defined fields, the value passed in the file should match the field name specified in the Terminal Setup Merchant Payment Fields Add New Field . |
| ssl_get_token | N | 1 | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token when passing card data. Valid values: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |

Examples

Batch Import Request

This is an example of a credit batch import request. The file contents must be sent in the `ssl_import_file`.

```

ssl_merchant_id=my_virtualmerchant_id
ssl_user_id=my_user_id
ssl_pin=my_pin
ssl_transaction_type=ccimport
ssl_import_file=C:\Program
Files\Elavon\VirtualMerchant\Import\ImportFile.csv
ssl_response_file=ImportFile051410025559
ssl_result_format=HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp

```

Batch Import Response

The application has indicated that the file upload was successful.

```
ssl_response_file=ImportFile051410025559
ssl_transaction_type=ccimport
ssl_start_date=05142010073000
ssl_end_date=05142010073200
ssl_number_trans=000100
ssl_user_id= my_User_ID
ssl_result=0
ssl_result_message=File Uploaded
ssl_recurring_batch_count = 250
```

This is an example of an error in an attempt to import a file with 900 recurring (`processxml.do`)

```
<txn>
  <errorCode>5062</errorCode>
  <errorMessage> File Exceeds Max Number of Transactions </errorMessage>
  <errorMessage> File contains more than 500 transactions.</errorMessage>
</txn>
```

Credit Batch Import File Example CSV

The following example demonstrates a CSV file. The first line is a header, which contains the field names. The contents of this file can be copied and saved in a text document with a CSV extension for testing and can be sent with a `ccimport` request:

```
"ssl_card_number","ssl_exp_date","ssl_amount","ssl_transaction_type","ssl
_approval_code","ssl_description"
"0000000000000000","0514","5.00","ccsale","","My First Sale"
"0000000000000000","0515","10.00","ccforce","12345","This is a force"
"0000000000000000","0517","12.00","ccauthonly","","Just Auth Only for now"
```

Credit Batch Import File Example XML

The following example demonstrates a properly nested XML file with two transactions. The contents of this file can be copied and saved in a text document with a CSV extension for testing and can be sent with a `ccimport` request:

```
<txnimport>
  <txn>
    <ssl_card_number>0000000000000000</ssl_card_number>
    <ssl_exp_date>1212</ssl_exp_date>
    <ssl_amount>2.00</ssl_amount>
    <ssl_transaction_type>ccsale</ssl_transaction_type>
  </txn>
  <txn>
    <ssl_card_number>0000000000000000</ssl_card_number>
    <ssl_exp_date>1213</ssl_exp_date>
    <ssl_amount>5.00</ssl_amount>
    <ssl_transaction_type>CCFORCE</ssl_transaction_type>
    <ssl_customer_code> FF1234</ssl_customer_code>
    <ssl_salestax>0.50</ssl_salestax>
    <ssl_invoice_number>1234</ssl_invoice_number>
    <ssl_approval_code>555555</ssl_approval_code>
    <ssl_description>test a force</ssl_description>
    <ssl_company/>
    <ssl_first_name>John</ssl_first_name>
    <ssl_last_name>Doe</ssl_last_name>
    <ssl_avs_address>123 Main</ssl_avs_address>
    <ssl_city>Atlanta</ssl_city>
    <ssl_state>GA</ssl_state>
    <ssl_avs_zip>30123</ssl_avs_zip>
    <ssl_country>USA</ssl_country>
  </txn>
</txnimport>
```

Credit Card Information Batch Import (cctokenimport)

The `cctokenimport` is a request that allows you to import and process a batch file of card numbers or recurring IDs in order to generate tokens; this is the bulk load of `ccgettoken`. This request should include the batch file to import in XML or CSV format. Once the file is uploaded, Converge will validate the file and return a response through `ProcessBatch.do`. This response is simply to indicate if the file upload was successful or not. Transactions in the file will be sent for tokenization and are shown into the appropriate batches in the User Interface. The import response files can be viewed and downloaded from the User Interface.

This transaction type is supported only when a terminal is setup for tokenization; refer to the [Tokenization](#) section for more information.

Request

| Input Field Name | Req? | Description |
|--------------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Credit Card Information Batch Import (<code>cctokenimport</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_import_file</code> | Y | Path and location and the name of the import file. The entire entry can have a maximum size of 255, the file name can only be 30 characters (including the extension) in length, and the extension can only be CSV or XML. The path must be sent along with the file name. Example: C:\Program Files\Elavon\VirtualMerchant\Import\ImportFile.csv. |
| <code>ssl_response_file</code> | Y | Response file friendly name, cannot be more than 30 characters and should not contain any of the following characters (\ / : * ? " < >). |
| <code>ssl_do_merchant_email</code> | N | Determine if the merchant would like an email when the import file is completed, possible values: T to send an email or F not to send. |
| <code>ssl_merchant_email</code> | N | Merchant's email address, if email is not specified and <code>ssl_do_merchant_email</code> is set to T then an email will be sent to the terminal email. |
| <code>ssl_result_format</code> | N | Valid values: HTML, XML, ASCII. When set to ASCII Virtual Merchant will generate a plain text key-value document. Default is HTML if not sent. |
| <code>ssl_receipt_link_method</code> | N | Only valid option is REDG so no receipt is displayed and data redirected to the link defined in request in the <code>ssl_receipt_link_url</code> or terminal settings. |

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_receipt_link_url | N | Target of the Redirect link for the batch import results. Ignored when the <code>ssl_result_format = ASCII</code> and <code>ssl_receipt_link_method = REDG</code> . |
| ssl_error_url | N | If present, the error will be redirected to the URL specified including the <code>errorCode</code> , <code>errorName</code> , and <code>errorMessage</code> fields. |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Batch Import Result code: a result of 0 indicates the file was successfully imported. A result of 1 indicates that the file import failed. |
| ssl_result_message | Batch Import Result message: a result of File upload successful indicates the file was imported successfully. File upload failed indicates the file was not imported successfully. |
| ssl_user_id | Converge User ID. |
| ssl_number_trans | Number of cards imported in the import file. |
| ssl_response_file | Response file name. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Credit Card Information Batch Import File Format

A Card Numbers Batch Import file is a properly formatted CSV or XML batch file of card numbers or recurring IDs or mix of both. This file must be uploaded with transaction type `cctokenimport`.

The first line in a CSV file is the header which contains the field names and values as outlined in the table. Each field in the header must be enclosed within double quotes and separated by a comma. The remaining lines in the file are for transaction data. Each transaction must be in a newline. The data in the transaction line must be placed in the same order as its corresponding field value in the header, enclosed within double quotes and separated by a comma.

The XML file will contain one single `<txnimport>` element and multiple transaction `<txn>` nested elements, every transaction is a set of XML tags and its corresponding values as shown in the table. The number of `<txn>` corresponds to the number of transactions in the file. All elements are closed in the order that they were opened.

Note: CVV2/CVC/CID values and Track Data should never be stored, therefore cannot be passed to Converge on file.

The file must be formatted as follows:

| Header Field Name | Req? | Length | Description |
|-----------------------------------|------|--------|---|
| <code>ssl_transaction_type</code> | Y | 20 | <code>ccgettoken</code> (Generate token) |
| <code>ssl_card_number</code> | C | 19 | Required for hand-keyed transactions. Credit Card Number as it appears on the credit card. |
| <code>ssl_exp_date</code> | C | 4 | Required for hand-keyed transactions. Expiration date as it appears on the credit card. Format MMY |
| <code>ssl_recurring_id</code> | C | 50 | Required for recurring or installment transactions. The ID number of the recurring record or installment record to be submitted for payment. This value was returned in the recurring ID when the original recurring record was added for recurring or populate with installment ID when the original installment was added for installment. Alphanumeric. |
| <code>ssl_verify</code> | N | 1 | Account Verify indicator to indicate that account verification is needed prior to generating a token. Valid values: Y, N. |
| <code>ssl_avs_address</code> | N | 30 | Recommended with Account Verify indicator, required with Add to Card Manager indicator. Cardholder's Street Address |

| Header Field Name | Req? | Length | Description |
|-------------------|------|--------|---|
| ssl_avs_zip | N | 9 | Recommended with Account Verify indicator, required with Add to Card Manager indicator. Customer's ZIP code used to process AVS |
| ssl_add_token | N | 1 | Use only with a terminal that is setup with <i>Tokenization</i> . Add to Card Manager indicator, used to indicate if you wish to store the token generated in Card Manager. Valid value: Y (add token) , N (do not add token) Defaulted to N To add the token to the card manager you must send the card data and cardholder first/last name, those are required. Once stored to Card Manager, the token number can be sent alone and will be used as a substitute for the stored information. |
| ssl_first_name | C | 20 | Required with Add to Card Manager indicator. Cardholder's first name. |
| ssl_last_name | C | 20 | Required with Add to Card Manager indicator. Cardholder's first name. |

Examples

Credit Card Information File Example CSV

The following example demonstrates a CSV file. The first line is a header, which contains the field names. The contents of this file can be copied and saved in a text document with a CSV extension for testing and can be sent with `cctokenimport` request:

```
"ssl_card_number","ssl_transaction_type","ssl_exp_date","ssl_avs_address","ssl_avs_zip","ssl_verify","ssl_add_token"
"0000000000000000","ccgettoken","1215","123 Main","31307","Y","Y"
"0000000000000000","ccgettoken","1216","","","N"
"5000000000000000","ccgettoken","1217","345 Main","99999","Y","Y"
```

Credit Card Information File Example XML

The following example demonstrates a properly nested XML file with 3 card information. The contents of this file can be copied and saved in a text document with a XML extension for testing and can be sent with `cctokenimport` request:

```
<txnimport>
  <txn>
    <ssl_card_number>0000000000000000</ssl_card_number>
    <ssl_exp_date>1215</ssl_exp_date>
    <ssl_transaction_type>CCGETTOKEN</ssl_transaction_type>
    <ssl_avs_address>123 Main</ssl_avs_address>
    <ssl_avs_zip>30123</ssl_avs_zip>
    <ssl_verify>Y</ssl_verify>
  </txn>
  <txn>
    <ssl_card_number>0000000000000000</ssl_card_number>
    <ssl_transaction_type>CCGETTOKEN</ssl_transaction_type>
    <ssl_exp_date>1215</ssl_exp_date>
    <ssl_verify>N</ssl_verify>
  </txn>
  <txn>
    <ssl_card_number>0000000000000000</ssl_card_number>
    <ssl_exp_date>1215</ssl_exp_date>
    <ssl_transaction_type>CCGETTOKEN</ssl_transaction_type>
    <ssl_verify>Y</ssl_verify>
    <ssl_avs_address>123 Main</ssl_avs_address>
    <ssl_avs_zip>30123</ssl_avs_zip>
    <ssl_add_token>Y</ssl_add_token>
    <ssl_first_name>john</ssl_first_name>
    <ssl_last_name>Doe</ssl_last_name>
  </txn>
</txnimport>
```

Credit Card Recurring Batch Import (ccrecimport)

The `ccrecimport` is a request that allows you to import and process a batch file of recurring and installment transactions. This request should include the batch file to import in XML or CSV format. Once the file is uploaded, Converge will validate the file and return a response via `processBatch.do`, this response is simply to indicate if the file upload was successful or not. Transactions in the file will be sent for authorization and are entered into the appropriate batches in the user interface. The import response files can be viewed and downloaded from the user interface.

Request

| Input Field Name | Req? | Description |
|--------------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Recurring Batch Import (<code>ccrecimport</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_import_file</code> | Y | Path, location and name of the import file. The entire entry can have a maximum size of 255. The file name can only be 30 characters (including the extension) in length and the extension can only be CSV or XML. The path must be sent along with the file name. Example: C:\Program Files\Elavon\VirtualMerchant\Import\ImportFile.csv. |
| <code>ssl_response_file</code> | Y | Response file friendly name, cannot be more than 30 characters and should not contain any of the following characters (\ / : * ? " < >) . . |
| <code>ssl_do_merchant_email</code> | N | Determine if the merchant would like an email when the import file is completed, possible values: T to send an email or F to not send. |
| <code>ssl_merchant_email</code> | N | Merchant's Email Address, if email not specified and <code>ssl_do_merchant_email</code> is set to T then an email will be sent to the terminal email. |
| <code>ssl_result_format</code> | N | Valid values are HTML, XML, and ASCII. When set to ASCII Virtual Merchant will generate a plain text key-value document. Default is HTML if not sent. |
| <code>ssl_receipt_link_method</code> | N | Only valid option is REDG so no receipt is displayed and data redirected to the link defined in the request in the <code>ssl_receipt_link_url</code> or terminal settings. |
| <code>ssl_receipt_link_url</code> | N | Target of the redirect link for the batch import results. Ignored when the <code>ssl_result_format=ASCII</code> and <code>ssl_receipt_link_method = REDG</code> . |

| Input Field Name | Req? | Description |
|------------------|------|---|
| ssl_error_url | N | If present, the error will be redirected to the URL specified including the <code>errorCode</code> , and <code>errorName</code> and <code>errorMessage</code> fields. |

Response

| Output Field Name | Description |
|---------------------------|---|
| ssl_result | Batch import result code: a result of 0 indicates the file was successfully imported. A result of 1 indicates that the file import failed. |
| ssl_result_message | Batch import result message: a result of File upload successful indicates the file was imported successfully. File upload failed indicates the file was not imported successfully. |
| ssl_transaction_type | Transaction type (ccrecimport). |
| ssl_user_id | Converge User ID. |
| ssl_number_trans | Number of transactions imported in the import file. |
| ssl_response_file | Response file name. |
| ssl_recurring_batch_count | Current recurring transaction number in the recurring batch prior to processing the file. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Recurring Batch Import File Format

A Recurring Batch Import file is a properly formatted CSV or XML batch file of recurring and installment transactions. This file must be uploaded with transaction type `ccrecimport`.

The first line in a CSV file is the header, which contains the field names and values as outlined in the table. Each field in the header should be enclosed within double quotes and separated by a comma. Remaining lines in the file are for transaction data. Each transaction must be in a new line. The data in the transaction line must be placed in the same order as its corresponding field value in the header, enclosed within double quote, and separated by a comma.

The XML file contains one single `<txnrecimport>` element and multiple transaction `<txn>` nested elements. Every transaction is a set of XML tags and its corresponding values as shown in the table. The number of `<txn>` corresponds to the number of transactions in the file. All elements are closed in the order that they were opened.

Note: CVV2/CVC/CID values and Track Data should never be stored, therefore cannot be passed to Converge on file.

The file must be formatted as follows:

| Header Field Name | Req? | Length | Value |
|------------------------------------|------|--------|---|
| <code>ssl_card_number</code> | Y | 19 | Credit card number. |
| <code>ssl_exp_date</code> | Y | 4 | In MMY format. |
| <code>ssl_amount</code> | Y | 13 | Must be a decimal, no dollar sign. |
| <code>ssl_transaction_type</code> | Y | 20 | Available transaction types: <ul style="list-style-type: none"> <code>ccaddrecurring</code> (Add Recurring) <code>ccaddinstall</code> (Add Installment) |
| <code>ssl_billing_cycle</code> | Y | | Billing cycle. Valid returned values, all caps and no hyphens: <ul style="list-style-type: none"> DAILY WEEKLY BIWEEKLY SEMIMONTHLY MONTHLY BIMONTHLY QUARTERLY SEMESTER SEMIANNUALLY ANNUALLY SUSPENDED |
| <code>ssl_next_payment_date</code> | Y | 10 | Next payment date. Format MM/DD/YYYY. |

| Header Field Name | Req? | Length | Value |
|--------------------|------|--------|--|
| ssl_bill_on_half | C | 1 | <p>Half of the month indicator for semi-monthly billing cycles. Valid values are 1 and 2:</p> <ul style="list-style-type: none"> • 1 = the 1st and the 15th of the month • 2 = the 15th and the last day of the month |
| ssl_end_of_month | C | 1 | <p>End of month indicator. Valid values Y or N</p> <p>You must indicate if the recurring transaction will be processed on the last day of the month for the monthly, bi-monthly, quarterly, semester, and semi-annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • • 30-Apr • • 30-June • • 30-Sept • • 30-Nov • • 28-Feb (non-leap year) • • 29-Feb (leap year) <p>You also need to provide the end of the month indicator for the annually billing cycles when the start payment date is any of the following dates:</p> <ul style="list-style-type: none"> • • 28-Feb (non-leap year) • • 29-Feb (leap year) |
| ssl_skip_payment | N | 1 | <p>Skip Payment field.</p> <p>Valid values: Y for yes or N for no. Defaulted to N.</p> |
| ssl_description | N | 255 | Transaction description. |
| ssl_invoice_number | N | 25 | Invoice number for MOTO transactions. |
| ssl_customer_code | N | 17 | Customer code for purchasing card transactions. |
| ssl_salestax | N | 10 | Sales tax for purchasing card transactions. |
| ssl_company | N | 50 | Customer's company name. |
| ssl_first_name | N | 20 | Customer's first name. |
| ssl_last_name | N | 30 | Customer's last name. |
| ssl_avs_address | N | 30 | Cardholder's street address. |
| ssl_address2 | N | 30 | Customer's address line two. |
| ssl_city | N | 30 | Customer's city. |
| ssl_state | N | 30 | Customer's state. |
| ssl_avs_zip | N | 9 | Customer's ZIP code used to process AVS. |
| ssl_country | N | 50 | Customer's country. |
| ssl_phone | N | 20 | Customer's phone number. |

| Header Field Name | Req? | Length | Value |
|------------------------|------|--------|--|
| ssl_email | N | 100 | Customer's email address. |
| ssl_ship_to_company | N | 50 | Ship To company name. |
| ssl_ship_to_first_name | N | 20 | Ship To first name. |
| ssl_ship_to_last_name | N | 30 | Ship To last name. |
| ssl_ship_to_address1 | N | 30 | Ship To address line 1. |
| ssl_ship_to_address2 | N | 30 | Ship To address line 2. |
| ssl_ship_to_city | N | 30 | Ship To city. |
| ssl_ship_to_state | N | 30 | Ship To state. |
| ssl_ship_to_zip | N | 10 | Ship To ZIP code. |
| ssl_ship_to_country | N | 50 | Ship To country. |
| ssl_ship_to_phone | N | 20 | Ship To phone number. |
| User Defined Field | N | | User defined, user should be able to pass up to 25 user defined fields, the value passed in the file should match the field name specified in the Terminal Setup Merchant Payment Fields Add New Field. |

Example**Recurring Batch Import File Example CSV**

The following example demonstrates a CSV file. The first line is a header, which contains fields. The fields in the header as well as the data in each line are delimited by commas, and each value that corresponds to the field header is enclosed within double quote. The content of this file can be copied and saved in a text document with a CSV extension for testing with transaction type ccrecimport:

```
"ssl_card_number","ssl_exp_date","ssl_amount","ssl_transaction_type","ssl_billing_cycle","ssl_next_payment_date","ssl_total_installments","ssl_skip_payment","ssl_customer_code","ssl_salestax","ssl_invoice_number","ssl_description","ssl_company","ssl_first_name","ssl_last_name","ssl_avs_address","ssl_address2","ssl_city","ssl_state","ssl_avs_zip","ssl_country","ssl_phone","ssl_email","ssl_ship_to_company","ssl_ship_to_first_name","ssl_ship_to_last_name","ssl_ship_to_address1","ssl_ship_to_address2","ssl_ship_to_city","ssl_ship_to_state","ssl_ship_to_zip","ssl_ship_to_country","ssl_ship_to_phone"
"00*****0000","1212","15.00","CCADDRECURRING","WEEKLY","10/21/2012","","N","C1234","1.00","IN123","weekly Sunday","My Company","Joe","Doe","1234 Main Street","Suite 100","Atlanta","GA","30328","USA","999-999-9999","anyemail@email.com","","","","","","","","","USA","999-999-9999"
"00*****0000","1225","25.00","CCADDINSTALL","MONTHLY","06/21/2012","10","Y","C44545","0.00","IN123","Monthly Mag","","Jane","Doe","1234 Main
```

Recurring Batch Import File Example XML

The following example demonstrates a properly nested XML file with two recurring transactions. All elements are closed in the order that they were opened. The root element `txnrecimport` contains the transaction element, and every transaction element has its own opening and closing `<txn>` tag:

```
<txnrecimport>
  <txn>
    <ssl_card_number>00*****0000</ssl_card_number>
    <ssl_exp_date>1212</ssl_exp_date>
    <ssl_amount>2.00</ssl_amount>
    <ssl_transaction_type>ccaddrecurring</ssl_transaction_type>
    <ssl_customer_code>CC1234</ssl_customer_code>
    <ssl_salestax>0.50</ssl_salestax>
    <ssl_invoice_number>1234</ssl_invoice_number>
    <ssl_description>test recurring</ssl_description>
    <ssl_billing_cycle>MONTHLY</ssl_billing_cycle>
    <ssl_next_payment_date>08/20/2012</ssl_next_payment_date>
    <ssl_skip_payment>N</ssl_skip_payment>
    <ssl_company>MyCompany</ssl_company>
    <ssl_first_name>John</ssl_first_name>
    <ssl_last_name>Doe</ssl_last_name>
    <ssl_avs_address>123 Main</ssl_avs_address>
    <ssl_city>Atlanta</ssl_city>
    <ssl_state>GA</ssl_state>
    <ssl_avs_zip>30123</ssl_avs_zip>
    <ssl_country>USA</ssl_country>
  </txn>
```

(continued)

```
<txn>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_exp_date>1213</ssl_exp_date>
  <ssl_amount>5.00</ssl_amount>
  <ssl_transaction_type>ccaddinstall</ssl_transaction_type>
  <ssl_customer_code>FF1234</ssl_customer_code>
  <ssl_billing_cycle>WEEKLY</ssl_billing_cycle>
  <ssl_next_payment_date>08/20/2012</ssl_next_payment_date>
  <ssl_total_installments>10</ssl_total_installments>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
</txn>
</txnrecimport>
```

(end)

Debit Card Transactions

Debit transactions require integration to a PIN pad that is injected with Elavon keys to retrieve the following information:

- **DUKPT Value:** This is the value returned by the PIN pad device which was used to encrypt the cardholder's 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.

Debit transactions may only be performed in a retail, service or face to face environments. Mail Order/Telephone Order and e-Commerce businesses cannot perform online PIN based debit transactions.

Note: 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.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Debit Purchase/Sale (<code>dbpurchase</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_track_data</code> | Y | 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. |
| <code>ssl_amount</code> | Y | Transaction base amount must be sent on request. This amount does not include the cash back amount or tip amount which must be sent separately. The system will then compute the total to send for authorization. For example: 1.00. |
| <code>ssl_cashback_amount</code> | N | Cash back. The amount of cash back that the customer will receive. Must be a number with two decimal places, if sent. |
| <code>ssl_tip_amount</code> | N | Tip or gratuity amount to be added to the transaction sale amount. Number with two decimal places, can be 0.00 to indicate no tip was provided. Only used in a <i>Service</i> market segment. |
| <code>ssl_server</code> | N | Server ID, this is the clerk, cashier, and waiter or waitress identification number. Only used in a <i>Service</i> market segment. Alphanumeric, Example: Jack. |
| <code>ssl_shift</code> | N | Shift, can refer to or be used to identify time period, course or type of service, Only used in a <i>Service</i> market segment. Alphanumeric, Example: Lunch. |
| <code>ssl_account_type</code> | Y | Account Type (0 = checking, 1 = saving). |
| <code>ssl_dukpt</code> | 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. This value cannot be stored. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. |
| ssl_pin_block | Y | The encrypted PIN block as returned from the PIN pad device. This value cannot be stored. |

Response

| Output Field Name | Description |
|----------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. |
| ssl_result_message | Transaction result message. For example: APPROVAL. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_txn_id | 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. |
| 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_base_amount | Base amount. The amount sent originally on the request. |
| 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_cashback_amount | Cash back. The amount of cash back that the customer will receive. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. Only used in a <i>Service</i> market segment. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. Only used in a <i>Service</i> market segment. |
| ssl_approval_code | Transaction approval code. |
| ssl_reference_number | Transaction reference number. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_server | Server ID submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |

| Output Field Name | Description |
|-------------------|--|
| ssl_shift | Shift information submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_promo_list | The promo list will contain a list of all promo products; the data for each promo product will be nested and embedded between beginning and ending elements <ssl_promo_product> up to 5 promo products. Each promo product <i>ssl_promo_product</i> will contain <i>ssl_promo_code</i> , <i>ssl_promo_code_name</i> , <i>ssl_promo_code_description</i> , and <i>ssl_promo_code_issue_points</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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 must be passed.

Note: The merchant must contact Elavon to make sure that Debit refunds are enabled for the terminal.

Request

| Input Field Names | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Debit Return (dbreturn). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_track_data</code> | Y | 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. |
| <code>ssl_amount</code> | Y | Transaction amount to refund. |
| <code>ssl_account_type</code> | Y | The debit account type (0 = checking, 1 = saving). |
| <code>ssl_dukpt</code> | Y | 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. |
| <code>ssl_pin_block</code> | Y | Encrypted PIN block as returned from the PIN pad device. This value cannot be stored under any circumstances. |
| <code>ssl_key_pointer</code> | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. |
| <code>ssl_original_date</code> | Y | Date of original transaction in MMDDYY format. |
| <code>ssl_original_time</code> | Y | Time of original transaction in HHMMSS format. |
| <code>ssl_reference_number</code> | Y | Transaction reference number is returned in the authorization response message. |

Response

| Output Field Name | Description |
|----------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction, which prevents it from being authorized. |
| ssl_result_message | Transaction result message. For example: APPROVAL. |
| ssl_txn_id | 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_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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 | Transaction reference number. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Debit Inquiry (dbbainquiry)

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

Note: Track II card swipe is required. Manual entry is not allowed.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Debit Inquiry (dbbainquiry). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |

| Input Field Name | Req? | Description |
|------------------|------|---|
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_track_data | Y | 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_account_type | Y | Account Type (0 = checking, 1 = saving). |
| 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. This value cannot be stored. |
| ssl_pin_block | Y | The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. |

Response

| Output Field Name | Description |
|----------------------|--|
| ssl_result | Outcome of a transaction. A response that contains a <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |
| ssl_account_balance | Balance on the card. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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 | Transaction reference number. |
| 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. |

| Output Field Name | Description |
|-------------------|---|
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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, service or face-to-face environments. 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 hand keyed. This transaction reduces the cardholder's limit to buy, and places the transaction into the open batch.

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Food Stamp Purchase (<code>fspurchase</code>). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_track_data | Y | 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. |
| ssl_amount | Y | Transaction Amount, must be number with two decimal places. |

| Input Field Name | Req? | Description |
|------------------|------|---|
| 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. This value cannot be stored. |
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. |
| ssl_pin_block | Y | The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |

Response

| Output Field Name | Description |
|----------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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 | Transaction reference number. |
| 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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 hand keyed. The Reference Number, Date and Time of Original Transaction are recommended to be passed.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Food Stamp return (<code>fsreturn</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_track_data</code> | Y | 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. |
| <code>ssl_amount</code> | Y | Transaction Amount. Must be number with 2 decimal places. |
| <code>ssl_dukpt</code> | Y | 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. |
| <code>ssl_pin_block</code> | Y | Encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |
| <code>ssl_key_pointer</code> | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions, value must be set to T. |
| <code>ssl_original_date</code> | | Date of original transaction in MMDDYY format. |
| <code>ssl_original_time</code> | | Time of original transaction in HHMMSS format. |
| <code>ssl_reference_number</code> | Y | Transaction reference number is returned in the authorization response message. |

Response

| Output Field Name | Description |
|----------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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 | Transaction reference number. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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).

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Food Stamp Inquiry (fsbainquiry). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_track_data | Y | 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. |
| 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. This value cannot be stored. |
| ssl_pin_block | Y | The encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for US Debit transactions. Value must be set to T. |

Response

| Output Field Name | Description |
|---------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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. |
| ssl_account_balance | Remaining balance on the gift card. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |

| Output Field Name | Description |
|----------------------|--|
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_reference_number | The Transaction Reference Number. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Food Stamp Force Purchase (fsforcepurchase)

This is the completion transaction for an EBT Food Stamp Purchase authorization obtained using the phone. This is a hand keyed card only. This transaction requires a 15-digit Voucher Clear Number from the 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.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Food Stamp Force Purchase (fsforcepurchase). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_card_number | Y | Required for hand-keyed transactions. Credit Card Number as it appears on the credit card. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| ssl_amount | Y | Transaction Amount. Must be number with two decimal places. |
| ssl_approval_code | Y | Voucher Clear Approval Code obtained previously using the phone. |
| ssl_voucher_number | Y | The 15-digit Voucher Clear Number from Merchant's EBT Food Stamp sales slip. |

Response

| Output Field Name | Description |
|----------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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_txn_id | 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Food Stamp Force Return (fsforcereturn)

This message format is used to hand key 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.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Food Stamp Force Return (fsforcereturn). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_card_number | Y | Required for hand-keyed transactions. Credit Card Number as it appears on the credit card. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| ssl_amount | Y | Transaction amount. Must be number with two decimal places. |
| ssl_approval_code | Y | The voucher clear approval code obtained previously using the phone. |
| ssl_voucher_number | Y | The 15-digit voucher clear number from the Merchant's EBT Food Stamp sales slip. |

Response

| 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 | Transaction result message. Example: APPROVAL. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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. |

| Output Field Name | Description |
|--------------------|---|
| ssl_txn_id | 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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 these transactions. This message format is for the EBT Cash Benefit Purchase transaction for either a magnetic stripe read (Track II) or hand keyed card.

Note: You must pass one of the following fields: `ssl_track_data` or `ssl_card_number`.

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Cash Benefit Purchase (cbpurchase). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_track_data | Y | Required for swiped transactions. 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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data. |

| Input Field Name | Req? | Description |
|---------------------|------|---|
| ssl_card_number | Y | Required for hand-keyed transactions. Credit Card Number as it appears on the credit card. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| ssl_amount | Y | Transaction total Amount including surcharge or cash back. Must be a number with two decimal places. |
| ssl_cashback_amount | N | Cash back. The amount of cash back that the customer will receive. Must be a number with two decimal places, if sent. |
| ssl_dukpt | Y | 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. |
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. |
| ssl_pin_block | Y | Encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |

Response

| Output Field Name | Description |
|----------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_base_amount | Base amount, the amount sent originally on the request. |
| ssl_card_number | 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 | Transaction reference number. |
| ssl_surcharge_amount | Surcharge amount as configured by merchant. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |

| Output Field Name | Description |
|-------------------|--|
| 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Cash Benefit Inquiry (cbbainquiry)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Cash Benefit Inquiry (cbbainquiry). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_track_data | Y | Required for swiped transactions. 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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data. |
| ssl_dukpt | Y | 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. |
| ssl_key_pointer | Y | Triple-DES DUKPT pointer that indicates to Elavon which encryption key was used for EBT transactions. Value must be set to T. |
| ssl_pin_block | Y | Encrypted PIN Block as returned from the PIN pad device. This value cannot be stored. |

Response

| Output Field Name | Description |
|----------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | Remaining balance on the gift card. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_reference_number | Transaction reference number. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Transactions

This message format is for either a whole track, track 1 or track 2 magnetic stripe read or hand keyed gift card transactions (EGC) available for all supported market segments.

Notes:

- Track data in the `ssl_track_data` for swiped transactions.
- The encrypted track data in the `ssl_encrypted_track1_data` and/or `ssl_encrypted_track2_data` fields extracted from an encrypting device. Refer to the [Encryption](#) section for more information.
- The card number in the `ssl_card_number` for hand-keyed transactions.
- The token in the `ssl_token` from a previously tokenized card number.

Gift Card Activation (egcactivation)

Gift cards must be activated prior to use.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Gift Card Activation (egcactivation). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_show_form</code> | N | Set value to true to show the Converge Payment Form (Available only for <code>process.do</code>), set to false otherwise. |
| <code>ssl_track_data</code> | Y | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information.</p> |

| Input Field Name | Req? | Description |
|---------------------|------|--|
| ssl_card_number | Y | Required for hand-keyed transactions. Gift card number as it appears on the gift card. |
| ssl_token | C | Required for hand-keyed transaction if gift card number is not sent. Use only with a terminal that is setup with <i>Tokenization</i> . Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Gift card expiry date as it appears on gift card. |
| ssl_amount | Y | Transaction sale amount, number with two decimal places. For example: 1.00. |
| 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: <ul style="list-style-type: none"> • 0 = cash • 1 = credit • 2 = debit • 3 = check |
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |

Response

| Output Field Name | Description |
|---------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | Remaining balance on the gift card. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |

| Output Field Name | Description |
|---------------------|---|
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | 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: <ul style="list-style-type: none"> • Cash • Credit • Debit • Check |
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_token | Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Sale/Redemption (egcsale)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|--------------------------------------|
| ssl_transaction_type | Y | Gift Card Sale/Redemption (egcsale). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge payment form (available only for <code>process.do</code>), set to false otherwise. |
| ssl_track_data | Y | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | Y | <p>Required for hand-keyed transactions.</p> <p>Gift card number as it appears on the gift card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if gift card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | Y | <p>Required for hand-keyed transactions.</p> <p>Gift card expiry date as it appears on gift card.</p> |
| ssl_amount | Y | Transaction sale amount, number with 2 decimal places. This amount does not include the tip amount which must be sent separately if needed. For example: 1.00. |
| ssl_tip_amount | N | Tip or gratuity amount to be added to the transaction sale amount. Number with 2 decimal places, can be 0.00 to indicate no tip was provided. Only used in a <i>Service</i> market segment. |
| ssl_server | N | <p>Server ID, this is the clerk, cashier, and waiter or waitress identification number. Only used in a <i>Service</i> market segment.</p> <p>Alphanumeric, Example: Jack.</p> |
| ssl_shift | N | <p>Shift, can refer to or be used to identify time period, course or type of service, Only used in a <i>Service</i> market segment.</p> <p>Alphanumeric, Example: Lunch.</p> |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_get_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N.</p> <p>Once generated, the token number can be stored and used as a substitute for a card number at later time.</p> |

Response

| Output Field Name | Description |
|---------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | Remaining balance on the gift card. |
| ssl_amount | The total transaction amount. Returned based on merchant setup. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. Only used in a <i>Service</i> market segment. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. Only used in a <i>Service</i> market segment. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_server | Server Id submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_shift | Shift information submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_token | <p>Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number.</p> <p>Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i>.</p> |

| Output Field Name | Description |
|--------------------|---|
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift 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.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Gift Card Refund (egccardrefund). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | N | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge payment form (available only for <code>process.do</code>). Set to false otherwise. |
| ssl_track_data | Y | Required for swiped transactions. 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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data. Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code> . Refer to the Encryption section for more information. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_card_number | Y | Required for hand-keyed transactions. Gift Card Number as it appears on the gift card. |
| ssl_token | C | Required for hand-keyed transaction if gift card number is not sent. Use only with a terminal that is setup with <i>Tokenization</i> . Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Gift Card Expiry date as it appears on gift card. |
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |

Response

| Output Field Name | Description |
|---------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | Remaining balance on the gift card. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_token | Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |

| Output Field Name | Description |
|--------------------|---|
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Replenishment/Reload (egcreload)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Gift Card Replenishment/Reload (egcreload). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge payment form (Available only for <code>process.do</code>), set to false otherwise. |
| ssl_track_data | Y | Required for swiped transactions. 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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data. |
| ssl_card_number | Y | Required for hand-keyed transactions. Gift card number as it appears on the gift card. |

| Input Field Name | Req? | Description |
|---------------------|------|--|
| ssl_token | C | Required for hand-keyed transaction if gift card number is not sent. Use only with a terminal that is setup with <i>Tokenization</i> . Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number. |
| ssl_exp_date | Y | Required for hand-keyed transactions. Gift card expiry date as it appears on gift card. |
| ssl_amount | Y | Transaction sale amount, number with two decimal places. For example: 1.00. |
| 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: <ul style="list-style-type: none"> • 0 = Cash • 1 = Credit • 2 = Debit • 3 = Check |
| ssl_get_token | N | Use only with a terminal that is setup with <i>Tokenization</i> . Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N. Once generated, the token number can be stored and used as a substitute for a card number at later time. |

Response

| Output Field Name | Description |
|---------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_account_balance | Remaining balance on the gift card. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |

| Output Field Name | Description |
|---------------------|---|
| 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: <ul style="list-style-type: none"> • Cash • Credit • Debit • Check |
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_token | Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Balance Inquiry (egcbalinquiry)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Gift Card Balance Inquiry (egcbalinquiry). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_show_form | N | Set value to true to show the Converge Payment Form (Available only for <code>process.do</code>), set to false otherwise. |
| ssl_track_data | Y | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | Y | <p>Required for hand-keyed transactions.</p> <p>Gift card number as it appears on the gift card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if gift card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | Y | <p>Required for hand-keyed transactions.</p> <p>Gift card expiry date as it appears on gift card.</p> |
| ssl_get_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N.</p> <p>Once generated, the token number can be stored and used as a substitute for a card number at later time.</p> |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |

| Output Field Name | Description |
|---------------------|---|
| 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_account_balance | Remaining balance on the gift card. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_token | Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Credit (egccredit)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Gift Card Credit (egccredit). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (Available only for <code>process.do</code>), set to false otherwise. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_track_data | Y | <p>Required for swiped transactions.</p> <p>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. Expiration date is included with a track data. First and last name of the Cardholder is also included with the Track I data.</p> <p>Note: If using an encrypting device, pass the encrypted track data in the <code>ssl_encrypted_track1_data</code> and/or <code>ssl_encrypted_track2_data</code>. Refer to the Encryption section for more information</p> |
| ssl_card_number | Y | <p>Required for hand-keyed transactions.</p> <p>Gift Card Number as it appears on the gift card.</p> |
| ssl_token | C | <p>Required for hand-keyed transaction if gift card number is not sent.</p> <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Gift Card Token, previously generated from a gift card number. A token can be stored and used as a substitute for a card number.</p> |
| ssl_exp_date | Y | <p>Required for hand-keyed transactions.</p> <p>Gift Card Expiry date as it appears on gift card.</p> |
| ssl_amount | Y | <p>Transaction Sale Amount, number with 2 decimal places. For example: 1.00.</p> |
| ssl_get_token | N | <p>Use only with a terminal that is setup with <i>Tokenization</i>.</p> <p>Generate Token indicator, used to indicate if you wish to generate a token after processing the installment. Valid value: Y (generate a token), N (do not generate token). Defaulted to N.</p> <p>Once generated, the token number can be stored and used as a substitute for a card number at later time.</p> |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | Transaction identification number. This is a unique number used to identify the transaction. |

| Output Field Name | Description |
|---------------------|---|
| 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_account_balance | Remaining balance on the gift card. |
| ssl_amount | Transaction amount. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_email | Returned based on merchant setup. |
| ssl_exp_date | Returned based on merchant setup. |
| ssl_token | Gift Card Token generated from the card number. A token can be stored and used as a substitute for a card number. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| ssl_token_response | Outcome of the token generation. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. Returned only if generating a token is requested in a terminal that setup for <i>Tokenization</i> . |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Gift Card Generate Token (egcgettoken)

The egcgettoken is a transaction that generates a token from a gift card number. The token generated can be used in place of a gift card number in any subsequent transactions. This transaction type is supported only when a terminal is setup for tokenization; refer to the [Tokenization](#) section for more information.

To perform an egcgettoken, you must submit either:

- Card number and expiration date for hand keyed transaction
- Track Data for swiped transaction

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Gift Card Generate Token (egcgettoken). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured within Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment Form (available only for process.do), set to false otherwise. |
| ssl_track_data | C | Gift Card Number as it appears on the gift card. |
| ssl_card_number | C | Gift Card Number as it appears on the gift card. |
| ssl_exp_date | C | Gift Card Expiry Date as it appears on gift card formatted as MMY. |

Electronic Check Transactions

Electronic Checks Service (ECS) supports two types of check processing: check present ECS Paper Check and check not present ACH ECheck.

To process ECS Paper Check, a check reader device is required to electronically read and capture the MICR check data and image.

To process an ACH ECheck, you must pass the hand keyed check information. Refer to the [Electronic Check ACH ECheck](#) section for more information and best practices.

Electronic Check Purchase (ecspurchase)

The `ecspurchase` is a transaction in which money is debited from a checking account using a check. Data is either captured manually (ACH ECheck : WEB, TEL, PPD, CCD) or from a paper check (ECS Paper Conversion: POP, BOC and ARC) using a check reader device.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|--|
| <code>ssl_transaction_type</code> | Y | Check Sale (<code>ecspurchase</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_micr_data</code> | Y | <p>Required only for ECS Paper Check conversion.</p> <p>MICR number as read through the check reader.</p> <p>The unformatted Magnetic Ink Character Recognition (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):</p> <ul style="list-style-type: none"> 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. |

| Input Field Name | Req? | Description |
|----------------------------|------|---|
| ssl_check_image | Y | Required only for ECS Paper Check conversion. Check Image Data base 64 TIFF image. Failure to properly format the image in a BASE64 encode message will result in an error when attempting to post the message. |
| ssl_aba_number | C | Required only for ACH ECheck. Routing/Transit Number as printed on the Check. |
| ssl_bank_account_number | C | Required only for ACH ECheck. Bank account number as printed on the Check. Not applicable for Paper Check conversion. |
| ssl_check_number | N | Optional for ACH ECheck. (Not applicable Paper Check conversion.) The check Number as printed on the Check. Check number is rarely used for ACH ECheck. |
| ssl_bank_account_type | C | Required only for ACH ECheck. The bank account type. One numeric digit value, valid values are: 0 for Personal, 1 for Business. |
| ssl_agree | C | Required only for ACH ECheck. The agreement flag. One numeric digit value, valid values are: 1 for I agree, 0 for I do not Agree. |
| ssl_amount | Y | Transaction Sale Amount. Must be number with 2 decimal places. This amount does not include the tip amount which must be sent separately if needed. |
| ssl_ecs_product_code | N | Optional for ACH ECheck. Not applicable for Paper Check conversion. The ACH type. 3 alpha value, valid values are: WEB, TEL, PPD, sent only when account type is personal to indicate the type of ACH transaction to process. This value is automatically set to CCD when the check is business. |
| ssl_first_name | C | Required for ACH ECheck with Personal Checks. The Consumer first name. Alphanumeric. |
| ssl_last_name | C | Required for ACH ECheck with Personal Checks. The Consumer last name. |
| ssl_company | C | Required for ACH ECheck with Business Checks. Company Name. |
| ssl_drivers_license_number | C | Required with ECS Paper Check – Guarantee. Driver's License number as entered by the user. Alphanumeric. |

| Input Field Name | Req? | Description |
|----------------------------------|------|--|
| ssl_drivers_license_phone_number | C | Required with ECS Paper Check – Guarantee. Customers 10 digit Phone number including the area code. |
| ssl_drivers_license_state | C | Required with ECS Paper Check – Guarantee. State Code |
| ssl_tip_amount | N | Only used in a <i>Service</i> market segment. Tip or gratuity amount to be added to the transaction sale amount. Number with 2 decimal places, can be 0.00 to indicate no tip was provided. |
| ssl_server | N | Only used in a <i>Service</i> market segment. Server ID, this is the clerk, cashier, and waiter or waitress identification number. Alphanumeric, Example: Jack. |
| ssl_shift | N | Only used in a <i>Service</i> market segment. Shift, can refer to or used to identify time period, course or type of service. Alphanumeric, Example: Lunch. |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_txn_id | 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_amount | The total transaction amount. Returned based on merchant setup. |
| ssl_base_amount | Only used in a <i>Service</i> market segment. Base amount. Transaction amount sent originally on the request. Returned based on merchant setup. |

| Output Field Name | Description |
|----------------------------------|--|
| ssl_tip_amount | Only used in a <i>Service</i> market segment. Tip or gratuity amount that was added or updated. Returned based on merchant setup. |
| ssl_approval_code | Transaction approval code. |
| ssl_reference_number | The reference number. |
| ssl_bank_account_number | Bank account number. |
| ssl_check_number | The check number. |
| 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_aba_number | Routing/Transit Number from the parsed MICR data. |
| ssl_server | Server Id submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_shift | Shift information submitted with the request. Only returned on <i>Service</i> market segment based on the merchant setup. |
| ssl_email | Returned based on merchant setup. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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. To perform an `ecsvoid` you must submit the transaction ID submitted in the original transaction.

Note: The `ssl_show_form` property does not apply on Void transactions.

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Check Void (<code>ecsvoid</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_txn_id</code> | Y | Unique identifier returned on the original transaction. |

Response

| Output Field Name | Description |
|---------------------------------|--|
| <code>ssl_result</code> | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction. |
| <code>ssl_result_message</code> | Transaction result message. Example: APPROVAL. |
| <code>ssl_txn_id</code> | Transaction identification number. This is a unique number used to identify the transaction. |
| <code>ssl_approval_code</code> | Transaction approval code. |
| <code>ssl_txn_time</code> | 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. |
| <code>errorCode</code> | Error code returned <i>only</i> 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. |
| <code>errorMessage</code> | Detailed explanation of the error returned <i>only</i> 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. |
| <code>errorName</code> | Error name or reason for the error returned <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

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 (p1dpurchase)

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

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | PINLess Debit Purchase (p1dpurchase). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_show_form | N | Set value to true to show the Converge Payment form (Available only for <code>process.do</code>), set to false otherwise. |
| ssl_account_type | Y | Account Type (0=checking, 1=saving). |
| ssl_amount | Y | Transaction base amount must be sent on request. This amount does not include the surcharge amount. Decimal, for example: 1.00. |
| ssl_card_number | Y | Debit Card Number as it appears on the debit card. |
| ssl_exp_date | N | Debit Card Expiry date as it appears on debit card. |
| ssl_customer_number | Y | This value is used to submit the Customer Account Number or Payee account number for PINLess Debit transactions. |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |
| ssl_amount | The total transaction amount including surcharge amount if any. |
| ssl_account_type | Account Type (checking = 0, saving = 1). Required. |
| ssl_approval_code | Transaction approval code. |
| ssl_base_amount | The base transaction amount. |

| Output Field Name | Description |
|----------------------|--|
| ssl_card_number | Masked debit card number passed on the original request. |
| ssl_customer_number | The customer number. |
| ssl_surcharge_amount | The surcharge amount as configured by merchant. |
| ssl_txn_id | 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. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Cash Tender Transactions

This message format is used to enter cash transactions. When processing Cash transactions you may add description, billing information, shipping information, tip and custom fields if applicable. Once processed, the Cash transaction will show under the Current Batches in the Cash Batch. Once closed, all cash transactions will show under Settle Batches Cash.

All Cash transactions will auto close at the end the day during auto settlement for those terminals set for auto settle. For those terminals set for manually settlement, they are automatically closed at 3:00 AM EST. A transaction type settle can be used to settle those types of transactions.

Notes:

- The `ssl_show_form` property does not apply on cash transactions.
- Cash transactions are not allowed for the *e-Commerce* market segment.

Cash Sale (cashsale)

The cashsale transaction is used to enter a purchase by cash.

Request

| Input Field Name | Req? | Description |
|----------------------|------|--|
| ssl_transaction_type | Y | Cash Sale (cashsale). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_amount | Y | Transaction Sale Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. The authorized amount passed on request should not contain the tip amount, tips must be passed separately. The total authorized amount will be calculated during the authorization if tip is provided. For example: 1.00. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |
| ssl_tip_amount | N | Use only in a <i>Service</i> market segment. Tip or gratuity amount to be added, must be 2 decimal places, can be 0.00 to reset or remove the original tip from a transaction. Example: 1.00. |
| ssl_server | N | Use only in a <i>Service</i> market segment. Server ID, this is the clerk, cashier, and waiter or waitress identification number. Alphanumeric, Example: Jack. |
| ssl_shift | N | Use only in a <i>Service</i> market segment. Shift, can refer to or be used to identify time period, course or type of service. Alphanumeric, Example: Lunch. |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an approved transaction. A response containing any other value for <code>ssl_result</code> represents a declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: APPROVAL. |

| Output Field Name | Description |
|-------------------|--|
| ssl_txn_id | 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_amount | The total transaction authorized or approved amount. This amount will include tip if tip has been provided in the request. Returned based on merchant setup. |
| ssl_base_amount | Base amount. Transaction amount sent originally on the request. Returned based on merchant setup, base amount includes tax. Returned in the <i>Service</i> market segment. |
| ssl_tip_amount | Tip or gratuity amount that was added or updated. Returned based on merchant setup. Returned in the <i>Service</i> market segment. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

Here's an XML example request for a \$3 sale:

Request

```

<txn>
  <ssl_merchant_id>My_Merchant_id</ssl_merchant_id>
  <ssl_user_id>My_User</ssl_user_id>
  <ssl_pin>XXXXXX</ssl_pin>
  <ssl_transaction_type>CASHSALE</ssl_transaction_type>
  <ssl_amount>3.00</ssl_amount>
  <ssl_salestax>0.21</ssl_salestax>
  <customField>test12</customField>
  <anotherCustomField>testAnother12</anotherCustomField>
  <ssl_show_form>>false</ssl_show_form>
</txn>

```

Response

```

<txn>
  <customField>test12</customField>
  <anotherCustomField>testAnother12</anotherCustomField>
  <ssl_txn_time>12/18/2013 09:03:36 AM</ssl_txn_time>
  <ssl_txn_id>AA776F-3CEC01A7-B70A-4584-AC8C-5040C0297354</ssl_txn_id>
  <ssl_amount>3.00</ssl_amount>
  <ssl_result>0</ssl_result><ssl_salestax>0.21</ssl_salestax>
  <anothercustom>testAnother12</anothercustom>
  <ssl_result_message>APPROVAL</ssl_result_message>
</txn>
    
```

Cash Return/Credit (cashcredit)

This transaction is used to enter a refund by cash. This refund is not tied to any previous purchases.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Cash Return or Cash Credit (cashcredit). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_amount | Y | Transaction Amount to be refunded. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |
| ssl_salestax | N | Recommended for purchasing cards. Sales tax amount applied to this transaction in decimal. Tax exempt transactions can pass 0.00 to properly reflect a tax exempt transaction. |

Response

| 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 | Transaction result message. Example: APPROVAL. |

| Output Field Name | Description |
|-------------------|--|
| ssl_txn_id | 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_amount | The total transaction authorized or approved amount. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Example

The following is an XML example request for a \$5.00 refund:

Request

```

<txn>
  <ssl_merchant_id>My_Merchant_id</ssl_merchant_id>
  <ssl_user_id>My_User</ssl_user_id>
  <ssl_pin>XXXXXX</ssl_pin>
  <ssl_transaction_type>CASHCREDIT</ssl_transaction_type>
  <ssl_amount>5.00</ssl_amount>
  <ssl_salestax>0.23</ssl_salestax>
  <customField>test12</customField>
  <anotherCustomField>testAnother12</anotherCustomField>
  <ssl_show_form>>false</ssl_show_form>
</txn>

```

Response

```

<txn>
  <customField>test12</customField>
  <anotherCustomField>testAnother12</anotherCustomField>
  <ssl_txn_time>12/18/2013 09:13:24 AM</ssl_txn_time>
  <ssl_txn_id>AA786F-4CEC21A7-B70A-4584-AC8C-
  5040C0297354</ssl_txn_id>
  <ssl_amount>5.00</ssl_amount>
  <ssl_result>0</ssl_result>
  <ssl_salestax>0.23</ssl_salestax>
  <anothercustom>testAnother12</anothercustom>
  <ssl_result_message>APPROVAL</ssl_result_message>
</txn>

```

Card Manager Transactions

This message format is for card manager transactions. The card manager stores information associated to a token generated from a credit card.

Notes:

- Terminal must be setup for tokenization.
- Card Manager transactions are applicable to Credit Card tokens only. Gift Card tokens can be stored in the Card Manager using the user interface only. Gift tokens cannot be added, modified, or deleted using Card Manager transaction types.

Token Query (ccquerytoken)

This request is used to retrieve the information associated to a token. Token must be present in the card manager.

Note: The `ssl_show_form` property does not apply

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Token Query (ccquerytoken). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |

| Input Field Name | Req? | Description |
|------------------|------|---|
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_token | Y | Use only with a terminal that is setup with <i>Tokenization</i> . Credit Card Token, previously generated from a card number and stored in the card manager. A token can be stored and used as a substitute for a card number. |

Response

| Output Field Name | Description |
|-------------------|---|
| ssl_token | Credit Card Token supplied in the authorization. |
| ssl_card_type | The card type, example: CREDITCARD, DEBITCARD. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Expiration date. Returned based on merchant setup. |
| ssl_company | Company name. Returned based on merchant setup. |
| ssl_customer_id | Customer ID. Returned based on merchant setup. |
| ssl_first_name | Customer's first name. Returned based on merchant setup. |
| ssl_last_name | Customer's last name. Returned based on merchant setup. |
| ssl_avs_address | Customer's address. Returned based on merchant setup. |
| ssl_address2 | Customer's address line 2. Returned based on merchant setup. |
| ssl_avs_zip | Customer's ZIP code. Returned based on merchant setup. |
| ssl_city | Customer's city. Returned based on merchant setup. |
| ssl_state | Customer's state. Returned based on merchant setup. |
| ssl_country | Customer's country. Returned based on merchant setup. |
| ssl_phone | Customer's phone number. Returned based on merchant setup. |
| ssl_email | Customer's email address. Returned based on merchant setup. |
| ssl_description | Transaction description. Returned based on merchant setup. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Token Update (ccupdatetoken)

This request is used to update the information associated to a token. Token must be present in the card manager.

Notes:

- Only the information you wish to update along with the token should be sent
- If card number is sent, a new token will be generated
- Some entries in the card manager can be set to null by sending empty values, first and last names cannot be sent null as they are required
- You can verify the account before update, if card is declined the token will not be updated
- The `ssl_show_form` property does not apply

Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Token Update (ccupdatetoken). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_token</code> | Y | Use only with a terminal that is setup with <i>Tokenization</i> . Credit Card Token, previously generated from a card number and stored in the card manager. A token can be stored and used as a substitute for a card number. |
| <code>ssl_card_number</code> | N | Credit Card Number as it appears on the credit card. |
| <code>ssl_exp_date</code> | N | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| <code>ssl_company</code> | N | Company name. |
| <code>ssl_customer_id</code> | N | Customer identifier as set by the merchant. |
| <code>ssl_first_name</code> | N | Cardholder first name. |
| <code>ssl_last_name</code> | N | Cardholder last name. |
| <code>ssl_avs_zip</code> | N | Cardholder ZIP code. |
| <code>ssl_avs_address</code> | N | Cardholder Address. |
| <code>ssl_address2</code> | N | Customer's address line 2. |
| <code>ssl_city</code> | N | Customer's city. |
| <code>ssl_state</code> | N | Customer's state. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_country | N | Customer's country. |
| ssl_email | N | Customer's email address. |
| ssl_phone | N | Customer's phone number. |
| ssl_description | N | ssl_description. |
| ssl_verify | N | Account Verify indicator to indicate that account verification is needed prior to updating a token. Valid values: Y, N. |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Generate Token Result code. A result of 0 indicates that the token was generated; 1 indicates that the token was not generated. Returned only when account verification is requested. |
| ssl_result_message | The account verification transaction result message returned only when ccgettoken is requested with Account verification flag. Example: APPROVAL, DECLINE. Refer to the Authorization Response Codes section for an extensive list of possible returned messages. Returned only when account verification is requested. |
| ssl_token | Credit Card Token supplied in the authorization. |
| ssl_token_response | Outcome of the token update. This value will be SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. |
| ssl_txn_id | Transaction identification number, returned only when account verification is requested. This is a unique number used to identify the transaction. |
| ssl_txn_time | Date and time when the transaction was processed, returned only when account verification is requested. Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM. |
| ssl_approval_code | Transaction approval code. Returned only when account verification is requested. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). Returned only when account verification is requested. |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). Returned only when account verification is requested. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_exp_date | Expiration date. Returned based on merchant setup. |
| ssl_company | Company name. Returned based on merchant setup. |
| ssl_customer_id | Customer ID. Returned based on merchant setup. |
| ssl_first_name | Customer's first name. Returned based on merchant setup. |

| Output Field Name | Description |
|-------------------|---|
| ssl_last_name | Customer's last name. Returned based on merchant setup. |
| ssl_avs_address | Customer's address. Returned based on merchant setup. |
| ssl_address2 | Customer's address line 2. Returned based on merchant setup. |
| ssl_avs_zip | Customer's ZIP code. Returned based on merchant setup. |
| ssl_city | Customer's city. Returned based on merchant setup. |
| ssl_state | Customer's state. Returned based on merchant setup. |
| ssl_country | Customer's country. Returned based on merchant setup. |
| ssl_phone | Customer's phone number. Returned based on merchant setup. |
| ssl_email | Customer's email address. Returned based on merchant setup. |
| ssl_description | Transaction description. Returned based on merchant setup. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Token Delete (ccdeletetoken)

This request is used to delete a token from the card manager; all the information associated with that token will be removed.

Note: The `ssl_show_form` property does not apply.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Token Delete (ccdeletetoken). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_token | Y | Use only with a terminal that is setup with <i>Tokenization</i> . Credit Card Token, previously generated from a card number and stored in the card manager. A token can be stored and used as a substitute for a card number. |

| Input Field Name | Req? | Description |
|------------------|------|--|
| ssl_card_number | N | Credit Card Number as it appears on the credit card. |
| ssl_exp_date | N | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| ssl_company | N | Company name. |
| ssl_customer_id | N | Customer identifier as set by the merchant. |
| ssl_first_name | N | Cardholder first name. |
| ssl_last_name | N | Cardholder last name. |
| ssl_avs_zip | N | Cardholder ZIP code. |
| ssl_avs_address | N | Cardholder Address. |
| ssl_address2 | N | Customer's address line 2. |
| ssl_city | N | Customer's city. |
| ssl_state | N | Customer's state. |
| ssl_country | N | Customer's country |
| ssl_email | N | Customer's email address |
| ssl_phone | N | Customer's phone number |
| ssl_description | N | ssl_description |

Response

| Output Field Name | Description |
|--------------------|--|
| ssl_token | The token value deleted from the card manager. |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. |
| ssl_result | Outcome of the transaction if the token generation was requested with verification or with an authorization. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. Refer to the Authorization Response Codes section for an extensive list of possible returned messages. |
| 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. |

End of Day Transactions

This message format is for End of Day transactions. Those transactions consist of searching for a transaction, resending email, or settling transactions.

Transaction Query (txnquery)

This request is used to search for a transaction based on a date range or card data or combination of both. One of the following fields must be provided:

- Card information providing one of the following fields:
 - ssl_card_number
 - ssl_track_data
 - ssl_card_suffix

Or/ And

- Date range ssl_search_start_date and ssl_search_end_date

Or

- Transaction identifier ssl_txn_id

The result will always include the number of transactions in the ssl_txn_count field and the detailed response of each transaction when more than a single transaction is returned. The detail response will contain the original response along with a transaction status in the ssl_trans_status field to indicate if the transaction status (opened, settled, pending, and so on). All types of transactions for all payment types, approved or declined within the provided date range will be returned and will be listed in descending date/time order.

Notes:

- Search is allowed using one single card information or a date range or combination of both. Passing multiple card information data in a single request is not allowed.
- When searching based on a date range or date range with other criteria:
 - Date range cannot exceed 31 days at the time in a single request. For example you can run a query to run transactions from 01/01/2014 to 01/02/2014
 - One date at minimum must be provided.
 - If you submit a start date without an end date, the end date will be set to start date plus 31 days (not the current date). If you submit an end date without a start date, the start date will be set to end date minus 31 days.

- When searching based on a card number, track data or a card suffix, the search will return a list of all matching transactions within the last two months from the Open and Settled batches.
- When searching based on combination of card information and date range, the search will return a list of all matching transactions within the specified date range from the Open and Settled batches.
- When searching based on a transaction identifier, a single match is always returned. Date range is not as the transaction identifier is always unique.
- All transactions are listed in descending date/time order.
- The `ssl_show_form` property does not apply.

Request

| Input Field Name | Req? | Description |
|------------------------------------|-------------|---|
| <code>ssl_transaction_type</code> | Y | Transaction Query (<code>txnquery</code>). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_txn_id</code> | C | Unique identifier returned on the original transaction. |
| <code>ssl_card_number</code> | C | Credit Card Number as it appears on the credit card. |
| <code>ssl_track_data</code> | C | The raw track I and/or II data from the magnetic strip on the card. |
| <code>ssl_card_suffix</code> | C | The last 4 digits of the card number. |
| <code>ssl_search_start_date</code> | C | Start date of the search window. Format MM/DD/YYYY. |
| <code>ssl_search_end_date</code> | C | End date of the search window. Format MM/DD/YYYY. |

Response

| Output Field Name | Description |
|---------------------------------|---|
| <code>ssl_txn_count</code> | The number of transactions matching the search query. |
| <code>ssl_result_message</code> | Transaction result message. Example: APPROVAL, PARTIAL APPROVAL. Refer to the Authorization Response Codes section for an extensive list of possible returned messages. |
| <code>ssl_txn_id</code> | Transaction identification number. This is a unique number used to identify the transaction. This value can be used to complete the Auth Only transaction. |
| <code>ssl_user_id</code> | Converge User ID that processed the transaction. |
| <code>ssl_trans_status</code> | Transaction status, example: STL (Settled). |

| Output Field Name | Description |
|----------------------|---|
| ssl_card_type | The card type, example: CREDITCARD, DEBITCARD. |
| ssl_transaction_type | Transaction type, example: SALE. |
| 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_first_name | Cardholder's first name. |
| ssl_last_name | Cardholder's last name. |
| ssl_card_number | Masked card number. Returned based on merchant setup. |
| ssl_entry_mode | The entry mode of the transaction, K for keyed or S for swiped. |
| ssl_avs_response | The Address Verification Response Code (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes). |
| ssl_approval_code | Transaction approval code. |
| ssl_amount | Transaction authorized or approved amount. Returned based on merchant setup. |
| ssl_invoice_number | The invoice or ticket number sent originally on the request. Returned based on merchant setup. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Examples

Query Based on Transaction ID Request

```

xmldata=<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>txnquery</s
sl_transaction_type><ssl_txn_id>AA48439-65E7D601-5E31-4809-882A-
2028CBC3A979</ssl_txn_id></txn>

```

Query Based on Card Number Request

```
xmldata=<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>txnquery</s
sl_transaction_type><ssl_card_number>[Full Card
number]</ssl_card_number></txn>
```

Query Based on Track data Request

```
xmldata=<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>
txnquery</ssl_transaction_type><ssl_track_data>[Track data]
</ssl_track_data></txn>
```

Query Based on Last 4 digit of a Card Number Request

```
xmldata=<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>txnquery</s
sl_transaction_type><ssl_card_suffix>3003</ssl_card_suffix></txn>
```

Query Response

This is an XML response of a query based on a transaction identifier:

```
<txn>
  <ssl_txn_id>AA7757-2C7226AA-C2A2-45E4-BB5F-
  1EC12309D9C3</ssl_txn_id>
  <ssl_user_id>my_user</ssl_user_id>
  <ssl_trans_status>STL</ssl_trans_status>
  <ssl_card_type>CREDITCARD</ssl_card_type>
  <ssl_transaction_type>SALE</ssl_transaction_type>
  <ssl_txn_time>07/09/2013 02:29:13 PM</ssl_txn_time>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_entry_mode>K</ssl_entry_mode>
  <ssl_avs_response />
  <ssl_cvv2_response />
  <ssl_amount>.37</ssl_amount>
  <ssl_invoice_number />
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_approval_code>N29032</ssl_approval_code>
</txn>
```

This is an ASCII response of a query based on a transaction identifier:

```
ssl_txn_id=AA7757-2C7226AA-C2A2-45E4-BB5F-1EC12309D9C3
ssl_user_id=my_user
ssl_trans_status=STL
ssl_card_type=CREDITCARD
ssl_transaction_type=SALE
ssl_txn_time=07/09/2013 02:29:13 PM
ssl_first_name=John
ssl_last_name=Doe
ssl_card_number=00*****0000
ssl_exp_date=1215
ssl_entry_mode=K
ssl_avs_response=
ssl_cvv2_response=
ssl_amount=.37
ssl_invoice_number=
ssl_result_message=APPROVAL
ssl_approval_code=N29032
```

This is an XML response of a query based on a card number. In this example 2 transactions were returned:

```
<txnlist>
<ssl_txn_count>2</ssl_txn_count>
<txn>
<ssl_txn_id>AA7757-EBB7E0B1-CA25-45FF-9728-
5D45BC222D9A</ssl_txn_id>
<ssl_user_id>my_user</ssl_user_id>
<ssl_trans_status>OPN</ssl_trans_status>
<ssl_card_type>CREDITCARD</ssl_card_type>
<ssl_transaction_type>AVSonly</ssl_transaction_type>
<ssl_txn_time>06/04/2013 10:12:49 AM</ssl_txn_time>
<ssl_first_name />
<ssl_last_name />
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_entry_mode>K</ssl_entry_mode>
<ssl_avs_response>G</ssl_avs_response>
<ssl_cv2_response />
<ssl_amount>0</ssl_amount>
<ssl_invoice_number />
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_approval_code>031719</ssl_approval_code>
</txn>
<txn>
<ssl_txn_id>AA7757-05E853A6-8DB6-4A0B-B907-
C5E96829CAC6</ssl_txn_id>
<ssl_user_id>my_user</ssl_user_id>
<ssl_trans_status>OPN</ssl_trans_status>
<ssl_card_type>CREDITCARD</ssl_card_type>
<ssl_transaction_type>AVSonly</ssl_transaction_type>
<ssl_txn_time>06/04/2013 10:12:28 AM</ssl_txn_time>
<ssl_first_name />
<ssl_last_name />
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_entry_mode>K</ssl_entry_mode>
<ssl_avs_response>G</ssl_avs_response>
<ssl_cv2_response />
<ssl amount>0</ssl amount>
```

This is an ASCII response of a query based on a card number. In this example two transactions were returned:

```
ssl_txn_count=2
ssl_txn_id=AA7757-EBB7E0B1-CA25-45FF-9728-5D45BC222D9A
ssl_user_id=my_user
ssl_trans_status=OPN
ssl_card_type=CREDITCARD
ssl_transaction_type=AVSonly
ssl_txn_time=06/04/2013 10:12:49 AM
ssl_first_name= Jane
ssl_last_name= Doe
ssl_card_number=00*****0000
ssl_exp_date=1215
ssl_entry_mode=K
ssl_avs_response=G
ssl_cvv2_response= M
ssl_amount=0.00
ssl_invoice_number=
ssl_result_message=APPROVAL
ssl_approval_code=031719
ssl_txn_id=AA7757-05E853A6-8DB6-4A0B-B907-C5E96829CAC6
ssl_user_id=my_user
ssl_trans_status=OPN
ssl_card_type=CREDITCARD
ssl_transaction_type=AVSonly
ssl_txn_time=06/04/2013 10:12:28 AM
ssl_first_name=John
ssl_last_name=Doe
ssl_card_number=00*****0000
ssl_exp_date=1215
ssl_entry_mode=K
ssl_avs_response=G
ssl_cvv2_response=
ssl_amount=0
ssl_invoice_number=
ssl_result_message=APPROVAL
ssl_approval_code=031719
```

Transaction Email (txnemail)

The `txnemail` is used to initiate or resend an email to the merchant or customer for a specific transaction. If the original transaction contains a customer email, both merchant and customer emails will be sent. However, specific flags can be sent to control if the email should be sent to only the customer or only to the merchant. Optionally, the request can include a new customer email address if different from the original transaction or if the original transaction did not have an email address. A flag can be also sent to update the email address on file with the new one sent with the email request. The terminal must be setup to receive email.

Notes:

- The `ssl_show_form` property does not apply for these transactions.
- An error is sent if the terminal is not setup for emails.

Request

| Input Field Name | Req? | Description |
|------------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Transaction Email (txnemail). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_txn_id</code> | Y | Unique identifier returned on the original transaction. |
| <code>ssl_email</code> | N | Customer's email address. Send this value if the existing transaction did not have a customer email or if the email needs to be sent to a different email address. If value not sent the system will attempt to send an email to the email address stored from the original transaction |
| <code>ssl_update_email</code> | N | Email update flag indicated if email needs to be updated after the email is resent. The new customer email must be sent in the <code>ssl_email</code> field. Valid values: Y or N. Default is N. |
| <code>ssl_do_customer_email</code> | N | Indicates if email needs to be sent to the customer, if not passed and customer email is present in the original transaction, it will be defaulted to True. Valid values are: <ul style="list-style-type: none"> • T = True • F = False |
| <code>ssl_do_merchant_email</code> | N | Indicates if email needs to be sent to the merchant, if not passed defaulted to True. Valid values are: <ul style="list-style-type: none"> • T = True • F = False |

Response

| Output Field Name | Description |
|--------------------|---|
| ssl_result | Outcome of a transaction. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> represents a Declined transaction preventing it from being authorized. |
| ssl_result_message | Transaction result message. Example: SUCCESS. |
| ssl_txn_id | 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 A.M. |
| errorCode | Error code returned <i>only</i> 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 <i>only</i> 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 <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |

Settle (settle)

This request is used to initiate a manual settlement on a single transaction or on a batch of transactions. Once the batch is successfully settled, the funds will be moved from the customer's account to the merchant account.

Open batches must be reviewed prior to settlement for accuracy, transactions set to pend or to review must be released prior to settlement. You may opt to set the terminal for auto-settlement or submit a manual settlement from the integrated application. Elavon highly recommends that batches be closed out on a daily basis.

Batch submissions are posted using `process.do` for key value pairs formatted request or `processxml.do` for XML formatted request.

To settle a single credit card transaction, the settlement request must include the transaction identification number in the `ssl_txn_id` field. If this value is omitted, the entire batch is settled.

Notes:

- When settling transactions, the `ssl_show_form` property does not apply.
- Card number or track data should not be sent

- Users must have the Batches-Settle Transactions user right in order to settle transactions
- The Scheduled for Settlement response indicates that the batch will be submitted for settlement. In order to check the status of a transaction, the integrated application may use the transaction query to obtain information on the status of a transaction
- The settlement can be initiated on open transactions only. Transactions that are pended or set to review must be released or set to unpend in order to settle.
- Settling a single gift or cash transaction is not supported, the gift or cash batch must be settled entirely.
- 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.
- Manual settlement is not allowed for those terminals setup for *Multi-Currency*.

Request

| Input Field Name | Req? | Description |
|-------------------------|-------------|---|
| ssl_transaction_type | Y | Settle transaction (settle). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_txn_id | N | Unique identifier returned on the original transaction. |

Response

| Output Field Name | Description |
|--------------------------|--|
| ssl_result | Outcome of a transaction. A response that contains ssl_result of 0 represents a successful batch submission transaction. A response containing any other value for ssl_result represents a failure preventing it from being settled, example: Empty batch, settlement in progress. |
| ssl_result_message | Transaction result message. Example: Scheduled for settlement, Empty batch, and settlement in progress. |
| ssl_txn_time | Date and time when the batch was submitted, Format: MM/DD/YYYY hh:mm:ss PM/AM. Example: 03/18/2010 10:34:10 AM. |
| ssl_txn_id | Unique identifier returned on the original transaction, this is to settle an individual transaction sent in the request. |
| ssl_txn_main_count | Total Transaction count in Main Batch to be settled. |
| ssl_txn_main_amount | Total Transaction amount in Main Batch to be settled. |
| ssl_txn_cash_count | Total cash count to be settled. |

| Output Field Name | Description |
|---------------------|---|
| ssl_txn_cash_amount | Total cash amount to be settled. |
| ssl_txn_ecg_count | Total gift card count to be settled. |
| ssl_txn_ecg_amount | Total gift card amount to be settled. |
| errorCode | Error code returned <i>only</i> 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 | Error name or reason for the error returned <i>only</i> if an error occurred. Refer to the Error Codes section for more information. |
| errorMessage | Detailed explanation of the error returned <i>only</i> 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. |

Examples

Example 1: process.do

The following XML code example demonstrates the initiation of a settlement transaction request and response for batch and single transaction settlement.

Batch Settlement Request

```
xmldata=<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>settle
</ssl_transaction_type></txn>
```

Transaction Settlement Request

```
xmldata=<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>settle
</ssl_transaction_type><ssl_txn_id>AA48439-65E7D601-5E31-4809-882A-
2028CBC3A979</ssl_txn_id></txn>
```

Batch Settlement Response

This is a successful attempt for settlement of a batch:

```
<txn>
<ssl_result>0</ssl_result>
<ssl_result_message>Scheduled for Settlement</ssl_result_message>
<ssl_txn_id/>
<ssl_txn_main_count>10</ssl_txn_main_count>
<ssl_txn_main_amount>220.00</ssl_txn_main_amount>
<ssl_txn_ecg_count>2</ssl_txn_ecg_count>
<ssl_txn_ecg_amount>50.00</ssl_txn_ecg_amount>
</txn>
```

This is a request of a settlement on a terminal that is not active (Not live):

```
<txn>
<ssl_result>0</ssl_result>
<ssl_result_message>Scheduled for Settlement</ssl_result_message>
<ssl_txn_id/>
<ssl_txn_main_count>999</ssl_txn_main_count>
<ssl_txn_main_amount>0.00</ssl_txn_main_amount>
<ssl_txn_ecg_count>999</ssl_txn_ecg_count>
<ssl_txn_ecg_amount>0.00</ssl_txn_ecg_amount>
</txn>
```

Transaction Settlement Response

This is a successful attempt for settlement of a single transaction:

```
<txn>
<ssl_result>0</ssl_result>
<ssl_result_message>Scheduled for Settlement</ssl_result_message>
<ssl_txn_id>AA48439-65E7D601-5E31-4809-882A-
2028CBC3A979</ssl_txn_id>
<ssl_txn_main_count>1</ssl_txn_main_count>
<ssl_txn_main_amount>16.00</ssl_txn_main_amount>
<ssl_txn_ecg_count>0</ssl_txn_ecg_count>
<ssl_txn_ecg_amount/>
</txn>
```

This is a successful attempt for settlement for a single transaction on a terminal that is not active (Not live):

```
<txn>
  <ssl_result>0</ssl_result>
  <ssl_result_message>Scheduled for Settlement</ssl_result_message>
  <ssl_txn_id>00000000-0000-0000-0000-000000000000</ssl_txn_id>
  <ssl_txn_main_count>1</ssl_txn_main_count>
  <ssl_txn_main_amount>0.00</ssl_txn_main_amount>
  <ssl_txn_ecg_count>0</ssl_txn_ecg_count>
  <ssl_txn_ecg_amount>0.00</ssl_txn_ecg_amount>
</txn>
```

Account Admin Transactions

This message format is for Admin transactions. Those types of transactions use XML and send the transaction to `accountxml.do` to retrieve the terminal account information.

Terminal Setup (terminalsetup)

This request is used to retrieve the terminal and merchant setup information located under **Terminal | Merchant | Main** option and **Terminal | Merchant | Terminal** option in the user interface terminal menu. The response contains the `<MerchantInformation>` element which has the merchant information and the `<TerminalInformation>` element which has the terminal information.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Terminal Setup Information (terminalsetup). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |

Response

| Output Field Name | Description |
|---------------------|--|
| MerchantInformation | Contains the <i>Merchant Information Elements</i> as described in the table below. |
| TerminalInformation | Contains the <i>Terminal Information Elements</i> as described table below. |

Merchant Information Elements

| Output Field Name | Description |
|-------------------|---|
| MerchantName | Merchant's information as setup in Elavon |
| Address1 | Merchant's Street Address |
| Address2 | Merchant's Street Address Line 2 |
| City | Merchant's City |
| StateProvince | Merchant's State or Province |
| PostalCode | Merchant's Zip Code |
| ContactName | Merchant's contact name |
| ContactPhone1 | Merchant's phone number |
| ContactPhone2 | Merchant's alternative phone number |
| ContactEmail | Merchant's Email address |
| TerminalEmail | Terminal Email address |
| SMSEmail | Merchant's Email |
| MerchantURL | Merchant's URL Address |

Terminal Information Elements

| Output Field Name | Description |
|--------------------|--|
| vm_friendly_name | Merchant's information as setup in Elavon |
| vm_region | Merchant's Street Address |
| vm_time_zone | Merchant's Street Address Line 2 |
| vm_currency | Merchant's City |
| vm_market_segment | Merchant's State or Province |
| vm_status | The terminal Status. Example: ACTIVE, NOT LIVE, SUSPENDED An active terminal can process transactions. |
| vm_processing_type | Terminal based only. |
| vm_payment_types | Contains the <i>payment type elements</i> and values as outlined in the payment type elements table below. |

| Output Field Name | Description | | | | | | | | | | |
|------------------------|--|-------------------|-------------|------------------------|---|------------------|---|-------------------|--------------------------------------|-----------|---|
| vm_credit_option | Contains the <i>credit option elements</i> and <i>values</i> as outlined in the credit option elements table below. | | | | | | | | | | |
| vm_recurring_option | Contains the <i>default recurring option elements</i> and <i>values</i> as outlined below. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Frequency</td> <td>Billing cycle, example: MONTHLY</td> </tr> <tr> <td>Payments</td> <td>Payment number, example: 12</td> </tr> </tbody> </table> | Output Field Name | Description | Frequency | Billing cycle, example: MONTHLY | Payments | Payment number, example: 12 | | | | |
| Output Field Name | Description | | | | | | | | | | |
| Frequency | Billing cycle, example: MONTHLY | | | | | | | | | | |
| Payments | Payment number, example: 12 | | | | | | | | | | |
| vm_debit_option | Contains the <i>debit option elements</i> and <i>values</i> as outlined below. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Surcharge</td> <td>If surcharge is setup for this terminal, Valid values: Y, N</td> </tr> <tr> <td>SurchargeAmount</td> <td>Surcharge amount. Example: 1.00</td> </tr> </tbody> </table> | Output Field Name | Description | Surcharge | If surcharge is setup for this terminal, Valid values: Y, N | SurchargeAmount | Surcharge amount. Example: 1.00 | | | | |
| Output Field Name | Description | | | | | | | | | | |
| Surcharge | If surcharge is setup for this terminal, Valid values: Y, N | | | | | | | | | | |
| SurchargeAmount | Surcharge amount. Example: 1.00 | | | | | | | | | | |
| vm_echeck_option | Contains the <i>check option elements</i> and <i>values</i> as outlined below. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>CheckType</td> <td>Check type, valid values: ACHECHECK, PAPERCHECK</td> </tr> <tr> <td>TransactionType</td> <td>Check transaction option, example: VERIFICATION</td> </tr> <tr> <td>ImageUploadOption</td> <td>Image-upload option for Paper check.</td> </tr> <tr> <td>Recurring</td> <td>If ACH recurring is setup for this terminal, valid values: Y, N</td> </tr> </tbody> </table> | Output Field Name | Description | CheckType | Check type, valid values: ACHECHECK, PAPERCHECK | TransactionType | Check transaction option, example: VERIFICATION | ImageUploadOption | Image-upload option for Paper check. | Recurring | If ACH recurring is setup for this terminal, valid values: Y, N |
| Output Field Name | Description | | | | | | | | | | |
| CheckType | Check type, valid values: ACHECHECK, PAPERCHECK | | | | | | | | | | |
| TransactionType | Check transaction option, example: VERIFICATION | | | | | | | | | | |
| ImageUploadOption | Image-upload option for Paper check. | | | | | | | | | | |
| Recurring | If ACH recurring is setup for this terminal, valid values: Y, N | | | | | | | | | | |
| vm_transaction_entry | Contains the <i>API option elements</i> and <i>values</i> as outlined below. <table border="1" style="width: 100%; margin-top: 5px;"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>EnableHTTPSTransaction</td> <td>If API is enabled. Y, N</td> </tr> <tr> <td>EnableHTTPSBatch</td> <td>If batch import API is enabled. Y, N</td> </tr> <tr> <td>EnableVMM</td> <td>If VM Mobile is enabled. Y, N</td> </tr> </tbody> </table> | Output Field Name | Description | EnableHTTPSTransaction | If API is enabled. Y, N | EnableHTTPSBatch | If batch import API is enabled. Y, N | EnableVMM | If VM Mobile is enabled. Y, N | | |
| Output Field Name | Description | | | | | | | | | | |
| EnableHTTPSTransaction | If API is enabled. Y, N | | | | | | | | | | |
| EnableHTTPSBatch | If batch import API is enabled. Y, N | | | | | | | | | | |
| EnableVMM | If VM Mobile is enabled. Y, N | | | | | | | | | | |
| vm_tokenization | Tokenization flag to indicate if the terminal is setup for tokenization. Valid values are Y for Yes or N for No. | | | | | | | | | | |

Payment type Elements

| Output Field Name | Description |
|-------------------|---|
| AVS | Indicates if Address Verification is setup for this terminal. Valid values: Y or N |
| CVN | Indicates if Card Verification is setup for this terminal. Valid values: Y or N |
| PurchaseCard | Indicates if Purchase Card is setup for this terminal. Valid values: Y or N |
| InvoiceNumber | Indicates if Invoice Number is setup for this terminal. Valid values: Y or N |
| DCC | Indicates if Dynamic Currency is setup for this terminal. Valid values: Y or N |
| MCC | Indicates if <i>Multi-Currency</i> is setup for this terminal. Valid values: Y or N |

| Output Field Name | Description |
|-------------------|---|
| Recurring | Indicates if Recurring is setup for this terminal. Valid values: Y or N |
| Last4digits | Indicates if Last 4 Digits is setup for this terminal. Valid values: Y or N |
| TravelData | Indicates if Travel Data is setup for this terminal. Valid values: Y or N |
| AccountUpdater | Indicates if Account Updater is setup for this terminal. Valid values: Y or N |

Credit Option Elements

| Output Field Name | Description |
|-------------------|--|
| Credit | Indicates if credit is setup for this terminal. Valid values: Y or N |
| Debit | Indicates if debit is setup for this terminal. Valid values: Y or N |
| Gift | Indicates if gift is setup for this terminal. Valid values: Y or N |
| ECheck | Indicates if ECheck is setup for this terminal. Valid values: Y or N |
| FoodStamp | Indicates if Food Stamp is setup for this terminal. Valid values: Y or N |
| CashBenefit | Indicates if Cash Benefit is setup for this terminal. Valid values: Y or N |
| Cash | Indicates if Cash is setup for this terminal. Valid values: Y or N |

Examples

The following XML code example demonstrates a request to retrieve the terminal setup using `accountxml.do`.

Terminal Setup Information Request

```
xmldata=<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_transaction_type>terminalsetup</ssl_transaction_type></txn>
```

Terminal Setup Information Response

```
<txn>
  <MerchantInformation>
    <MerchantName>ABC COMPANY</MerchantName>
    <Address1>123 MAIN STREET</Address1><Address2/>
    <City>ANY CITY</City>
    <StateProvince>GA</StateProvince>
    <PostalCode>999999</PostalCode>
    <ContactName/><ContactPhone1/><ContactPhone2/>
    <ContactEmail>email@elavon.com</ContactEmail>
    <TerminalEmail>email@elavon.com</TerminalEmail>
    <MerchantURL/>
  </MerchantInformation>
  <TerminalInformation>
    <vm_friendly_name>MY STORE</vm_friendly_name>
    <vm_region>USA</vm_region>
    <vm_time_zone>EST</vm_time_zone>
    <vm_currency>USD</vm_currency>
    <vm_market_segment>Retail</vm_market_segment>
    <vm_status>ACTIVE</vm_status>
    <vm_processing_type>TerminalBased</vm_processing_type>
    <vm_payment_types>
      <Credit>Y</Credit>
      <Debit>Y</Debit>
      <Gift>Y</Gift>
      <ECheck>Y</ECheck>
      <FoodStamp>Y</FoodStamp>
      <CashBenefit>Y</CashBenefit>
      <Cash>Y</Cash>
    </vm payment types>
```

(continued)

```
<vm_credit_option>
  <AVS>N</AVS><CVN>Y</CVN>
  <PurchaseCard>Y</PurchaseCard>
  <InvoiceNumber>Y</InvoiceNumber>
  <DCC>N</DCC><MCC>N</MCC>
  <Recurring>Y</Recurring>
  <Last4digits>N</Last4digits>
  <TravelData>N</TravelData>
  <AccountUpdater>Y</AccountUpdater>
</vm_credit_option>
<vm_level3>N</vm_level3>
<vm_recurring_option>
  <Frequency>MONTHLY</Frequency><Payments>12</Payments>
</vm_recurring_option>
<vm_debit_option>
  <Cashback>Y</Cashback>
  <Surcharge>Y</Surcharge>
  <SurchargeAmount>1.00</SurchargeAmount>
</vm_debit_option>
<vm_echeck_option>
  <CheckType>ACHECHECK</CheckType>
  <TransactionType>VERIFICATION</TransactionType>
  <Recurring>Y</Recurring>
</vm_echeck_option>
<vm_transaction_entry>
  <EnableHTTPSTransaction>Y</EnableHTTPSTransaction>
  <EnableHTTPSBatch>Y</EnableHTTPSBatch>
  <EnableVMM>Y</EnableVMM>
</vm_transaction_entry>
<vm_tokenization>Y</vm_tokenization>
</TerminalInformation>
</txn>
```

(end)

Payment Field (fieldsetup)

This request is used to retrieve the payment field information as setup under the **Terminal | Merchant | Payment Fields** option in the user interface terminal menu. The response contains the root xml beginning and ending element `<txn>`; within that element, section information is presented. Data for each section will be nested and embedded between beginning element `<Section>`, and ending element `</Section>`, there are several section elements.

Within each Section element, fields are presented. Data for each field resides between beginning `<Field>` and ending `</Field>`, there are several field elements.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Terminal Setup Information (fieldsetup). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |

Response

| Output Field Name | Description |
|-------------------|---|
| Name | Section name as configured by Converge. Example: OrderSection |
| DisplayName | Section label name, as configured by the merchant. Example: Order Section |
| ShowinVT | Indicates if the section will show in the Virtual Terminal under the payment menu. Valid values: Y, N |
| ShowOnPaymentForm | Indicates if the section will show in the Payment page under the payment menu. Valid values: Y, N |
| ShowinReceipt | Indicates if the section will show in the receipt. Valid values: Y, N |
| CustomerEmail | Indicates if the section will show in the customer email. Valid values: Y, N |
| MerchantEmail | Indicates if the section will show in the merchant email. Valid values: Y, N |

| Output Field Name | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|-------------------|-------------|------|------------|-------------|--------------------|---------|---------------------------------------|-----------|-------------------------------|-----|----------------------|-----|----------------------|----------|---|----------|---|----------------------|---|-------------------|---|---------------|--|---------------|---|---------------|---|-------------------|---|------------------|--|-------------|--|
| Field | <p>Contains the <i>field data elements</i> and <i>values</i> as outlined below.</p> <table border="1" data-bbox="591 327 1414 1467"> <thead> <tr> <th data-bbox="591 327 943 357">Output Field Name</th> <th data-bbox="943 327 1414 357">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="591 357 943 407">Name</td> <td data-bbox="943 357 1414 407">Field name</td> </tr> <tr> <td data-bbox="591 407 943 457">DisplayName</td> <td data-bbox="943 407 1414 457">Field display name</td> </tr> <tr> <td data-bbox="591 457 943 508">Section</td> <td data-bbox="943 457 1414 508">Which section the field is located in</td> </tr> <tr> <td data-bbox="591 508 943 558">FieldType</td> <td data-bbox="943 508 1414 558">Field type, for example: TEXT</td> </tr> <tr> <td data-bbox="591 558 943 609">Min</td> <td data-bbox="943 558 1414 609">Minimum allowed size</td> </tr> <tr> <td data-bbox="591 609 943 659">Max</td> <td data-bbox="943 609 1414 659">Maximum allowed size</td> </tr> <tr> <td data-bbox="591 659 943 730">Required</td> <td data-bbox="943 659 1414 730">Indicates if the field is set to required. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 730 943 802">ShowinVT</td> <td data-bbox="943 730 1414 802">Indicates if the field is set to show in the Virtual Terminal. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 802 943 911">ChangedOnPaymentForm</td> <td data-bbox="943 802 1414 911">Indicates if the field can be changed in the payment form in the Virtual Terminal. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 911 943 991">ShowOnPaymentForm</td> <td data-bbox="943 911 1414 991">Indicates if the field is set to show in the payment page. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 991 943 1071">ShowinReceipt</td> <td data-bbox="943 991 1414 1071">Indicates if the field is set to show in the receipt. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 1071 943 1150">CustomerEmail</td> <td data-bbox="943 1071 1414 1150">Indicates if the field is set to show in the customer email. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 1150 943 1230">MerchantEmail</td> <td data-bbox="943 1150 1414 1230">Indicates if the field is set to show in the merchant email. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 1230 943 1310">ForwardOnApproval</td> <td data-bbox="943 1230 1414 1310">Indicates if the field is set to be forwarded on approval. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 1310 943 1390">ForwardOnDecline</td> <td data-bbox="943 1310 1414 1390">Indicates if the field is set to be forwarded on decline. Valid values: Y, N</td> </tr> <tr> <td data-bbox="591 1390 943 1467">SystemField</td> <td data-bbox="943 1390 1414 1467">Indicates if the field is a system or custom field. Valid values: Y, N</td> </tr> </tbody> </table> | Output Field Name | Description | Name | Field name | DisplayName | Field display name | Section | Which section the field is located in | FieldType | Field type, for example: TEXT | Min | Minimum allowed size | Max | Maximum allowed size | Required | Indicates if the field is set to required. Valid values: Y, N | ShowinVT | Indicates if the field is set to show in the Virtual Terminal. Valid values: Y, N | ChangedOnPaymentForm | Indicates if the field can be changed in the payment form in the Virtual Terminal. Valid values: Y, N | ShowOnPaymentForm | Indicates if the field is set to show in the payment page. Valid values: Y, N | ShowinReceipt | Indicates if the field is set to show in the receipt. Valid values: Y, N | CustomerEmail | Indicates if the field is set to show in the customer email. Valid values: Y, N | MerchantEmail | Indicates if the field is set to show in the merchant email. Valid values: Y, N | ForwardOnApproval | Indicates if the field is set to be forwarded on approval. Valid values: Y, N | ForwardOnDecline | Indicates if the field is set to be forwarded on decline. Valid values: Y, N | SystemField | Indicates if the field is a system or custom field. Valid values: Y, N |
| Output Field Name | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Name | Field name | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DisplayName | Field display name | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Section | Which section the field is located in | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FieldType | Field type, for example: TEXT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Min | Minimum allowed size | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Max | Maximum allowed size | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Required | Indicates if the field is set to required. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ShowinVT | Indicates if the field is set to show in the Virtual Terminal. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ChangedOnPaymentForm | Indicates if the field can be changed in the payment form in the Virtual Terminal. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ShowOnPaymentForm | Indicates if the field is set to show in the payment page. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ShowinReceipt | Indicates if the field is set to show in the receipt. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CustomerEmail | Indicates if the field is set to show in the customer email. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MerchantEmail | Indicates if the field is set to show in the merchant email. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ForwardOnApproval | Indicates if the field is set to be forwarded on approval. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ForwardOnDecline | Indicates if the field is set to be forwarded on decline. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SystemField | Indicates if the field is a system or custom field. Valid values: Y, N | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Examples

The following XML code example demonstrates a request to retrieve the payment field setup for a terminal using `accountxml.do`.

Payment Field Setup Information Request

```
xmldata=<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_transaction_type>fieldsetup</ssl_transaction_type></txn>
```

Payment Field Setup Response

```
<txn>
  <Section>
    <Name>OrderSection</Name>
    <DisplayName>Order Section</DisplayName>
    <ShowinVT>Y</ShowinVT>
    <ShowOnPaymentForm>Y</ShowOnPaymentForm>
    <ShowinReceipt>Y</ShowinReceipt>
    <CustomerEmail>Y</CustomerEmail>
    <MerchantEmail>Y</MerchantEmail>
    <Field>
      <Name>ssl_account_data</Name>
      <DisplayName>Order Section</DisplayName>
      <Section>OrderSection</Section>
      <FieldType>TEXT</FieldType>
      <Min>0</Min>
      <Max>19</Max>
      <Required>Y</Required>
      <ShowinVT>Y</ShowinVT>
      <ChangedOnPaymentForm>N</ChangedOnPaymentForm>
      <ShowOnPaymentForm>Y</ShowOnPaymentForm>
      <ShowinReceipt>Y</ShowinReceipt>
      <CustomerEmail>Y</CustomerEmail>
      <MerchantEmail>Y</MerchantEmail>
      <ForwardOnApproval>Y</ForwardOnApproval>
      <ForwardOnDecline>Y</ForwardOnDecline>
      <SystemField>Y</SystemField>
    </Field>
```

(continued)

```
<Section>
.....
</Section>
<Section>
<Name>CustomFields</Name>
<DisplayName>Custom Fields</DisplayName>
<ShowinVT>Y</ShowinVT>
<ShowOnPaymentForm>Y</ShowOnPaymentForm>
<ShowinReceipt>Y</ShowinReceipt>
<CustomerEmail>Y</CustomerEmail>
<MerchantEmail>Y</MerchantEmail>
<Field>
<Name>MyCustom</Name>
<DisplayName>Custom Fields</DisplayName>
<Section>CustomFields</Section>
<FieldType>TEXT</FieldType>
<Min>5</Min>
<Max>12</Max>
<Required>Y</Required>
<ShowinVT>Y</ShowinVT>
<ChangedOnPaymentForm>Y</ChangedOnPaymentForm>
<ShowOnPaymentForm>Y</ShowOnPaymentForm>
<ShowinReceipt>Y</ShowinReceipt>
<CustomerEmail>Y</CustomerEmail>
<MerchantEmail>Y</MerchantEmail>
<ForwardOnApproval>N</ForwardOnApproval>
<ForwardOnDecline>N</ForwardOnDecline>
<SystemField>N</SystemField>
</Field>
</Section>
</txn>
```

(end)

Printer Setup (printersetup)

This request is used to retrieve the printer information as setup under the **Terminal | Merchant | Printer** option in the user interface. When sending the request you must use XML and send the transaction to `accountxml.do`. Response contains the root xml beginning and ending element `<txn>`.

Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Terminal Setup Information (printersetup). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured on Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |

Response

| Output Field Name | Description | | | | | | | | | | | | | | |
|----------------------|--|-------------------|-------------|---------------|--|-------------|--|-------------|---|-------------|--|-------------|----------------------------|-------------|---------------------------|
| PrinterOptions | Indicates the printer options as described in the table below: <table border="1" data-bbox="633 966 1412 1102"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>PrinterType</td> <td>Type of printer, example: Serial Printer.</td> </tr> <tr> <td>PaperType</td> <td>Type of paper, example: Multi Ply.</td> </tr> </tbody> </table> | Output Field Name | Description | PrinterType | Type of printer, example: Serial Printer. | PaperType | Type of paper, example: Multi Ply. | | | | | | | | |
| Output Field Name | Description | | | | | | | | | | | | | | |
| PrinterType | Type of printer, example: Serial Printer. | | | | | | | | | | | | | | |
| PaperType | Type of paper, example: Multi Ply. | | | | | | | | | | | | | | |
| ReceiptHeaderOptions | Indicates the header of the receipt as described in the table below: <table border="1" data-bbox="633 1144 1412 1585"> <thead> <tr> <th>Output Field Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Justification</td> <td>The location of the header in the receipt. For example: Center</td> </tr> <tr> <td>HeaderLine1</td> <td>First line of the header, for example: Thank you for shopping.</td> </tr> <tr> <td>HeaderLine2</td> <td>Second line of the header, for example: Store ABC</td> </tr> <tr> <td>HeaderLine3</td> <td>Third line of the header, for example: 123 Main Street</td> </tr> <tr> <td>HeaderLine4</td> <td>Fourth line of the header.</td> </tr> <tr> <td>HeaderLine5</td> <td>Fifth line of the header.</td> </tr> </tbody> </table> | Output Field Name | Description | Justification | The location of the header in the receipt. For example: Center | HeaderLine1 | First line of the header, for example: Thank you for shopping. | HeaderLine2 | Second line of the header, for example: Store ABC | HeaderLine3 | Third line of the header, for example: 123 Main Street | HeaderLine4 | Fourth line of the header. | HeaderLine5 | Fifth line of the header. |
| Output Field Name | Description | | | | | | | | | | | | | | |
| Justification | The location of the header in the receipt. For example: Center | | | | | | | | | | | | | | |
| HeaderLine1 | First line of the header, for example: Thank you for shopping. | | | | | | | | | | | | | | |
| HeaderLine2 | Second line of the header, for example: Store ABC | | | | | | | | | | | | | | |
| HeaderLine3 | Third line of the header, for example: 123 Main Street | | | | | | | | | | | | | | |
| HeaderLine4 | Fourth line of the header. | | | | | | | | | | | | | | |
| HeaderLine5 | Fifth line of the header. | | | | | | | | | | | | | | |

| Output Field Name | Description | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------|--|-------------------|-------------|---------------|--|--------------|---|--------------|---|--------------|---|--------------|--|--------------|--|--------------|---|--------------|---|--------------|--|--------------|--|
| ReceiptTrailerOptions | <p>Indicates the trailer/footer of the receipt as described in the table below:</p> <table border="1"> <thead> <tr> <th data-bbox="651 327 878 357">Output Field Name</th> <th data-bbox="951 327 1081 357">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="651 369 789 399">Justification</td> <td data-bbox="951 369 1373 432">The location of the trailer in the receipt. For example: Center</td> </tr> <tr> <td data-bbox="651 453 789 483">TrailerLine1</td> <td data-bbox="951 453 1390 516">First line of the trailer, for example: Come back and shop with us again.</td> </tr> <tr> <td data-bbox="651 537 789 567">TrailerLine2</td> <td data-bbox="951 537 1357 600">Second line of the trailer, for example: Store ABC</td> </tr> <tr> <td data-bbox="651 621 789 651">TrailerLine3</td> <td data-bbox="951 621 1373 684">Third line of the trailer, for example: 123 Main Street</td> </tr> <tr> <td data-bbox="651 705 789 735">TrailerLine4</td> <td data-bbox="951 705 1211 735">Fourth line of the trailer.</td> </tr> <tr> <td data-bbox="651 747 789 777">TrailerLine5</td> <td data-bbox="951 747 1195 777">Fifth line of the trailer.</td> </tr> </tbody> </table> | Output Field Name | Description | Justification | The location of the trailer in the receipt. For example: Center | TrailerLine1 | First line of the trailer, for example: Come back and shop with us again. | TrailerLine2 | Second line of the trailer, for example: Store ABC | TrailerLine3 | Third line of the trailer, for example: 123 Main Street | TrailerLine4 | Fourth line of the trailer. | TrailerLine5 | Fifth line of the trailer. | | | | | | | | |
| Output Field Name | Description | | | | | | | | | | | | | | | | | | | | | | |
| Justification | The location of the trailer in the receipt. For example: Center | | | | | | | | | | | | | | | | | | | | | | |
| TrailerLine1 | First line of the trailer, for example: Come back and shop with us again. | | | | | | | | | | | | | | | | | | | | | | |
| TrailerLine2 | Second line of the trailer, for example: Store ABC | | | | | | | | | | | | | | | | | | | | | | |
| TrailerLine3 | Third line of the trailer, for example: 123 Main Street | | | | | | | | | | | | | | | | | | | | | | |
| TrailerLine4 | Fourth line of the trailer. | | | | | | | | | | | | | | | | | | | | | | |
| TrailerLine5 | Fifth line of the trailer. | | | | | | | | | | | | | | | | | | | | | | |
| ReceiptCustomFields | <p>Indicates the custom fields setup to show in the receipt.</p> <table border="1"> <thead> <tr> <th data-bbox="651 821 878 850">Output Field Name</th> <th data-bbox="919 821 1057 850">Description</th> </tr> </thead> <tbody> <tr> <td data-bbox="651 863 813 892">CustomLabel1</td> <td data-bbox="919 863 1390 926">First custom field label setup to show in the receipt. Example: My Custom Data</td> </tr> <tr> <td data-bbox="651 947 813 976">CustomField1</td> <td data-bbox="919 947 1325 1010">First custom field setup to show in the receipt. Example: MyField</td> </tr> <tr> <td data-bbox="651 1031 813 1060">CustomLabel2</td> <td data-bbox="919 1031 1373 1094">Second custom field label setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1115 813 1144">CustomField2</td> <td data-bbox="919 1115 1357 1178">Second custom field setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1199 813 1228">CustomLabel3</td> <td data-bbox="919 1199 1390 1262">Third custom field label setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1283 813 1312">CustomField3</td> <td data-bbox="919 1283 1341 1346">Third custom field setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1367 813 1396">CustomLabel4</td> <td data-bbox="919 1367 1373 1430">Fourth custom field label setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1451 813 1480">CustomField4</td> <td data-bbox="919 1451 1357 1514">Fourth custom field setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1535 813 1564">CustomLabel5</td> <td data-bbox="919 1535 1390 1598">Fifth custom field label setup to show in the receipt.</td> </tr> <tr> <td data-bbox="651 1619 813 1648">CustomField5</td> <td data-bbox="919 1619 1325 1682">Fifth custom field setup to show in the receipt.</td> </tr> </tbody> </table> | Output Field Name | Description | CustomLabel1 | First custom field label setup to show in the receipt. Example: My Custom Data | CustomField1 | First custom field setup to show in the receipt. Example: MyField | CustomLabel2 | Second custom field label setup to show in the receipt. | CustomField2 | Second custom field setup to show in the receipt. | CustomLabel3 | Third custom field label setup to show in the receipt. | CustomField3 | Third custom field setup to show in the receipt. | CustomLabel4 | Fourth custom field label setup to show in the receipt. | CustomField4 | Fourth custom field setup to show in the receipt. | CustomLabel5 | Fifth custom field label setup to show in the receipt. | CustomField5 | Fifth custom field setup to show in the receipt. |
| Output Field Name | Description | | | | | | | | | | | | | | | | | | | | | | |
| CustomLabel1 | First custom field label setup to show in the receipt. Example: My Custom Data | | | | | | | | | | | | | | | | | | | | | | |
| CustomField1 | First custom field setup to show in the receipt. Example: MyField | | | | | | | | | | | | | | | | | | | | | | |
| CustomLabel2 | Second custom field label setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomField2 | Second custom field setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomLabel3 | Third custom field label setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomField3 | Third custom field setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomLabel4 | Fourth custom field label setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomField4 | Fourth custom field setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomLabel5 | Fifth custom field label setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |
| CustomField5 | Fifth custom field setup to show in the receipt. | | | | | | | | | | | | | | | | | | | | | | |

Examples

The following XML code example demonstrates a request to retrieve the printer setup for a terminal using `accountxml.do`.

Printer Setup Information Request

```
xmldata=<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_transaction_type>printersetup</ssl_transaction_type></txn>
```

Printer Setup Information Response

```
<txn>  
<PrinterOptions>  
<PrinterType>Serial Print</PrinterType>  
<PaperType>Multi Ply</PaperType>  
</PrinterOptions>  
<ReceiptHeaderOptions>  
<Justification>Center</Justification>  
<HeaderLine1>RECEIPT LINE 1</HeaderLine1>  
<HeaderLine2>RECEIPT LINE 2</HeaderLine2>  
<HeaderLine3>RECEIPT LINE 3</HeaderLine3>  
<HeaderLine4>RECEIPT LINE 4</HeaderLine4>  
<HeaderLine5>RECEIPT LINE 5</HeaderLine5>  
</ReceiptHeaderOptions>  
<ReceiptTrailerOptions>  
<Justification>Center</Justification>  
<TrailerLine1>RECEIPT LINE 6</TrailerLine1>  
<TrailerLine2>RECEIPT LINE 7</TrailerLine2>  
</ReceiptTrailerOptions>  
<ReceiptCustomFields>  
<CustomLabel1>Custom Field</CustomLabel1>  
<CustomField1>MyCustom</CustomField1>  
<CustomLabel2 />  
<CustomField2 />  
<CustomLabel3 />  
<CustomField3 />  
<CustomLabel4 />  
<CustomField4 />  
</ReceiptCustomFields>  
</txn>
```

Chapter 5: Integration Reference

The following chapter reviews integration reference guidelines and examples to process, format, and send transactions using `process.do`, `processxml.do`, `processBatch.do` and `accountxml.do`.

Topics include:

- Supported transaction input fields
- Fraud prevention matrix
- ISO country codes
- ISO currency codes

Supported Transaction Input Fields

| Field Name | Length | Default | Req | Description |
|-----------------------------------|--------|-----------------|-----|--|
| <code>ssl_3dsecure_value</code> | 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). |
| <code>ssl_account_type</code> | 1 | | Y | Account Type (0 = checking, 1 = saving). |
| <code>ssl_add_token</code> | 1 | N | N | Add token to Card Manager <ul style="list-style-type: none"> • Y = add, • N = do not add |
| <code>ssl_address2</code> | 30 | | N | Customer's address line 2. |
| <code>ssl_amount</code> | 13 | | Y | Transaction total amount. |
| <code>ssl_approval_code</code> | 6 | | Y | Return code generated by credit card processor. Must be passed on force. |
| <code>ssl_avs_address</code> | 30 | | N | Customer's address used to process AVS. |
| <code>ssl_avs_zip</code> | 9 | | N | Customer's ZIP code used to process AVS. |
| <code>ssl_background_color</code> | 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: <ul style="list-style-type: none"> • DAILY • WEEKLY • BIWEEKLY • SEMIMONTHLY • MONTHLY • BIMONTHLY • QUARTERLY • SEMESTER • SEMIANNUALLY • ANNUALLY • SUSPENDED |
| ssl_bill_on_half | 1 | | Y | Half of the month or Semimonthly indicator. Valid values are 1 and 2: <ul style="list-style-type: none"> • 1 = 1st and the 15th of the month • 2 = 15th and the last day of the month |
| ssl_bin_override | 1 | | N | The indicator that allows a Customer Code and/or a Sales Tax value to be passed in the transaction request for non-corporate cards. The value is always 1. |
| 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_card_suffix | 4 | | C | The last 4 digits of the card number. |
| ssl_card_type | | | | The card type, example: CREDITCARD, DEBITCARD |
| ssl_cardholder_ip | 40 | | C | Cardholder IP address. |
| ssl_cardholder_tip_amount | 8 | | C | Cardholder tip amount. |
| 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 ISO code. |

| Field Name | Length | Default | Req | Description |
|----------------------------------|--------|-----------------|-----|--|
| 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_cvv2cvc2 | 4 | | N | CVV2/CVC2 value from the card |
| ssl_cvv2cvc2_indicator | 1 | | N | CVV2 Indicator. Valid values are: <ul style="list-style-type: none"> • 0 = Bypassed • 1 = Present • 2 = Illegible • 9 = Not Present |
| ssl_description | 255 | | N | Transaction description. |
| ssl_do_customer_email | 1 | Set In Terminal | N | Valid values are: <ul style="list-style-type: none"> • T = True • F = False |
| ssl_do_merchant_email | 1 | Set In Terminal | N | Valid values are: <ul style="list-style-type: none"> • 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_dynamic_dba | 21 | | N | DBA Name provided by the merchant with each transaction The maximum allowable Length of DBA Name variable provided by Merchant can be 21, 17 or 12 based on the length setup for the DBA constant in the field setup. |
| ssl_eci_ind | 1 | | Y | Sent on 3D Secure authenticated transactions only. |

| Field Name | Length | Default | Req | Description |
|------------------------------|--------|-----------------|-----|---|
| 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: <ul style="list-style-type: none"> • 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. |
| ssl_email_footer | 255 | | N | Customer order confirmation email footer. If present, overrides values for <code>ssl_email_apprvl_footer_html</code> and <code>ssl_email_decl_footer_html</code> . |
| ssl_email_header | 255 | | N | Customer order confirmation email header. If present, overrides values for <code>ssl_email_apprvl_header_html</code> and <code>ssl_email_decl_header_html</code> . |
| ssl_end_of_month | 1 | | Y | End of month indicator. Valid values Y or N. Must be passed on add or 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 <code>errorCode</code> , <code>errorName</code> , and <code>errorMessage</code> fields. Refer to the Error Codes section for an extensive list of possible codes. |

| Field Name | Length | Default | Req | Description |
|----------------------|--------|-----------------|-----|--|
| ssl_eWallet | 10 | | N | Wallet identifier. Defaulted for <code>process.do</code> (true), only needed for <code>process.do</code> (false) and <code>processxml.do</code> to indicate if Merchant wishes to offer MasterPass. Valid values: MasterPass. The merchant must be set up with MasterPass. |
| ssl_eWallet_shipping | 1 | | N | Indicates if Merchant is wanting to use the shipping information provided with MasterPass, valid values: Y or N. |
| 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 <code>ssl_show_form = False</code> . |
| ssl_get_token | 1 | N | N | Generate Token indicator (Y = generate token, N = do not generate token) |
| 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 <code>ssl_show_form = False</code> . |
| ssl_image_type | 3 | | Y | Image format, required for signature transactions. Possible values, must be capital: <ul style="list-style-type: none"> • GIF • TIF • JPG • PNG |
| ssl_import_file | 255 | | Y | Path/Location and Name of Batch Import File being imported. |
| 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. |

| Field Name | Length | Default | Req | Description |
|--------------------------------|--------|-----------------|-----|--|
| 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 | Converge 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_number_trans | | | N | Number of Transactions imported in the import file. |
| 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 | <p>Partial Auth indicator required to indicate the support of partial approval, defaulted to 0 if not sent.</p> <p>Valid values:</p> <ul style="list-style-type: none"> 0 – Partial Auth not supported 1 – Partial Auth supported |
| ssl_payment_count | 2 | | N | Installment Count (Total Number of Payments). |
| ssl_payment_number | 2 | | N | Installment Sequence Number (Payment Number). |
| ssl_phone | 20 | | N | Customer's phone number. |
| ssl_pin | 6 | | Y | Converge PIN as configured within Converge, case sensitive. |
| ssl_pin_block | | | Y | The encrypted personal identification number entered by debit/EBT cardholder as identification for transaction. |
| ssl_product_string | 200 | | N | Product string for MasterPass transactions. |
| ssl_receipt_apprvl_footer_html | 255 | Set In Terminal | N | Receipt form footer for approved transaction. Ignored when <code>ssl_result_format = ASCII</code> . |

| 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 <code>ssl_result_format = ASCII</code> and <code>ssl_receipt_link_method = LINK</code> or <code>POST</code> . |
| ssl_receipt_apprvl_header_html | 255 | Set In Terminal | N | Receipt form header for approved transaction. Ignored when <code>ssl_result_format = ASCII</code> . |
| ssl_receipt_apprvl_method | 4 | Set In Terminal | N | REDG = No receipt displayed. Data redirected to <code>ssl_receipt_link_url</code> . LINK = Receipt displayed. Link provided to return to <code>ssl_receipt_link_url</code> . GET = Receipt displayed. Button provided to send GET data to <code>ssl_receipt_link_url</code> . POST = Receipt displayed. Button provided to send POST data to <code>ssl_receipt_link_url</code> . LINK, GET, POST ignored when <code>ssl_result_format = ASCII</code> . If present, overwrites <code>ssl_receipt_decl_method</code> and <code>ssl_receipt_link_method</code> . |
| 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 <code>ssl_result_format = ASCII</code> . |
| 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 <code>ssl_result_format = ASCII</code> . |
| 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 <code>ssl_result_format = ASCII</code> and <code>ssl_receipt_link_method = LINK</code> or <code>POST</code> . |

| Field Name | Length | Default | Req | Description |
|------------------------------|--------|-----------------|-----|--|
| ssl_receipt_decl_header_html | 255 | Set In Terminal | N | Receipt form header for declined transaction. Ignored when <code>ssl_result_format = ASCII</code> . |
| ssl_receipt_decl_method | 4 | Set In Terminal | N | REDG = No receipt displayed. Data redirected to <code>ssl_receipt_link_url</code> . LINK = Receipt displayed. Link provided to return to <code>ssl_receipt_link_url</code> . GET = Receipt displayed. Button provided to send GET data to <code>ssl_receipt_link_url</code> . POST = Receipt displayed. Button provided to send POST data to <code>ssl_receipt_link_url</code> . LINK, GET, POST ignored when <code>ssl_result_format = ASCII</code> . If present, overwrites <code>ssl_receipt_decl_method</code> and <code>ssl_receipt_link_method</code> . |
| 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 <code>ssl_result_format = ASCII</code> . |
| 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 <code>ssl_result_format = ASCII</code> . |
| ssl_receipt_header_html | 255 | Set In Terminal | N | Receipt form header. Ignored when <code>ssl_result_format = ASCII</code> . |

| Field Name | Length | Default | Req | Description |
|---------------------------|--------|-----------------|-----|--|
| ssl_receipt_link_method | 4 | Set In Terminal | N | <p>REDG = No receipt displayed. Data redirected to ssl_receipt_link_url.</p> <p>LINK = Receipt displayed. Link provided to return to ssl_receipt_link_url.</p> <p>GET = Receipt displayed. Button provided to send GET data to ssl_receipt_link_url.</p> <p>POST = Receipt displayed. Button provided to send POST data to ssl_receipt_link_url.</p> <p>LINK, GET, POST ignored when ssl_result_format = ASCII.</p> <p>If present, overwrites ssl_receipt_apprvl_method and ssl_receipt_decl_method.</p> |
| ssl_receipt_link_text | 40 | Set In Terminal | N | <p>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.</p> |
| ssl_receipt_link_url | 255 | Set In Terminal | N | <p>Target of the Redirect or the link generated at the bottom of the Converge drawn receipt. Ignored when ssl_result_format = ASCII and ssl_receipt_link_method = GET, POST, or REDG.</p> <p>If present, overwrites ssl_receipt_apprvl_post_url, ssl_receipt_decl_post_url, ssl_receipt_apprvl_get_url, and ssl_receipt_decl_get_url.</p> |
| ssl_recurring_batch_count | 4 | | N | <p>Current number of transactions sitting in the recurring batch after the installment transaction has been added.</p> |

| Field Name | Length | Default | Req | Description |
|----------------------|--------|---------|-----|--|
| ssl_recurring_flag | 1 | | N | <p>The recurring flag must be sent to indicate if a credit card sale transaction is a recurring or an installment payment. This option should only be used if maintaining your own recurring and installment database.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 1 = Recurring • 2 = Installment <p>Note: When the flag is indicated as Installment, the payment number and payment count must be passed.</p> |
| ssl_recurring_id | 50 | | Y | <p>The ID number of the recurring record to be updated. Required on update. Alphanumeric.</p> <p>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.</p> |
| ssl_reference_number | 40 | | N | Transaction reference number is returned in the authorization response. |
| ssl_response_file | 30 | | Y | The batch import response file friendly name, should not contain any of the following characters (\ / : * ? " < >). |
| 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, Converge 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. |

| Field Name | Length | Default | Req | Description |
|------------------------|--------|--------------------|-----|---|
| ssl_salestax | 10 | | N | Sales tax. |
| ssl_search_start_date | 10 | End date minus 31 | C | Start date of the search window. Format MM/DD/YYYY. If you submit a start date without an end date, the end date will be set to start date plus 31 days (not the current date). |
| ssl_search_end_date | 10 | Start date plus 31 | C | End date of the search window. Format MM/DD/YYYY. If you submit an end date without a start date, the start date will be set to end date minus 31 days. |
| ssl_server | 8 | | N | Server ID, this is the clerk, cashier, waiter or waitress identification number,. Alphanumeric. |
| ssl_shift | 4 | | N | Shift, can refer to or used to identify time period, course or type of service. Alphanumeric. |
| 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 ISO code. |
| 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. |
| ssl_show_form | 5 | TRUE | N | When set to false, Converge 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. |

| Field Name | Length | Default | Req | Description |
|--------------------------|--------|-----------------|-----|---|
| 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_tip_amount | 8 | | C | Tip or gratuity amount to be added or updated, must be decimal, can be 0.00. |
| ssl_token | 19 | | C | Credit Card Token generated from the card number. A token can be stored and used as a substitute for a card number. |
| ssl_total_installments | 3 | | Y | Number of payments. Numeric. Max 999. |
| ssl_track_data | 76 | | Y | Track 1 or 2 data as read from a magnetic stripe reader (MSR). |
| ssl_trans_status | | | | Transaction status, example: STL (Settled). |
| ssl_transaction_currency | 3 | USD or CAD | N | <p>Transaction currency must be sent in the 3-digit Alpha ISO 4217 code. For a complete list refer to the ISO Currency Codes section.</p> <p>Note: For those terminals setup for <i>Multi-Currency</i>, you must send a transaction currency to specify the currency used to process the transaction. If a currency is not specified then it will default to the merchant currency. Do not send a transaction currency if your terminal is not setup for <i>Multi-Currency</i></p> |

| Field Name | Length | Default | Req | Description |
|----------------------|--------|---------|-----|---|
| ssl_transaction_type | 20 | | Y | Credit Transactions <ul style="list-style-type: none"> • Sale (ccsale) • Auth Only (ccauthonly) • Return/ Credit (ccccredit) • Force (ccforce) • AVS Only (ccavsonly) • Verification (ccverify) • Balance Inquiry (ccbalinquiry) • Generate Token (ccgettoken) • Enhanced Return/ Credit (ccreturn) • Void (ccvoid) • Completion (cccomplete) • Delete (ccdelete) • Signature (ccsignature) • Update Tip (ccupdatetip) |
| | | | | Recurring Transactions <ul style="list-style-type: none"> • Add (ccaddrecurring) • Update (ccupdaterecurring) • Delete (ccdeleterecurring) • Submit (ccrecurringsale) |
| | | | | Installment Transactions <ul style="list-style-type: none"> • Add (ccaddininstall) • Update (ccupdateinstall) • Delete (ccdeleteinstall) • Submit (ccinstallsale) |
| | | | | Batch Import Transactions <ul style="list-style-type: none"> • Credit Batch Import (ccimport) • Recurring batch Import (ccrecimport) |
| | | | | Debit Transactions <ul style="list-style-type: none"> • Purchase (dbpurchase) • Return (dbreturn) • Inquiry (dbbainquiry) |

| Field Name | Length | Default | Req | Description |
|------------|--------|---------|-----|--|
| | | | | EBT Transactions <ul style="list-style-type: none"> • 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) |
| | | | | Gift Card Transactions <ul style="list-style-type: none"> • Activation (egcactivation) • Sale / Redemption (egcsale) • Card Refund (egccardrefund) • Replenishment / Reload (egcreload) • Balance Inquiry (egcbainquiry) • Credit (egccredit) |
| | | | | Check Transactions (ECS) <ul style="list-style-type: none"> • Electronic Check Purchase (ecspurchase) • Void (ecsvoid) |
| | | | | PINLess Debit Transactions <ul style="list-style-type: none"> • PINLess Debit Purchase (pldpurchase) |
| | | | | Cash Transactions <ul style="list-style-type: none"> • Cash Sale (cashsale) • Cash Credit (cashcredit) |
| | | | | Card Manager Transactions <ul style="list-style-type: none"> • Token Query (ccquerytoken) • Token Update (ccupdatetoken) • Token Delete (ccdeletetoken) |
| | | | | End of Day Transactions <ul style="list-style-type: none"> • Transaction Query (txnquery) • Transaction Email (txnemail) • Settle (settle) |

| Field Name | Length | Default | Req | Description |
|----------------------------------|--------|---------|-----|---|
| | | | | Account Admin Transactions <ul style="list-style-type: none"> Terminal Setup (<code>terminalsetup</code>) Field Setup (<code>fieldsetup</code>) Printer Setup (<code>printersetup</code>) |
| <code>ssl_txn_count</code> | | | | The number of transactions matching the search query. |
| <code>ssl_txn_ecg_count</code> | | | | Total gift card count to be settled. |
| <code>ssl_txn_ecg_amount</code> | | | | Total gift card amount to be settled. |
| <code>ssl_txn_id</code> | 50 | | Y | Transaction identification number. This is a unique number used to identify the transaction. Required for void and delete transactions. |
| <code>ssl_txn_main_count</code> | | | | Total Transaction count in Main Batch to be settled. |
| <code>ssl_txn_main_amount</code> | | | | Total Transaction amount in Main Batch to be settled. |
| <code>ssl_user_id</code> | 15 | | Y | Converge User ID as configured on Converge, case sensitive. |
| <code>ssl_voucher_number</code> | | | Y | The voucher number from an EBT sales slip. Used for Voucher Clear Food Stamp transactions. |
| <code>ssl_xid</code> | | | Y | Sent on 3D Secure authenticated transactions Only. Unique transaction identifier assigned by eMPI. |

Fraud Prevention Matrix

The following table provides the list of fraud prevention rules and applicable transaction types. The fraud prevention rules are set up through the user interface and triggered through requests initiated through `process.do`, `processxml.do`, or `processBatch.do`. Merchant will receive a result message (`ssl_result_message`) of Declined when a fraud rule is triggered. The reason for the decline will display under the error batch as Declined – [Rule Name].

Notes:

- The cardholder IP address (`ssl_cardholder_ip`) is required for the IP address filter and the IP Address velocity filter.
- The billing country ISO code (`ssl_country`) is required for billing country filter.
- The shipping country ISO code (`ssl_ship_to_country`) is required for shipping country filter.
- The cardholder IP address (`ssl_cardholder_ip`) and the billing country ISO code (`ssl_country`) are required for the IP address and billing country mismatch filter.
- The cardholder IP address (`ssl_cardholder_ip`) and the shipping country ISO code (`ssl_ship_to_country`) are required for the IP address and shipping country mismatch filter.
- The email (`ssl_email`) is required for Email Address filter.
- The cardholder IP address (`ssl_cardholder_ip`) value should reflect the IP address from which the transaction originates. For example: Consumer IP for a website clerk workstation IP address.

| Transaction Type | Auto Pend Filter | Merchant IP Address Filter | IP Address Filter | Billing and Shipping Country Filter | IP Address & Country Mismatch Filter | Email Address Filter | Card Number Filter | Email Domain Filter | Velocity Filter |
|------------------|------------------|----------------------------|-------------------|-------------------------------------|--------------------------------------|----------------------|--------------------|---------------------|-----------------|
| ccaddinstall | | X | X | X | X | X | X | X | X |
| ccaddrecurring | | X | X | X | X | X | X | X | X |
| ccauthonly | X | X | X | X | X | X | X | X | X |
| ccavonly | | X | X | X | X | X | X | X | X |
| ccbalinquiry | | X | X | X | X | X | X | X | X |
| ccccredit | X | X | X | X | X | X | X | X | X |
| ccforce | X | X | X | X | X | X | X | X | X |
| ccimport | | X | X | | | | | | X |

| Transaction Type | Auto Pend Filter | Merchant IP Address Filter | IP Address Filter | Billing and Shipping Country Filter | IP Address & Country Mismatch Filter | Email Address Filter | Card Number Filter | Email Domain Filter | Velocity Filter |
|------------------|------------------|----------------------------|-------------------|-------------------------------------|--------------------------------------|----------------------|--------------------|---------------------|-----------------|
| ccrecimport | | X | X | | | | | | X |
| ccreturn | X | X | X | | | | | | |
| ccsale | X | X | X | X | X | X | X | X | X |
| ccverify | | X | X | X | X | X | X | X | X |
| pdpurchase | | X | X | | | | | | X |

ISO Country Codes

The following is a complete ISO 3 encoding list of the countries which are assigned official codes. It is listed in alphabetical order by the English short country name.

| Country | Code | Country | Code |
|---------------------|------|--|------|
| Afghanistan | AFG | Liberia | LBR |
| Åland Islands | ALA | Libya | LBY |
| Albania | ALB | Liechtenstein | LIE |
| Algeria | DZA | Lithuania | LTU |
| American Samoa | ASM | Luxembourg | LUX |
| Andorra | AND | Macao | MAC |
| Angola | AGO | Macedonia, the former Yugoslav Republic of | MKD |
| Anguilla | AIA | Madagascar | MDG |
| Antarctica | ATA | Malawi | MWI |
| Antigua and Barbuda | ATG | Malaysia | MYS |
| Argentina | ARG | Maldives | MDV |
| Armenia | ARM | Mali | MLI |
| Aruba | ABW | Malta | MLT |
| Australia | AUS | Marshall Islands | MHL |
| Austria | AUT | Martinique | MTQ |
| Azerbaijan | AZE | Mauritania | MRT |
| Bahamas | BHS | Mauritius | MUS |

| Country | Code | Country | Code |
|----------------------------------|-------------|---------------------------------|-------------|
| Bahrain | BHR | Mayotte | MYT |
| Bangladesh | BGD | Mexico | MEX |
| Barbados | BRB | Micronesia, Federated States of | FSM |
| Belarus | BLR | Moldova, Republic of | MDA |
| Belgium | BEL | Monaco | MCO |
| Belize | BLZ | Mongolia | MNG |
| Benin | BEN | Montenegro | MNE |
| Bermuda | BMU | Montserrat | MSR |
| Bhutan | BTN | Morocco | MAR |
| Bolivia, Plurinational State of | BOL | Mozambique | MOZ |
| Bonaire, Sint Eustatius and Saba | BES | Myanmar | MMR |
| Bosnia and Herzegovina | BIH | Namibia | NAM |
| Botswana | BWA | Nauru | NRU |
| Bouvet Island | BVT | Nepal | NPL |
| Brazil | BRA | Netherlands | NLD |
| British Indian Ocean Territory | IOT | New Caledonia | NCL |
| Brunei Darussalam | BRN | New Zealand | NZL |
| Bulgaria | BGR | Nicaragua | NIC |
| Burkina Faso | BFA | Niger | NER |
| Burundi | BDI | Nigeria | NGA |
| Cambodia | KHM | Niue | NIU |
| Cameroon | CMR | Norfolk Island | NFK |
| Canada | CAN | Northern Mariana Islands | MNP |
| Cape Verde | CPV | Norway | NOR |
| Cayman Islands | CYM | Oman | OMN |
| Central African Republic | CAF | Pakistan | PAK |
| Chad | TCD | Palau | PLW |
| Chile | CHL | Palestinian Territory, Occupied | PSE |
| China | CHN | Panama | PAN |
| Christmas Island | CXR | Papua New Guinea | PNG |
| Cocos (Keeling) Islands | CCK | Paraguay | PRY |
| Colombia | COL | Peru | PER |
| Comoros | COM | Philippines | PHL |

| Country | Code | Country | Code |
|---------------------------------------|-------------|--|-------------|
| Congo | COG | Pitcairn | PCN |
| Congo, the Democratic Republic of the | COD | Poland | POL |
| Cook Islands | COK | Portugal | PRT |
| Costa Rica | CRI | Puerto Rico | PRI |
| Côte d'Ivoire | CIV | Qatar | QAT |
| Croatia | HRV | Réunion | REU |
| Cuba | CUB | Romania | ROU |
| Curaçao | CUW | Russian Federation | RUS |
| Cyprus | CYP | Rwanda | RWA |
| Czech Republic | CZE | Saint Barthélemy | BLM |
| Denmark | DNK | Saint Helena, Ascension and Tristan da Cunha | SHN |
| Djibouti | DJI | Saint Kitts and Nevis | KNA |
| Dominica | DMA | Saint Lucia | LCA |
| Dominican Republic | DOM | Saint Martin (French part) | MAF |
| Ecuador | ECU | Saint Pierre and Miquelon | SPM |
| Egypt | EGY | Saint Vincent and the Grenadines | VCT |
| El Salvador | SLV | Samoa | WSM |
| Equatorial Guinea | GNQ | San Marino | SMR |
| Eritrea | ERI | Sao Tome and Principe | STP |
| Estonia | EST | Saudi Arabia | SAU |
| Ethiopia | ETH | Senegal | SEN |
| Falkland Islands (Malvinas) | FLK | Serbia | SRB |
| Faroe Islands | FRO | Seychelles | SYC |
| Fiji | FJI | Sierra Leone | SLE |
| Finland | FIN | Singapore | SGP |
| France | FRA | Sint Maarten (Dutch part) | SXM |
| French Guiana | GUF | Slovakia | SVK |
| French Polynesia | PYF | Slovenia | SVN |
| French Southern Territories | ATF | Solomon Islands | SLB |
| Gabon | GAB | Somalia | SOM |
| Gambia | GMB | South Africa | ZAF |

| Country | Code | Country | Code |
|-----------------------------------|-------------|--|-------------|
| Georgia | GEO | South Georgia and the South Sandwich Islands | SGS |
| Germany | DEU | South Sudan | SSD |
| Ghana | GHA | Spain | ESP |
| Gibraltar | GIB | Sri Lanka | LKA |
| Greece | GRC | Sudan | SDN |
| Greenland | GRL | Suriname | SUR |
| Grenada | GRD | Svalbard and Jan Mayen | SJM |
| Guadeloupe | GLP | Swaziland | SWZ |
| Guam | GUM | Sweden | SWE |
| Guatemala | GTM | Switzerland | CHE |
| Guernsey | GGY | Syrian Arab Republic | SYR |
| Guinea | GIN | Taiwan, Province of China | TWN |
| Guinea-Bissau | GNB | Tajikistan | TJK |
| Guyana | GUY | Tanzania, United Republic of | TZA |
| Haiti | HTI | Thailand | THA |
| Heard Island and McDonald Islands | HMD | Timor-Leste | TLS |
| Holy See (Vatican City State) | VAT | Togo | TGO |
| Honduras | HND | Tokelau | TKL |
| Hong Kong | HKG | Tonga | TON |
| Hungary | HUN | Trinidad and Tobago | TTO |
| Iceland | ISL | Tunisia | TUN |
| India | IND | Turkey | TUR |
| Indonesia | IDN | Turkmenistan | TKM |
| Iran, Islamic Republic of | IRN | Turks and Caicos Islands | TCA |
| Iraq | IRQ | Tuvalu | TUV |
| Ireland | IRL | Uganda | UGA |
| Isle of Man | IMN | Ukraine | UKR |
| Israel | ISR | United Arab Emirates | ARE |
| Italy | ITA | United Kingdom | GBR |
| Jamaica | JAM | United States | USA |
| Japan | JPN | United States Minor Outlying Islands | UMI |
| Jersey | JEY | Uruguay | URY |

| Country | Code | Country | Code |
|--|------|-----------------------------------|------|
| Jordan | JOR | Uzbekistan | UZB |
| Kazakhstan | KAZ | Vanuatu | VUT |
| Kenya | KEN | Venezuela, Bolivarian Republic of | VEN |
| Kiribati | KIR | Viet Nam | VNM |
| Korea, Democratic People's Republic of | PRK | Virgin Islands, British | VGB |
| Korea, Republic of | KOR | Virgin Islands, U.S. | VIR |
| Kuwait | KWT | Wallis and Futuna | WLF |
| Kyrgyzstan | KGZ | Western Sahara | ESH |
| Lao People's Democratic Republic | LAO | Yemen | YEM |
| Latvia | LVA | Zambia | ZMB |
| Lebanon | LBN | Zimbabwe | ZWE |
| Lesotho | LSO | | |

ISO Currency Codes

The following is a complete ISO 4217 encoding list of the currencies which are assigned official codes. It is listed in alphabetical order by the English short currency name.

| Currency | Code | Currency | Code |
|-------------------------------|------|-------------------|------|
| United Arab Emirates Dirham | AED | Kazakhstan Tenge | KZT |
| Netherlands Antillean Guilder | ANG | Lebanese Pound | LBP |
| Argentine Peso | ARS | Sri Lanka Rupee | LKR |
| Australian Dollar | AUS | Lithuanian Litas | LTL |
| Aruban Florin | AWG | Latvian Lats | LVL |
| Azerbaijani Manat | AZN | Libyan Dinar | LYD |
| Barbados Dollar | BBD | Moroccan Dirham | MAD |
| Bangladeshi Taka | BDT | Macedonian Denar | MKD |
| Bulgarian Lev | BGN | Mauritian Rupee | MUR |
| Bahraini Dinar | BHD | Malawian Kwacha | MWK |
| Bermudian Dollar | BMD | Mexican Peso | MXN |
| Brazilian Real | BRL | Malaysian Ringgit | MYR |
| Bahamian Dollar | BSD | Namibian Dollar | NAD |
| Botswana Pula | BWP | Nigerian Naira | NGN |
| Canadian Dollar | CAD | Norwegian Krone | NOK |

| Currency | Code | Currency | Code |
|---------------------|-------------|----------------------------|-------------|
| Congolese Franc | CDF | Nepalese Rupee | NPR |
| Swiss Franc | CHF | New Zealand Dollar | NZD |
| Chilean Peso | CLP | Omani Rial | OMR |
| China Yuan Renminbi | CNY | Peruvian Nuevo Sol | PEN |
| Colombian Peso | COP | Philippine Peso | PHP |
| Costa Rican Colon | CRC | Pakistani Rupee | PKR |
| Czech Koruna | CZK | Polish Zloty | PLN |
| Danish Krone | DKK | Qatari Rial | QAR |
| Dominican Peso | DOP | Romanian New Leu | RON |
| Algerian Dinar | DZD | Serbian Dinar | RSD |
| Estonian Kroon | EEK | Russian Ruble | RUB |
| Egyptian Pound | EGP | Saudi Riyal | SAR |
| Ethiopian Birr | ETB | Swedish Krona | SEK |
| Euro | EUR | Singapore Dollar | SGD |
| Fiji Dollar | FJD | Syrian Pound | SYP |
| Pound Sterling | GBP | Thai Baht | THB |
| Guatemalan Quetzal | GTQ | Tunisian Dinar | TND |
| Hong Kong Dollar | HKD | Turkish Lira | TRY |
| Croatian Kuna | HRK | Trinidad and Tobago Dollar | TTD |
| Haitian Gourde | HTG | New Taiwan Dollar | TWD |
| Hungarian Forint | HUF | Ukrainian Hryvnia | UAH |
| Indonesian Rupiah | IDR | US Dollar | USD |
| Israeli Shekel | ILS | Venezuelan Bolivar Fuerte | VEF |
| Indian Rupee | INR | Vietnamese Kong | VND |
| Iranian Rial | IRR | Gabon Franc | XAF |
| Icelandic Krona | ISK | East Caribbean Dollar | XCD |
| Jamaican Dollar | JMD | Ivory Coast Franc | XOF |
| Jordanian Dinar | JOD | Fr. Polynesia Franc | XPF |
| Japanese Yen | JPY | South African Rand | ZAR |
| Kenyan Shilling | KES | Zambian Kwacha | ZMK |
| South Korean Won | KRW | Zimbabwe Dollar | ZWL |
| Kuwaiti Dinar | KWD | | |

Chapter 6: Additional Processing Options

The following sections review the transaction processes and provide implementation guidelines and examples to format, process, and send payments with specific needs using `process.do`, `processxml.do` and `processBatch.do`.

Topics include:

- 3D Secure
- MasterPass
- Dynamic Currency Conversion (DCC)
- Multi-Currency Conversion (MCC)
- Tokenization
- Encryption
- Electronic Check ACH ECheck

3D Secure

Converge uses the Elavon eMPI engine to allow processing of 3D secure transactions. The following transactions types are supported when processing 3D secure transactions:

| Transaction Types | Description |
|-------------------|-----------------------|
| ccsale | Credit Card Sale |
| ccauthonly | Credit Card Auth Only |

Notes:

- The terminal must be setup for e-Commerce.
 - The terminal must be setup for 3D Secure.
 - 3D Secure processing is supported with MasterCard and Visa transactions only. Other card types will process as normal and will not trigger 3D secure processing.
-

The process of authentication is to retrieve and pass the following Issuer Authentication variables:

| Field Name | Req? | Description |
|--------------------|------|--|
| ssl_eci_ind | Y | <p>e-Commerce indicator or ECI Values</p> <p>(fully authenticated) - There is a liability shift and the merchant is protected from chargeback</p> <p>(VbV has been attempted) - There is a liability shift and the merchant is protected from chargeback</p> <p>(Non-VbV transaction) – The merchant is no longer protected from chargeback</p> |
| ssl_3dsecure_value | Y | Cardholder Authentication Verification Value or CAVV |
| ssl_xid | C | Unique transaction identifier assigned by eMPI |

Transaction Flow

Using process.do

If you are using `process.do` to integrate to Converge, there is no additional work required to utilize the eMPI, as the authentication piece is built in and the authentication variables are passed for you. The following steps outline the process of sending a 3D Secure transaction with `process.do`:

1. The website processes Auth only or Sale transactions and collects the payment information using `process.do`. Refer to [Credit Card Sale \(ccsale\)](#) and [Credit Card Auth Only \(ccaauthonly\)](#) sections for more information.
2. Consumer enters Visa or MasterCard card number at the check-out.
3. The website is redirected automatically to the issuer website.
4. Cardholder completes the authentication process by either enrolling in the program for the first time and successfully authenticating, or authenticating using an existing account with the issuer or simply by opting out if allowed by the issuer.
5. The issuer passes the Issuer Authentication variables to Converge and redirects the cardholder browser back to Converge.
6. Converge processes the transaction with the authentication variables and returns a response to the website displaying the results and the ECI response code in the response.

Using processxml.do

If you are using `processxml.do` to integrate to Converge, you have to pass the Issuer Authentication variables `ssl_eci_ind`, `ssl_3dsecure_value` and `ssl_xid` in the Credit Card Sale or the Credit Card Auth Only requests. Those values are obtained by integrating your system to any eMPI capable engine of your choice. Converge has a built-in integration to the Elavon eMPI service that can be utilized free of charge.

The following steps outline the process of sending a 3D secure transaction with processxml.do using Elavon built-in eMPI engine:

1. Cardholder enters a Visa or a MasterCard card number at the checkout.
2. The website collects the payment information using processxml.do and sends an enrollment request to Converge.
3. Converge will determine if the card is eligible for 3D Secure processing and responds to the website with a URL, a merchant data value and an encoded payer request value.

Note: This completes the first stage in the authentication process.

Shown below is an example of a card enrollment request using processxml.do:

Step 1: Send a card enrollment request

```
xmldata=<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_transaction_type>paenrolled</ssl_transaction_type>
<ssl_card_number>0000000000000000</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_amount>1.00</ssl_amount>
</txn>
```

Receive a card enrollment response along with the issuer URL:

```

<txn>
  <acs_url>https://secure.issuerwebsite.com/Visaormastercard.jsp
  </acs_url>
  <pareq>eJxVUu9PwjAQ/VeWfYe2uOEktxoEiSYWBaeJn0zTHTJlHXRFYH+9LQ5/fG
  nuXa/v3r0rXO7LVfCJpi4qnYasS8MAtaryQr+l4VM26SRhUFupc7mqNKbhAevwkkO
  2NIjjR1RbgxwElrV8w6DI03AtDW5ekySO8hgXnyj17ogX553kgkUddY6SLlS/r2Qv
  5PAwnOOGQ9ufu/bdHpATdLxGLaW2HKTaXN10+d39ahj3KkZA2gSuaG7HXIwoZUC+A
  WhZIRDY28D1jgBUtdXWHHgSUSANAFuz4ktr1wNCdrtdt9AWjUZbH8fqqoE4kuA/A
  p52PqodpT7IufiQ5xNG9VMs/lePEZsRieTeTazz+PrFIivgFxa5D3KIhr3koD1B1E
  0oDGQYx5k6bW4eyerjWhtWwz/XPxNgJNm3IZOo5wQ4H7tFuSfAPmJgffzqHd14G5V1
  Dk2z61g0H40YD3fi/eVMNC/sfpam3tZjgWcrnEMS9ra2AIinIO3OSL0F/37DF+Vks
  sSF</pareq>
  <md>r00ABXNyACZjb20ubm92YWluZm8uYWRhcHRlcj5lbXBpLk1lcmNoYW50RGF0Y
  QAAAAAAAAABAgAETAACaWR0ABJMamF2YS9sYW5nLlN0cm1uZztMAApwYXJhbWV0ZX
  JzdAATTGphdmEvdXRpbC9IYXNoTWFwO0wAFnRocmVlRfNlY3VyZU1lcmNoYW50SWR
  xAH4AAUwAA3hpZHEAfgABeHB0ACFOdnpiV3VNBGQ4TnA3ZkE3RjU2X1FGdTEwMTY0
  MTMyNTlzcGARAamF2YS5ldGlsLkhhc2hNYXAFB9rBwxZg0QMAAkYACmxvYWRGYWN0b
  3JJAAl0ahJlc2hvbGR4cD9AAAAAAAAAMdwgAAAAQAAAAAnQACXNlc3Npb25JZHEAfg
  AEdAAMZGNjUmVxdWVzdGVkdAABTnh0AApFTVBjLTAwMDAxZAAUMkM3NzcZNRFM54
  1Q0FFRTQtVDE=</md>
  <enrolled>Y</enrolled>
</txn>

```

- If Converge indicates that the card is enrolled, the website will need to redirect the cardholder to the URL returned in the acs_url field and posts the results received from Converge. The post must include the merchant URL so the issuer can redirect the cardholder back to the merchant once the authentication has been completed. The website will receive a new merchant data value and an encoded payer response value from the issuer.
- The website sends a second request to Converge with the merchant data value and an encoded payer request value. PaRes value obtained from the issuer in step 1.

Notes:

- This completes the second stage in the authentication process.
- There is a 15-minute window given to allow the website to respond back to Converge.

Shown below is an example of a payer authentication:

Step 2: Send a Payer Authentication Request

The Merchant will redirect the Customers Browser to the Issuer website and post the authentication request (`paReq`). Inline authentication Windows with or Without Frames are recommended. Popup windows are *not* allowed. 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="eJxVUu9PwjAQ/VeWfYe2uOEktxoEiSYWBaeJn0zTHTJLHXRFYH+9LQ5/fGn
uXa/v3r0rXO7LVfCJpi4qnYasS8MAtaryQr+l4VM26SRhUFupc7mqNKbhAevwkk02N
IjjR1RbgxwElrV8w6DI03AtDW5ekySO8hgXnYj17ogX553kgkUddy6SL1S/r2Qv5PA
wn0OGQ9ufu/bdHpATdLxGLaW2HKTaXN10+d39ahj3KkZA2gSUAg7HXIwoZUC+AWhZI
rdY28D1jgBUtdXWHHgSUSAnAFuz4ktr1wNCdrtdt9AWjUZbH8fqqqoE4kuA/Ap52Pq
odpT7IufiQ5xNG9VMs/1EPEZsRieTeTazz+PrFIivgFxa5D3KIhr3koD1B1E0oDGQY
x5k6bW4eyerjWhtWwz/XPxNgJNm3IZOo5wQ4H7tFuSfAPmJgfzqHd14G5V1Dk2z61g
0H40YD3fi/eVMNC/sfpam3tZjgWcrnEMs9ra2AIinIO3OSL0F/37DF+VksSF">
<input type="hidden" id="TermUrl" name="TermUrl"
value="https://www.merchantwebsite.com/3DSReturn.jsp"> <input
type="hidden" id="MD" name="MD"
value="r00ABXNyACZjb20ubm92YWluZm8uYWRhcHRlci5lbXBpLk11cmNoYW50RGF
0YQAAAAAAAAABAgAETAACaWR0ABJMamF2YS9sYW5nL1N0cm1uZztMAApwYXJhbWV0Z
XJzdAATTGphdmEvdXRpbC9lYXNoTWFwO0wAFnRocmVlRfNlY3VyZU11cmNoYW50SWR
xAH4AAUwAA3hpZHEAfgABeHB0ACFOdnpiV3VNbGQ4Tna3Zke3RjU2X1FGdTEwMTY0M
TMyNTlzcGARamF2YS5ldGlsLkhhc2hNYXAFB9rBwxZg0QMAAkYACmxvYWRGYWN0b3J
JAA10ahJlc2hvbGR4cD9AAAAAAAAAMdwgAAAAQAAAAAnQACXNlc3Npb25JZHEAfgAEd
AAMZGNjUmVxdWVzdGVkdAABTnh0AApFTVBjLTAwMDAxdAAUMkM3NzczNTRFMS41Q0F
FRtQtVDE="> <input type="submit" name="Proceed to Issuer Website">
</form>
```

Receive the Payer Authentication response:

```

PaRes =
eJzFWMmSo8qS3ddXlNVbyrKYEVxTpVkwCIEUSCAmsbnGJECMyhDD1zfKrKzKV32trV
9vWhT5eES4H3ePODhsjKSJLuEcBX0Tvw5g1LZeHH1Nwx/faq J7n9HHhUEBM2
sASBvPA0hb8wjHd9wQIcw5k1g1Ek9u11cwJ61P7a1v4dYER49ejgJaJY9oUMmfCF8c
PwBV0H1wilInLNRMu2R9S0aVW Yt/R7/gG RguOJog8crudeMfD05WXw9HHhz
tuQN8lOxKaJGF14tGUWxDfI
2JReEb12Udt9XXRvg01Q9WXXTK8MiW6Qj8Gmb/LXpOvqvxBkGIbvadlFTR117Vsavg
dVsUGeSzbIbyCn/imli8kxDV9hZqLQgJR
U3k9pURV2PKqkXWWIP7YIM8Vm9DrolccxUiUwamvGPKXuv4LZzfIm37jFU8sy/wC66
e8qZ8uwKeJz4rNAq2JyuAjli/RJhrrqoyeWzbIL3mD/MZbe Ur
umHLb/F9qLdGM7rpkuLP3CyfxHLgjf9pu28rm9fLxvkp7QJvMf jFRryrAomrs4xBQV
xgrOGqbeAPBpgif9tySYK0leUWkAt/2
7QB5XTdolxRPqvys2yBMK8naMXjfnNC4XZ030dSzysv3x7VotBuJ71cTIkh8UQV1kW
RC2afyvb 7olAur9V/ti33yqPMay9PZ69bTh
MuqQKv/7C9k9mDP1pCUN0kX9ZTL0EGFm PDUogVHfke8R/G
s/Qmqab2XNVGwpyE9ukbPQkdfTV3 8elf/9H1EtJ4uQz/Fwgf7t8tWF7eR6
t7EQmc jhmdGS1Y63O6rv2eSGMMNS888rN8gv2Iv8uy6f8vK
8MgnD8m/zFHL7ixZJmNWQCZvq9qlR3IeSZ/97LYyrxoj8rF/de3hAtNjdGfpiEw9dN
WxcgWv5hCG82B8KZ1c7c5YpnEFmumCpNc2IXCVJu7unsR6pVfHV9NvS5jtwMXDmxxv
pxOXAbFQR53SS1cSc1q8YpKA6l YAFfbNUZNDHPXUk066S13PTVwp
pkM5lg32UEKvMefEwJNM
7PYaLIX2ED79SOJkrZ1lx6NZJLeB1X/rAlCPixl9Q8igHV9Hr9dNIYI7onMeYs3u9s
s7MqmGvHpk/gCMw59W zUigY/4tqPQpZjQBP6xWYsPevlzn0aZUqzuY02N2
ZS3vMzZ3Vf3NSYAD/Z0RmWE9uPHe I/JXuzj6b3KizCT5V
Br8HsAr7fLniBw/50mVe16pEHLWgWgch2j0B2bUjn8zxmjtCzzaK brzRR
pHgCvsvKC7Z04n6L8BuqjSis2oV8cDhw5dPKFd0taOHC6qjBMZX4gETGncQmWi65d
KAjJzZOS/HAFPI21df5jY6Q87xyD/qa6WkMVe9NJHT1Pf0iGtoTT3RnWOA0GSXcdUa
kOfE3G3gDczRXYFJNuvY1TGgdX2tEWV
1W8/e5Fzd45aukHQ0CzFewlpFvitybimqx01D7c3LWaJhgxDU6vB7aATdE6gDFpqB
HYyAgTLyEq8Y8keln1b5aWXDwUpCoiQVU1fIMKXU1p3kb7f10q5pUJrO8iovarntNiO
u2E3AIHGXXGGL4VuUjyxvxg6yBBrgNiv7m7n
rCvJbdCiUFbzOe5f4qOnS68Jcy/MEyrJQCDwPKj4Gg8yBWNABYUIPAqEOYsZVCg44J
zoX0mUQtIuyr1w5eQQq0MQtp4EhEMQDBJKEMFPkEshrqDiKAjhysWotxgwoZftQ2k6
ezYyIAU7v
sDgRTbXClf2pTGHejBstYtgadpeHCjFL/U8KKg83Kr5hbAm1xBNyM1vPvgRHnRpi17
OnODaFBoU5igY4PBuFxcZKmmA27JFChAYajIC7P28twNETSfurmp3Q3joVaO/Dv/
iVxUATNEDUIyHd/HNwZOIVeHKX2Z1GHgHmPdYSyGw
bsMhbXzLH7QysdwyVIWT5ErM1
YWFmrh1Cx1liVEbxPjNx0EcOtF13NyX2MG3rT4URA9y8ZtdMELdNTHtfoY4DYOxbmO
Yv8TrE/IQx2IKASrx57t0ln1C0EQOaCYApMwJA3jo70G11FDjJ8pq3dupz1A/oLjp1
g2POMNAkrvUI3W8TCTH2j36Kh02
mOVMOkXnsObeZf4tULbkm9Iosr9cJtcTdcjSawU1wq2WxHN64GWL8p8Q8TAX7e86Zb
luGMQnN8H2mO6xLay6
1TvWYPJyrrdo9p3Tgafu/II17cZVfdTcLhjqkMQJRjMESYSuYhWxBaOVkh150mBbgr
Opt29c73SqbsWP64xmFqoolrieWtXsOj4AVTRqze
Mo/FNale7agdGcIWYPBMJ3PeNjtrnfnsznUMA1Vp0R20jJC2OL1RTRU2qHd4f3Y
3EHM9pT/sMXOOLjBKrelYET06ZDrjpPp17kxEiNjyZc1T5gw9iyAEg3WQBRBAMz3qF
4iByyKDxEIBBENZXR09A2yFLEQQQ2/Gf9eJ
1mupkX03t3bIMcrg7UakOWdlxQzuecYurL70REZ7pFey7Tlr17mkBP7QptpLVhLjT7
d05lp1nKkdjgpHTY31gz7ZQTqToRrvj3KvMSmzonjysCK2ebsv7Lagdc6BN9oLJOzS

```

(continued)

```

YgFisAXpEAGeBnWK3I7cQZCtU8fgu44SwtsNqNDFtXUjddo5x70h52qMWteK3VissR
LOxmpIekwb8KtWYCdZJiK5yGdNvukJJOi6MPupHXvYs
POSf1BXp7yqz58WOZcYL3tuw
GLjWkaTHpFJxa9cXMiBnZrAcDNpZqQVbUziQ1OfhUcRC31IYtG9E/SIHeFUT/CowL/
tjOseKBoe8QQdbm6dsBNcN8ifD/SPlxU/KK9IPytMkpBsa6NjaqsebZnc7VKw2L0ng
4uyeZ0lyUdHnVdsCgRv P
ji4uiJLKr5Qo1P0lQhaD/oQnyjCwNE2wEdoQBIaGgDFExSFUJv0U1/6uA0jPwM1Hf/
FwNkKgfln8dX02RxCE
LnyQo8gfUxUEY3vQ7cUhoF7vDPXtsFwwO5Mz3mBN4PC9U7DpK7zq64doq9qTlrQDO7
z5ag0flJjTEWMMXutst8WnowL/bFcRBzw2LU5a6cPJ/y7m4BWB5h9MY8Jzn4/0ii4B
kKm57u06aNleg1NSEvVqVd3puVHYGVSk/HLJcFfNv
ZLNHGf2vxM6FgeZ2g9NrlvZgfEMK8odcyq1x8wiYctLe55uHqVKYyWL2s0jz6e6Odi
oDIQZq yxokHMO2iicPq5JERXZ1Li5CcPHCuJjjk4iBZq8fVmnOs1S/Cid11t9dJ
aJUTlNK5qobz/tagEQ9DI4iUw02wrL4oZ2kX7ViJjsG9P5ME268SFFjTWIejrTthce
e1CuCodL65u6SYDddSZM5A1JUbW4fBVqQrR2hAnc9FQ
KMc4eRtlLOHXIzpzqXqBQ7ojPwafG5nQ5kN
/by3m/WwqSONxC7Vs19luerWdfu0ldW4ht6Siq8tJ81G6nL5R3XbpoAUhPWGoxyP0P
1Ea0TJykhZcIdMJHR3UdwnFJNjMNibHe9zp9Jsney
jbnkTO1C5UUYi7pS/zfHy1jSHLdhY3a3x62ZNms yxNNuvOMNu4aVu
61jUgEpuTtUMo9WEAnqqqd44yYvQM7bsn
Ep4khRpZTee2UoKca7pCu16zQIYPjbrQIZqUISJrIJwEaTUI9m
4rPjtGzxp780nwDFs5N
mRLJxBPqeZCCdrWDptnbxMsOtGymXNHCOVow6DrLEhdehBo911dlIq/HSpcWltZA89
PSAuagplYI8EW17y7tF4km8QEIZ0Rd52e mK70
XhxmcmEmr1GJ0GW056qV8Z1NBjZN5btXetstNo8CbSJJL2D5SG/G7okF/N e
2/e37w9sHmef78ecPNf8FfxDQ0w==
MD =
r00ABXNyACZjb20ubm92YwluZm8uYWRhcHRlci5lbXBpLk11cmNoYW50RGF0YQAAAA
AAAAABAgAETAACaWR0ABJMamF2YS9sYW5nL1N0cmLuZztMAApwYXJhbWV0ZXJzdAAT
TGphdmEvdXRpbC9IYXNoTWFW00wAFnRocmVlRfNlY3VyZU11cmNoYW50SWRxAH4AAU
wAA3hpZHEAfgABeHB0ACFZTKE2cWFpU28yZDBzb0FBZGtGTDDaZDEwMTY0MTMyNThz
cgARamF2YS5ldGlsLkhhc2hNYXAFB9rBwxZg0QMAAkYACmxvYWRGYWN0b3JJAAl0aH
Jlc2hvbGR4cD9AAAAAAAMdwgAAAAQAAAABXQADXL1bmRvckFwcE5hbWV0AAB0AA1z
ZXNzaW9uSWRxAH4ABHQADGRjY1JlcXVlc3RlZHQAAU50AAh2ZW5kb3JjZHB0ABB2ZW
5kb3JBcHBWZXJzaW9ucQB
AAh4dAAKRUIQSS0wMDAwMXQAFDJFNDEzOUYzQkYuRDQxQjU5LVQx
    
```

(end)

- 6. Converge will validate the PaRes value sent and responds with the Issuer Authentication variables needed to complete the ccsale or ccauthonly transaction.

Note: This completes the third and final stage in the authentication process.

Shown below is an example of a PaRes verification request using processxml.do:

Step 3: Send a PaRes Verification request, include the PaRes and MD values received from the issuer in step 2:

```

xmldata=
<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_transaction_type>parequest</ssl_transaction_type>
<md>r00ABXNyACZjb20ubm92YWluZm8uYWRhcHRlci5lbXBpLk11cmNoYW50RGF
0YQAAAAAABAgAETAACaWR0ABJMamF2YS9sYW5nL1N0cmLuZztMAApwYXJhbW
V0ZXJzdAATTGphdmEvdXRpbC9lYXNoTWFWw0wAFnRocmVlRfNlY3VyZU11cmNoY
W50SWRxAH4AAUwAA3hpZHEAfgABeHB0ACFZTke2cWFpU28yZDBzb0FBZGtGTDda
ZDEwMTY0MTMyNThzcgARamF2YS51dGlsLkhhc2hNYXAFB9rBwxZg0QMAAkYACmx
vYWRGYWN0b3JJAAl0aHJlc2hvbGR4cD9AAAAAAAMdwgAAAAQAAAAABXQADXZlbn
RvckFwCE5hbWV0AAB0AA1zZXNzaW9uSWRxAH4ABHQADGRjY1JlclXVlc3RlZHQAA
U50AAh2ZW5kb3JlZHB0ABB2ZW5kb3JBcHBWZXJzaW9ucQB
AAh4dAAKRUIQSS0wMDAwMXQAFDJFNDEzOUYzQkYuRDQxQjU5LVQx</md>

  <pires>
eJzFWMmSo8qS3ddXlNVbyrKYEVxTpVkwCIEUSCAmsbnGJECMyhDD1zfKrKzKV3
2trV9vWht5eES4H3ePODhsjKsJIuEcBX0TvW5g1LZeHH1Nwx/faq
J7n9HHhUEBM2
sASBvpA0hb8wjHd9wQIcw5k1g1Ek9u11cwJ61P7a1v4dYER49ejgJaJY9oUMmf
CF8cPwBV0H1wilInLNRmu2R9S0aVW Yt/R7/gG
Rgu0Jog8crudeMfd05Wxw9HHhz tuQN810xKaJGF14tGUWxDfI
2JReEb12Udt9XXRvg01Q9WXXTK8MiW6Qj8Gmb/LXpOvqvxBkGIbvadlFTR117V
savgdVsUGeSzbIbyCn/imli8kxDV9hZqLQgJR
U3k9pURV2PKgkXWWIP7YIM8Vm9DrolccxUiUwamvGpkXuv4LZzfIm37jFU8sy/
wC66e8qZ8uwKeJz4rNAq2JyuAjlI/RJhrrqoyeWzbIL3mD/MZbe Ur
umHLb/F9qLdGM7rpkuLP3CyfxHLgjf9pu28rm9fLxvkp7QJvMfjFRryrAomrs4
xBQVxgrOGqbeAPBpgif9tySYK0leUWkAt/2
7QB5XTdolxRPqvys2yBMK8naMXjfnNC4XZ030dSzysv3x7V0tBuJ71cTIkh8UQ
VlkWRC2afyvb 7olAur9V/tI33yqpMAy9PZ69bTh
MuqQKv/7C9k9mDP1pCUN0kX9ZTL0EGFm PDUogVHfke8R/G
s/Qmqab2XNvGwpyE9ukbPQkdfTV3 8e1f/9H1EtJ4uQz/Fwg7t8tWF7eR6
t7EQmcjhmhdGS1Y6306rv2eSGMMNS888rN8gv2Iv8uy6f8vK
8MgnD8m/zFHL7ixZJmNWQCZvq9qlR3IeSZ/97LYrxoj8rF/de3hAtNjdGfpiE
w9dNWxcgWv5hCG82B8KZ1c7c5YpneFmumCpNc2IXCVJu7unsR6pVfHV9NvS5jt
wMXDmxxvpxOXAbFQR53SS1cSc1q8YpKA61 YAFFbNUZNDHPXuk066S13PTVwp
pkM5lg32UEKvMefEwJNM
7PYaLIX2ED79SOJkRZ1lx6NZJLeB1X/rAlCPixl9Q8igHV9Hr9dNIYI7onMeYs
3u9ss7MqmGvHpk/gCMw59W
zUigY/4tqPQpZjQBP6xWYsPevlzn0aZUqzuY02N2
ZS3vMzZ3Vf3NSYAD/Z0RmWE9uPHe I/JXuzj6b3KizCT5V
Br8HsAr7fLnibW/50mVel6pEHLWgWgcH2j0B2bUjn8zxmjtCzzaK brzRR
pHgCvsvKC7ZO4n6L8BuqjSis2oV8cDhw5dPKFd0taOHC6qnjBMZX4gETGncQmW
i65dKAjJzZOS/HAFPI21df5jY6Q87xyD/qa6WkMVe9NJHT1Pf0iGTtoTT3RnWOA
0GSXcdUakOFe3G3gDczRXyFJNuvY1TGdx2tEWV

```

(continued)

```

1W8/e5Fzd45aukHQ0CzFewlpFviTybmimqx01D7c3LWaJhgxDU6vB7aATdE6gD
FpqbHYyAgTLyEq8Y8kelnl5aWXDwUpCoiQVU1fIMKXU1p3kb7f10q5pUJrO8i
ovartNiOu2E3AIHGXXG14VuUjyxvvg6yBBrgNiv7m7n
rCvJbdCiUFbzOe5f4qOnS68Jcy/MEyrJQCDwPKj4Gg8yBWNaBYuIPaqEOYszVC
g44JzoX0mUQtIuyrlw5eQQq0MQtp4EhEMQDBJkEMFPkEshrqDiKAjhyWotxgw
OZftQ2k6ezYyIAU7v
sDgRTbxCLf2pTGHejBstYtgadpeHCjFL/U8KKg83Kr5hbAmlxBNyMlvPvgrHnR
pi17OnODaFBoU5igY4PBuFxcZKmmMA27JFChAYa jIC7P28twNETSfurmp3Q3j
oVaO/Dv/iVxUATNEDUIyHd/HNwZOIVEHKX2Z1GHgHmPdYSygW
bsMhbXzLH7QysdwyVIWT5ErM1
YWFmrh1Cx1liVEbxPjNx0EcOtF13NyX2MG3rT4URA9y8ZtdMELdNTHtfoY4DYO
xbmOYv8TrE/IQx2IKASrx57t0ln1C0EQOaCYApMwJA3jo70G11FDjJ8ppq3dupz
1A/oLjplg2PomNAkrvUI3W8TCTH2j36KhO2
mOVMoKXnsObeZf4tULbkM9Iosr9cJtcTdCjSawUlwq2WxHN64GWL8p8Q8TAX7e
86ZbluGMQnN8H2m06xLay6
1TvWYPJyrrdo9p3Tgafu/II17cZVfdTcLhjgkMQJRjMESYSuYhWxBAOVkh150m
BbgrOpt29c73SqbsWP64xmFqoolrieWtXsOj4AVTRqzE
Mo/FNale7agdGcIWYPBMJ3PeNjtrnfnaSznUMA1VpOR20jjc2OL1RTRU2qHd4f
3Y
3EHM9pT/sMX0LjBKrelYETo6ZDrjpPp17kxEiNjyZc1T5gw9iyAEg3WQBRBAM
z3qF4iByyKDXEIBBENZXR09A2yFLEQQQ2/Gf9eJ
lmupkX03t3bIMcrg7UakOWdlxQzuecYuRL70REZ7pFey7Tlr17mkBP7QptpLVh
LjT7d05lp1nKkDjgpHTY3lgz7ZQTqToRrvj3KVMsmzonjysCK2ebsv7Lagdc6B
N9oLJOzSYgFisAXpEAGEBnWK3I7cQZCtU8fgu44SwtSNqNDFtXUjddo5x70h52
qMWteK3VissRLOxmpIekwb8KtWYCDZJiK5yGdNvukJJOi6MPupHXvYs
POSf1BXp7yqz58WOZcYL3tuw
GLjWkaTHpFJxa9cXMiBnZrAcDNpZqQVbUziQ10fhUcRC31IYtG9E/SIHeFUT/C
owL/tjOseKBoe8QQdbm6dsBNcN8ifD/SPlxU/KK9IPytMkpBsa6NjaqsebZnc7
VKw2L0ng4uyeZ0lyUdHnVdsCgRv P
ji4uiJLKr5Qo1P0lQhaD/oQnyjCwNE2wEdoQBiaGgDFExSFUJv0U1/6uA0jPwM
lHF/FwNkKgf1n8dX02RxCE
LnyQo8gfUxUEY3vQ7cUhoF7vDPXtsFwwO5Mz3mBN4PC9U7DpK7zq64doq9qTlr
QDO7z5ag0f1JJTEWMMXutst8WnowL/bFcrBzW2LU5a6cPJ/y7m4BWB5h9MY8Jz
n4/0ii4BkM57u06anleg1NSEvVqVd3puVHYGVsk/HLJcFfVn
ZLNHGf2vxM6FgeZ2g9Nr1vZgFEMK8odcyq1x8wiYctLe55uHqVKYVWL2s0jZ6e
6OdiDIQZq
yxokHMO2iiCPq5JERXZ1Li5CcPHCuJjjk4iBZq8fVmN0s1S/Cid11t9dJ
aJUTlNK5qobz/tagEQ9DI4iUw02wrL4oZ2kX7ViJjsG9P5ME268SFFjTWIEjrT
thceelCuCodL65u6SYDddSZM5A1JUbw4fBVqQR2hAnC9FQ
KMc4eRtlLOHXIzpzqXqBQ7ojPwaFG5nQ5kN
/by3m/WwqSONxC7Vs19luerWdfu01DW4ht6Siq8TJ81G6nL5R3XbpoAUhPWgOx
yP0PlEa0TJykhZcIdMJHR3UdwnFJNJmNibHe9zp9JsneY
jBvkT01C5UUYi7pS/ZfHy1jSHLdhY3a3x62ZNM5 yxNNuvOMNu4aVu
61jUgEpuTtUMo9WEAnqqd44yYvQM7bsn
Ep4khrPzTee2UoKca7pCu16zQIYPjbrQIZqUISJrIJwEaTUI9m
4rPjtGzxp780nWDFs5N
mRLJxBPqeZCCdrWDptnxbMsOtGYmXNHCOVow6DrLEhdehBo911dlIq/HSpCWLt
ZA89PSAuagplYI8EW17y7tF4km8QEIZ0Rd52e mK70
XhxmcEmR1GJ0GW056qV8Z1NBjZN5btXetstNo8CbSJJL2D5SG/G7okF/N e
2/e37w9sHmef78ecPNf8FfxDQ0w== </pairs>

```

</txn>

(end)

Receive a PaRes Verification response along with the 3D Secure authentication variables:

```

<txn>
<xid>2C77797CAF.7392B4-T1</xid>
<cavv>MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=</cavv>
  <eci>02</eci>
  <result>0</result>
</txn>

```

- The merchant now has to include the `ssl_eci_ind`, `ssl_3dsecure_value`, and `ssl_xid` values in the transaction request `ccsale` and `ccaauthonly` sent to the converge gateway.

Note: There is a leading (0) in the ECI value returned from Converge in the second step. It must be removed prior to sending it to Converge.

- The website processes Sale or Auth Only collecting the payment information along with the 3D Secure variables obtained, using `processxml.do`. Refer to the [Credit Card Sale \(ccsale\)](#) and [Credit Card Auth Only \(ccaauthonly\)](#) sections for more information.

Below is an example of a `ccsale` request with `ssl_eci_ind`, `ssl_3dsecure_value`, and `ssl_xid` values obtained from the previous steps.

```

xmldata=<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>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>1.00</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>2</ssl_eci_ind>
  <ssl_3dsecure_value>
MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=</ssl_3dsecure_value>
  <ssl_xid>2C77797CAF.7392B4-T1</ssl_xid>
</txn>

```

9. Converge processes the transaction and returns a response to the website displaying the results. A 3D secure transaction will show in the batch with correct ECI indicator.

Transaction Format

The tables below define the transaction request and response data field elements when using `processxml.do` for:

- Card Enrollment
- Payer Authentication
- Verify PaRes

Note: If you are using `process.do` to integrate to Converge, the authentication piece is built in and the authentication variables are passed for you. This section applies to `processxml.do` only.

Step 1: Verify Card Enrollment Request

| Input Field Name | Req? | Description |
|-----------------------------------|------|---|
| <code>ssl_transaction_type</code> | Y | Credit Card 3D Secure enrollment (paenrolled). |
| <code>ssl_merchant_id</code> | Y | Converge ID as provided by Elavon. |
| <code>ssl_user_id</code> | Y | Converge User ID as configured within Converge, case sensitive. |
| <code>ssl_pin</code> | Y | Converge PIN as configured within Converge, case sensitive. |
| <code>ssl_card_number</code> | Y | Credit Card Number as it appears on the credit card. Must be Visa or MasterCard to be eligible. |
| <code>ssl_exp_date</code> | C | Credit Card Expiry Date as it appears on credit card formatted as MMY. |
| <code>ssl_amount</code> | Y | Transaction Sale Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. |

Verify Card Enrollment Response

| Output Field Name | Data Type | Description |
|----------------------|------------|--|
| <code>acs_Url</code> | 1-2048 ANS | Cardholder issuer URL. The website must redirect the cardholder to this website in order to authenticate with 3D Secure. |
| <code>paReq</code> | ANS | Encoded Payer Authentication Request. |

| Output Field Name | Data Type | Description |
|-------------------|------------|---|
| enrolled | 1 A | Indicates whether the card can be authenticated, valid values are: <ul style="list-style-type: none"> N: The Issuer is not participating. Proceed to authorize the transaction with an with ECI set to 07 (Visa or JCB) or 01 (MC) U: Unable to Authenticate: "U" is used whether the Issuer's inability to authenticate the account is due to technical difficulties or business reasons. Proceed to authorize the transaction with an ECI of 07 (Visa or JCB) or 01 (MC) or cancel the transaction. Y: The card is enrolled and authentication is available. Redirect the cardholder to the Issuer site for authentication. After the authentication Process, send request to eMPI to obtain the decrypted ECI/CAVV. |
| md | 0-1024 ANS | Merchant Data. This is a unique reference to identify the 3D Secure request is genuine and can be then matched up with Elavon. |

Step 2: The Payer Authentication Request

| Input Field Name | Req? | Data Type | Description |
|------------------|------|------------|--|
| PaReq | Y | ANS | Encoded Payer Authentication Request. |
| TermUrl | Y | 1-2048 ANS | The merchant URL to which the final reply must be posted. Fully qualified https URL. |
| md | Y | 0-1024 ANS | Merchant Data. Recommend using request id. |

The Payer Authentication Response

| Output Field Name | Data Type | Description |
|-------------------|------------|--|
| PaRes | ANS | Encoded Payer Authentication Response. The PARes message is sent by the ACS in response to the PaReq, regardless of whether authentication is successful. PaRes is a Signed XML String which is Compressed and Base 64 Encoded returned by the Issuer. |
| md | 0-1024 ANS | Merchant Data. |

Step 3: Verify PaRes Request

| Input Field Name | Req? | Description |
|----------------------|------|---|
| ssl_transaction_type | Y | Credit Card 3D PaRes verification (parequest). |
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge User ID as configured within Converge, case sensitive. |
| ssl_pin | Y | Converge PIN as configured within Converge, case sensitive. |
| PaRes | Y | Encoded Payer Authentication key |
| md | | |

Verify PaRes Response

| Input Field Name | Data Type | Description |
|------------------|-----------|---|
| result | | Transaction verification result. |
| xID | 28 ANS | Base 64 Encoded transaction ID. |
| CAVV | 28 ANS | Cardholder Authentication Verification Value. Note: Called UCAF for MasterCard. Universal Cardholder Authentication Field <ul style="list-style-type: none"> Base 64 Encoded (28 characters) |
| eci | 0-2 N | e-Commerce indicator. For a list of possible values, see the ECI Codes section below |

ECI Codes

The tables below define the ECI code field elements for:

- Returned ECI Codes
- ECI Codes to Send to Converge
- Protection Status

Returned ECI Codes

| ECI | Visa | MasterCard |
|-----|---|---|
| 01 | N/A | MCSC has been attempted: Merchant is no longer protected from Chargeback. |
| 02 | N/A | Fully authenticated: There is a liability shift and the merchant is protected from chargebacks. |
| 05 | Fully authenticated: There is a liability shift and the merchant is protected from chargebacks. | N/A |

| ECI | Visa | MasterCard |
|---------|--|---|
| 06 | VbV has been attempted: There is a liability shift and the Merchant is protected from chargebacks. | N/A |
| 07 | Non-VbV transaction: Merchant is no longer protected from Chargebacks. | |
| <blank> | | Non-MCSC transaction: Merchant is no longer protected from Chargebacks. |

ECI Codes to Send to Converge

| ECI (returned by Converge) | ssl_eci_ind | ssl_3DSecure_Value | ssl_xid |
|----------------------------|-------------|--------------------|------------------|
| 01 | 6 | CAVV Optional | Must contain XID |
| 02 | 5 | Must contain CAVV | Must contain XID |
| 05 | 5 | Must contain CAVV | Must contain XID |
| 06 | 6 | CAVV Optional | Must contain XID |
| 07 | 7 | No CAVV | XID optional |
| <blank> | 7 | No CAVV | XID optional |

Transaction Examples

Example 1: process.do

Shown below are the key value pairs for a Sale transaction with 3D Secure enabled – No additional work is needed – authentication is automatic

```

ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_card_number=0000000000000000
ssl_exp_date=1215
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 aprrvl get url=http://www.url.com/cgi-bin/testtran.cgi
    
```

Response below contains the **ECI** indicator of Fully 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>
```

Example 2: processxml.do

Shown below is an example of a ccsale request with `ssl_eci_ind`, `ssl_3dsecure_value`, and `ssl_xid` values obtained from an integrated eMPI solution to indicate this 3D Secure qualified transaction:

```
xmldata=<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>0000000000000000</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>00*****0000</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>
```

MasterPass

Converge offers MasterPass as a service that enables consumers to check out on a merchant website using any MasterPass connected wallet. Cardholders store and manage their payment and shipment information at the MasterPass website, when they are ready to purchase goods, their information will be securely shared with the merchant website, speeding up the checkout process. Converge uses the Elavon eMPI engine to allow processing of MasterPass wallet transactions.

The following transactions types are supported when processing MasterPass transactions:

| Transaction Types | Description |
|-------------------------|-----------------------|
| <code>ccsale</code> | Credit Card Sale |
| <code>ccauthonly</code> | Credit Card Auth Only |

Notes:

- The terminal must be setup under region Canada and market segment e-Commerce.
- The terminal must be setup for MasterPass.
- All card types are supported under the MasterPass wallet.

The following information is supported for MasterPass transactions:

| Field Name | Req? | Description |
|----------------------|------|--|
| ssl_eWallet | C | The wallet identifier , valid values: MasterPass Note: Required for <code>process.do(false)</code> and <code>processxml.do</code> to indicate that the Cardholder has opted to pay using MasterPass. For <code>process.do(true)</code> this value is sent automatically by the payment form when the Cardholder selects to pay using MasterPass. |
| ssl_eWallet_shipping | N | Wallet shipping option to indicate if the shipping address information used is retrieved from the MasterPass website or the merchant's website. Default is N. Valid values: <ul style="list-style-type: none"> • Y = Use the Shipping address stored in MasterPass • N = Use the Shipping address from the merchant website or payment form |
| ssl_product_string | N | Product string. This is the list of products purchased at the merchant website. All products will be displayed as line item on MasterPass site along with the total amount. The product string have to be formatted as following: (Unit Price:: Quantity: Description: Image URL) of the product, several products can be provided, each separated by " ". Converge will send a generic product string if not provided. Example: 10.00::1::shirt::https://merchantwebsite/shirtpic.gif 22.00::1::pants::https://merchantwebsite/pantspic.gif 15.00::1::hat::https://merchantwebsite/hatpic.gif 5.00::1::socks::https://merchantwebsite/sockspic.gif |

Transaction Flow

Using process.do(true)

If you are using `process.do` with the payment form to integrate to Converge, there is no additional work required to use MasterPass, as the authentication piece is built in. The following steps outline the process of sending a MasterPass transaction with the payment form:

1. Cardholder purchases goods/services from the merchant website and clicks on the checkout page.
2. The integrated application/website sends a request using HTTPS, either by the HTTP GET or POST for a Sale or Auth Only request using the payment page.

Shown below are the key value pairs from the header by themselves for a credit card Auth Only transaction indicating that the shipping address as stored in the ewallet is supported.

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=true  
ssl_transaction_type=cacauthonly  
ssl_amount=100.00  
ssl_eWallet_shipping=Y
```

Notes:

- Do not send `ssl_eWallet_shipping` if shipping information is to be collected at the merchant website or the payment page
- `ssl_eWallet_shipping` value is ignored if the Cardholder opted to pay with credit card

-
3. The Merchant website redirects the Cardholder to the Converge payment page.
 4. Cardholder is presented with a choice to enter a credit card number or select the MasterPass wallet by clicking on the MasterPass logo located in the payment page above the card number. The Cardholder selects to pay with the MasterPass wallet.
 5. Cardholder is redirected automatically to the MasterPass website. Cardholder completes the authentication process by logging to their MasterPass wallet and selecting payment information.
 6. Cardholder is redirected back to the payment page.
 7. Payment page displays the card information with billing and shipping information. Cardholder confirms payment.
 8. Converge processes the transaction and returns a response to the website displaying the results.

Using process.do(false)

If you are using `process.do` without the payment form to integrate to Converge, you must send the card information or indicate that the Cardholder is opting to pay using the MasterPass wallet. The MasterPass wallet logo must display in your website.

1. Cardholder purchases goods/services from the merchant website and clicks on the checkout page
2. The integrated application/website sends a Sale or Auth Only request using HTTPS, either by the HTTP GET or POST indicating that the Cardholder is using a MasterPass wallet. Refer to [Credit Card Sale \(ccsale\)](#) or [Credit Card Auth Only \(ccauthonly\)](#) sections for the list of recommended fields.

Shown below are the key value pairs from the header by themselves for a credit card sale transaction indicating that the Cardholder has opted to pay using the MasterPass wallet and the shipping address will be retrieved from the ewallet profile (optional):

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccsale  
ssl_amount=100.00  
ssl_eWallet_shipping=Y  
ssl_eWallet= MasterPass
```

3. Cardholder is redirected automatically to the MasterPass website. Cardholder completes the authentication process by logging to their MasterPass wallet and selecting payment information.
4. Converge processes the transaction and returns a response to the website displaying the results.

Using processxml.do

If you are using `processxml.do` to integrate to Converge, you must provide the choice to the cardholder to pay using a credit card or a MasterPass wallet, the MasterPass logo must be displayed in the website. The following steps outline the process of sending a MasterPass transaction with `processxml.do`:

1. Cardholder purchases goods/services from the merchant website and chooses to pay using MasterPass wallet on the checkout page.
2. The integrated application/website sends an XML formatted Sale or Auth Only request using `processxml.do` using HTTPS, either by HTTP GET or POST indicating that the Cardholder is using a MasterPass wallet and providing a call back URL. Refer to [Credit Card Sale \(ccsale\)](#) or [Credit Card Auth Only \(ccauthonly\)](#) sections for recommended fields.

Shown below is an example of the XML request for a credit card Sale transaction indicating that the Cardholder has opted to pay using the MasterPass wallet and the shipping address will be selected from the ewallet profile (optional):

```

xmldata=<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_transaction_type>ccsale</ssl_transaction_type>
  <ssl_amount>10.00</ssl_amount>
  <ssl_eWallet>MasterPass</ssl_eWallet>
  <ssl_product_string>1.00::1::T-Shirt::Image
  URL</ssl_product_string>
  <ssl_eWallet_shipping>Y</ssl_eWallet_shipping>
  <ssl_callback_url>https://www.merchantwebsite.com/payments/r
  esponse.do</ssl_callback_url>
</txn>

```

3. Converge will send a request to MasterPass.
4. MasterPass responds with a token and a URL.

Shown below is an example of the XML request:

```

<txn>
  <id>rsz9cfRaLPGYU1azyT5mp8Z1016413257</id>
  <oauth_token>40f3dab23aeb9c4f484192acb294d2cd</oauth_token>
  <redirectURL>https://sandbox.masterpass.com/Checkout/Authoriz
  e?oauth_token=40f3dab23aeb9c4f484192acb294d2cd&acceptable_car
  ds=master&checkout_identifier=a4a6x1yrgh71dhgw5g9d31hh3uali82
  xcm&version=v3&suppress_shipping_address=true</redirectURL>
</txn>

```

Note: There is a 15-minute window given for the merchant to use the token and redirect the Cardholder to MasterPass.

5. Merchant redirects the Cardholder Browser to the Authorize URL and uses the Request Token.
6. Cardholder completes the authentication process by logging to their MasterPass wallet and selecting card, billing, and shipping information.
7. MasterPass redirects the Cardholder Back to the Merchant Call back URL with an access token and a session id.

Shown below is an example of the data posted back to the merchant:

```
osessionId=rsz9cfRaLPGYU1azyT5mp8Z1016413257
showform=false
merchantId=EMPI-00012
xid=2C102E7F0D.C234D7-T1
oauth_token=40f3dab23aeb9c4f484192acb294d2cd
oauth_verifier=8cc86efa1f6c3483a23c32a02b7beb33
checkout_resource_url=https://sandbox.api.mastercard.com/online/
v3/checkout/2036737
```

8. Merchant uses the session id and sends the access token back to Converge.

Shown below is an example of the XML request:

```
<txn>
  <osessionId>rsz9cfRaLPGYU1azyT5mp8Z1016413257</osessionId>
  <merchantId>EMPI-00012</merchantId>
  <xid>2C102E7F0D.C234D7-T1</xid>
  <oauth_token>40f3dab23aeb9c4f484192acb294d2cd </oauth_token>
  <oauth_verifier>8cc86efa1f6c3483a23c32a02b7beb33</oauth_veri
fier>
  <checkout_resource_url>
https://sandbox.api.mastercard.com/online/v3/checkout/203673
7</checkout_resource_url>
</txn>
```

9. Converge sends the access token to MasterPass through and obtains the credit card number, expiration date, billing and shipping information.
10. Converge processes the transaction and returns a response to the merchant website displaying the results.

Transaction Examples

Example 1: process.do

Shown below is an example of a `ccsale` request with wallet indicator, shipping will be provided by the MasterPass wallet along with the card information:

```
ssl_amount" value=5.00
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccsale
ssl_show_form" value=false
ssl_eWallet_shipping=Y
ssl_eWallet= MasterPass
ssl_invoice_number=123-ABC
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

Example 2: process.do

Send a ccauthonly request with wallet indicator, shipping will be provided by the merchant:

```
ssl_amount" value=5.00
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccaauthonly
ssl_show_form" value=false
ssl_invoice_number=123-ABC
ssl_eWallet_shipping=N
ssl_eWallet= MasterPass
ssl_ship_to_address1=999 Main
ssl_ship_to_last_name=Doe
ssl_ship_to_first_name=Jane
ssl_ship_to_city=Any City
ssl_ship_to_zip=99999
ssl_ship_to_state=GA
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl method=REDG
```

Example 3: process.do

Send a Sale transaction with the payment page, billing and shipping will be provided by the MasterPass wallet along with the card information– No additional work is needed

```
ssl_amount" value=5.00
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccsale
ssl_show_form" value=true
ssl_eWallet_shipping=Y
ssl_invoice_number=123-ABC
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl method=REDG
```

Dynamic Currency Conversion (DCC)

Converge uses Elavon conversion tools to process Dynamic Currency Conversion (DCC) for merchants. The merchant is pricing in USD or CAD, but when the card is entered and is determined to be DCC eligible, Cardholder is then given a choice to pay with their currency (foreign); this agreed currency becomes the transaction currency, regardless of the merchant's pricing currency. Cardholder accepts to pay with their local currency and the conversion rates associated with it:

- Merchant price, sell, authorize, settle and is funded in one single currency (The merchant currency USD or CAD)
- Cardholders are prompted for a choice to pay in their local currency (foreign)

The following transactions types are supported when processing DCC transactions:

| Transaction Types | Description |
|-------------------|---------------------------|
| ccsale | Credit Card Sale |
| ccauthonly | Credit Card Auth Only |
| ccforce | Credit Card Force |
| cccredit | Credit Card Credit/Refund |

Notes:

- The terminal must be setup for DCC.
- DCC is supported with MasterCard and Visa transactions only. Other card types will process as normal and will not trigger DCC processing.
- The conversion is triggered when the card is swiped or entered, and rates will be presented for any of the following currencies: US Dollar, Canadian Dollar, Pound Sterling, Australian Dollar, Euro and Japanese.

The process of dynamic currency is to retrieve the rates and pass the following DCC Transaction variables:

| Field Name | Req? | Description |
|-----------------------|------|--|
| ssl_txn_currency_code | Y | The cardholder's Billing Currency. |
| ssl_markup | Y | The Mark-up applied to the Reference Exchange Rate when calculating the Exchange Rate used to convert from the merchant's Pricing Currency to the cardholder's Billing Currency. |
| ssl_conversion_rate | Y | Exchange rates applied to the conversion of the transaction amount. |
| ssl_cardholder_amount | Y | The transaction amount in the cardholder's Billing Currency |

Transaction Flow

Using process.do

If you are using `process.do` to integrate to Converge, there is no additional work required to process DCC, as the conversion piece is built and Cardholders are presented with the choice to pay in their currency, the DCC variables are then passed for you. The following steps outline the process of sending a DCC transaction with `process.do`:

1. The website or Point of Sale (POS) application processes Sale, Auth Only, or Force transactions and collects the payment information using `process.do`. Refer to the [Credit Card Sale \(ccsale\)](#), [Credit Card Auth Only \(ccaauthonly\)](#), or [Credit Card Force \(ccforce\)](#) sections for more information.
2. Cardholder enters or swipes a Visa or MasterCard card at the checkout.
3. Converge will determine if the card is a foreign card. If the card is determined to be foreign the website or POS is redirected automatically to the DCC choice page
4. The DCC choice page displays the rate, the markup percentage, the merchant amount, and the card holder amount.
5. Cardholder decides in what currency the transaction is processed in.
6. Converge processes the transaction and returns a response to the website displaying the results with conversion rates and amounts agreed on.

Notes:

- There is a 15-minute window given to the Cardholder to opt for an option
 - If you are using `process.do` and `ssl_result_format` field is set to ASCII, the flow of the transaction will be similar to `processxml.do` outlined below.
-

Using processxml.do

If you are using `processxml.do` to integrate to Converge, you must provide the choice to the cardholder to accept or decline the DCC service and display all relevant information to make an informed decision. The following steps outline the process of sending a DCC transaction with `processxml.do`:

1. The website or POS application processes Sale, Auth Only, or Force transactions and collects the payment information using `process.do`. Refer to the [Credit Card Sale \(ccsale\)](#), [Credit Card Auth Only \(ccaauthonly\)](#), or [Credit Card Force \(ccforce\)](#) sections for more information.
2. Cardholder enters or swipes a Visa or MasterCard card at the checkout.
3. Converge will determine if the card is a foreign card

4. Converge will return a decision response providing a transaction ID, rates, markup, and currency information.
5. The website or POS application displays the option to the cardholder with the information returned by Converge
6. The Cardholder makes a choice at the website or POS application :
 - a) If the Cardholder opted to process in home currency, the website or POS application sends the Transaction ID for that transaction with DCC option value of Y.
 - b) If the Cardholder opted to process in merchant currency, the website or POS application sends the Transaction ID for that transaction with DCC option value of N.
7. Converge processes the transaction and returns a response to the website displaying the results with conversion rates and amounts agreed on.

Note: There is a 15-minute window given to allow the website or POS application to send the option.

Transaction Examples

Example 1: process.do

Send a ccsale request

```
ssl_amount" value=5.00
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccsale
ssl_show_form" value=false
ssl_invoice_number=123-ABC
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

Receive a DCC decision Page

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

| SALE - Dynamic Currency Confirmation | |
|--------------------------------------|--------|
| Transaction Currency | EUR |
| Conversion Rate | .76373 |
| Markup(%) | 3.25 |
| Total (USD) | 1.00 |
| Total (EUR) | 0.76 |

Please charge my purchase in my home currency

Do not charge me in my home currency; charge my purchase in the foreign currency

All currency choices are final.

Example 2: `processxml.do`

Send a ccsale request

```
xmldata=<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>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>1.00</ssl_amount>
</txn>
```

Receive a DCC decision response

```

xmldata=<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> <dccooption>
<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>
</dccooption>
</txn>

```

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

```

xmldata=<txn>
<ID>ekU9j0L0iFO9m9FELAqK8E6</ID>
<dccooption>Y</dccooption>
</txn>

```

Receive a final response and print receipt

```

xmldata=<txn>
<ssl_card_number>00*****0000</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>

```

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

```

xmldata=<txn>
<ID>iQ4AezhcjmJznh3BYLZ0-W9</ID>
<dccooption>N</dccooption>
</txn>

```

Receive a final response

```

xmldata=<txn>
<ssl_card_number>00*****0000</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>

```

Send a ccsale with tip request

```

xmldata=<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_amount>10.00</ssl_amount>
<ssl_tip_amount>2.00</ssl_tip_amount>
<ssl_track_data>%B00*****0000^ELAVONTEST/TESTCARD^151290154321
3961456789123456789001?;00*****0000=151290154321396?</ssl_trac
k_data>
</txn>

```

Receive a response showing Cardholder base, Tip and Total amounts after customer opted to pay in DCC

```
<txn>
<ssl_approval_code>CVI045</ssl_approval_code>
<ssl_cvv2_response/>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_server/>
<ssl_cardholder_amount>9.52</ssl_cardholder_amount>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_markup>3.25</ssl_markup>
<ssl_tip_amount>2.00</ssl_tip_amount>
<ssl_base_amount>10.00</ssl_base_amount>
<ssl_amount>12.00</ssl_amount>
<ssl_txn_id>AA4843B-893BE162-0F1A-4DF5-9076-
D93CB9421914</ssl_txn_id>
<ssl_result>0</ssl_result>
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_cardholder_tip_amount>1.59</ssl_cardholder_tip_amount>
<ssl_cardholder_base_amount>7.94</ssl_cardholder_base_amount>
<ssl_cardholder_currency>EUR</ssl_cardholder_currency>
<ssl_txn_time>10/08/201211:57:25PM</ssl_txn_time>
<ssl_conversion_rate>.79362</ssl_conversion_rate>
</txn>
```

Update tip in Cardholder amount (Cardholder leaves a tip on a DCC transaction)

```
xmldata=<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>CCUPDATE TIP</ssl_transaction_type>
<ssl_cardholder_tip_amount>3.00</ssl_cardholder_tip_amount>
<ssl_txn_id>AA4843B-893BE162-0F1A-4DF5-9076-
D93CB9421914</ssl_txn_id>
</txn>
```

Receive a final response

```
<txn>
<ssl_result_message>SUCCESS</ssl_result_message>
<ssl_tip_amount>3.78</ssl_tip_amount>
<ssl_base_amount>10.00</ssl_base_amount>
<ssl_amount>13.78</ssl_amount>
<ssl_cardholder_tip_amount>3.00</ssl_cardholder_tip_amount>
<ssl_cardholder_base_amount>7.94</ssl_cardholder_base_amount>
<ssl_cardholder_amount>10.94</ssl_cardholder_amount>
<ssl_cardholder_currency>EUR</ssl_cardholder_currency>
<ssl_conversion_rate>.79362</ssl_conversion_rate>
<ssl_txn_id>AA4843B-893BE162-0F1A-4DF5-9076-
D93CB9421914</ssl_txn_id>
<ssl_txn_time>10/08/201211:57:25PM</ssl_txn_time>
</txn>
```

Receipt Requirements

Receipt Requirements for VISA

All Environments (Face-to-Face, MOTO, 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.

Receipt Requirements for MasterCard**All Environments (Face-to-Face, MOTO, 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 |
|--|--|
| <p style="text-align: center;">Header1 FRIENDLY TERMINAL XXXXXXXXXXXXXXXXXXXX</p> <p>Date: 11/04/2010 11:07:30 AM</p> <p style="text-align: center;">CREDIT CARD SALE</p> <p>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</p> <p>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}.</p> <p>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.</p> <p>I AGREE TO THE TRANSACTION RECEIPT INFORMATION BY MARKING THE ACCEPT BOX BELOW.</p> <p>[] I ACCEPT</p> <p>X _____ JPY STANDARD VI</p> <p>Trailer1</p> <p>Merchant Copy</p> | <p style="text-align: center;">Header1 FRIENDLY TERMINAL XXXXXXXXXXXXXXXXXXXX</p> <p>Date: 11/04/2010 11:07:30 AM</p> <p style="text-align: center;">CREDIT CARD SALE</p> <p>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</p> <p>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}.</p> <p>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.</p> <p>Trailer1</p> <p>Customer Copy</p> |

Multi-Currency Conversion (MCC)

Multi-Currency processing allows the merchant to sell their products and services in a variety of currencies but receive their funding from Elavon in just one currency. Multi-Currency payment solutions help merchants offer enhanced customer service, and attract more international customers.

This feature is beneficial to those merchants who have websites targeted to a particular audience or demographic in a country other than their own. Examples:

- A US merchant has a website targeted to Europe, which displays prices and processes payments in Euros (EUR), payments are then cleared in a US bank account in USD.
- A Canadian merchant has a website targeted to customers from Australia; it processes transactions in Australian Dollars (AUD) and deposits the payments to a Canadian domiciled bank account in CAD.

Elavon supports almost a hundred different transaction currencies (refer to the [ISO Currency Codes](#) section) for both MasterCard and Visa payments. Transaction amounts are converted from the submission currency to the funding currency on your merchant account, utilizing the exchange rates provided by Elavon's designated currency exchange desk provider.

The following transactions types are supported when processing Multi-Currency transactions:

| Transaction Types | Description |
|-------------------|--|
| ccsale | Credit Card Sale |
| ccauthonly | Credit Card Auth Only |
| ccforce | Credit Card Force |
| cccredit | Credit Card Return/Credit |
| ccimport | A Batch file of credit card transactions |

Notes:

- The terminal must be setup for Multi-Currency.
 - Once a terminal is setup for Multi-Currency, the terminal can process by default the following currencies: US Dollar (USD), Canadian Dollar (CAD), Pound Sterling (GBP), Australian Dollar (AUD), Euro (EUR) and Japanese (YEN). You can add or remove currencies in the Converge user interface.
 - Multi-Currency is supported with MasterCard and Visa transactions only. Other card types cannot be sent with a transaction other than merchant currency: USD or CAD.
 - Manual settlement is not allowed for those terminals setup for Multi-Currency.
-

Transaction Flow

To process a transaction as Multi-Currency, simply pass the 3 character ISO currency code in the `ssl_transaction_currency` field along with the `ssl_amount`, if the transaction currency is not specified, it is assumed to be the same as the merchant domestic currency (USD or CAD).

The following steps outline the process of sending a Multi-Currency transaction:

1. The website/POS application quotes the customer a price in the variety of currencies the merchant accepts.
 - Currency Conversion is done by the Merchant on the Merchant Website
 - Merchant price, sell, authorize, and settle in variety of currencies
2. Cardholder keys or swipes a Visa or MasterCard card at the checkout.
3. The integrated application/website processes Sale, Auth Only, Refund, or Force transactions and collects the payment information using `process.do` or `processxml.do` or a batch import using `processBatch.do`. Refer to [Credit Card Sale \(ccsale\)](#), [Credit Card Auth Only \(ccautionly\)](#), or [Credit Card Force \(ccforce\)](#), [Credit Card Return Credit \(cccredit\)](#), or [Credit Card Batch Import \(ccimport\)](#) sections for more information.
4. The integrated application/website sends the transaction with the amount and the transaction currency along with the transaction data using HTTPS, either by the HTTP GET or POST. Shown below are the key value pairs from the header by themselves for a credit card sale transaction processed in EUR:

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccsale  
ssl_card_number=0000000000000000  
ssl_exp_date=1214  
ssl_amount=100.00  
ssl transaction currency=EUR
```

5. Converge processes the transaction and returns a response to the website displaying the results with the transaction currency.

Notes:

- Merchant is funded in one single currency (The merchant currency USD or CAD).
- Do not send transaction currency if adjustment to the original transaction is needed; ccomplete, ccvoid, ccupdatetip or ccreturn will carry the original transaction currency.
- When specifying amounts, be sure to submit the correct number of decimal places for the transaction currency. All currencies support two exponents (decimals,) other than the following:

| Currency Code | Currency Name | Exponents |
|---------------|-------------------------|-----------|
| JPY | Japanese Yen | 0 |
| KRW | South Korean Won | 0 |
| XAF | Gabon Franc | 0 |
| XOF | Ivory Coast Franc | 0 |
| XPF | French Polynesian Franc | 0 |
| BHD | Bahraini Dinar | 3 |
| JOD | Jordanian Dinar | 3 |
| KWD | Kuwaiti Dinar | 3 |
| LYD | Libyan Dinar | 3 |
| OMR | Omani Rial | 3 |
| TND | Tunisian Dinar | 3 |

- For Japanese Yen, there are no currency exponents after the decimal place. Any numbers included after the decimal place will be ignored.
 - For the Bahraini Dinar, there are 3 possible digits after the decimal place. Your website may wish to take advantage of the precision this audience of customers will expect. However, currently the Converge will accommodate only 2 decimals. If you submit three, it will automatically round your transaction to two decimals. (for example: 1.235 will round to 1.23 and 1.236 will round to 1.24).
-

Transaction Examples

Example 1: process.do

Send a ccsale request

```
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccsale
ssl_amount" value=200
ssl_transaction_currency=JPY
ssl_show_form =false
ssl_invoice_number=123-ABC
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl receipt aprvl get url=http://www.website.com/approval.asp
```

Example 2: processxml.do

The following XML code example demonstrates the initiation of a Sale transaction with currency:

```
xmldata=<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>0000000000000000</ssl_card_number>
<ssl_exp_date>1215</ssl_exp_date>
<ssl_amount>200</ssl_amount>
<ssl_transaction_currency>JPY</ssl_transaction_currency>
</txn>
```

Receive a final response and print receipt

```
xmldata=<txn>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_amount>200</ssl_amount>
  <ssl_transaction_currency>JPY</ssl_transaction_currency>
  <ssl_result>0</ssl_result>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_txn_id>AA47AE-BD32FF54-BE12-4046-98E9-
  F78E9D75212D</ssl_txn_id>
  <ssl_approval_code>N05465</ssl_approval_code>
  <ssl_txn_time>11/19/2013 12:02:29 AM</ssl_txn_time>
</txn>
```

Tokenization

The following section provides the guidelines on how to obtain tokens and use them to process transactions with your integrated application.

Tokenization is a powerful security feature that allows a merchant to support all of their existing business processes that require card data without the risk of holding card data and without any security implications therefore protecting the customers' confidential information; because tokens are useless to criminals, they can be saved by the merchant as they do not represent any threat, the liability and costs associated with PCI compliance is substantially reduced and the risk of storing sensitive data is eliminated. Tokenization applies to credit card and gift card.

Merchants set up for the tokenization service receive responses that include a token. The token generated is not linked to a specific transaction but to a specific card number and the token generated for that transaction will be identical for every use of that card number and merchant. Furthermore, you can generate a token and save the token with associated information in the Card Manager.

Card manager enables merchants to tokenize and store their customers' sensitive payment information on our secure servers, simplifying their PCI DSS compliance as well as the payments process for returning customers. The Card Manager will allow you access to all tokens that have been previously generated from your integrated application or directly added to your integrated application. The Card Manager can be only accessed directly from the user interface and will give you the ability to search for a token previously added, add a card number, and process transactions.

Use Cases may include the following scenarios:

1. Generate tokens using credit or gift card numbers:
 - Requesting token only.
 - Requesting token after validating the card.
 - Processing a transaction and requesting a token.
2. Generate tokens using existing recurring transactions:
 - Requesting token only.
 - Processing a recurring/installment transaction and requesting a token.
3. Store tokens from credit cards in the Card Manager after token generation.
4. Process transactions using tokens:
 - Providing the token, expiration date and billing information.
 - Providing the token only and Converge will use the stored information in the Card Manager to send the expiration date and billing information for you. Applicable to credit cards only at this time.

Notes:

- Tokens replace credit or gift card numbers.
 - The terminal must be enabled to accept tokenization.
 - Tokens are unique for each merchant, for example: the same card will produce a different token for each merchant.
 - Tokens match the format of the initiating PAN and do not overlap major brand (Visa, MC, AMEX, and Discover) BIN ranges (first digit is 0-2 or 7-9) and share last four digits with corresponding PAN.
 - Merchants with multiple terminals sharing tokenization domains will receive the same token for a unique card and the token can be used across their stores if they wish to do so.
 - Merchants may supply the token in place of card information in any subsequent transaction.
 - Card Manager can be accessed from the user interface.
 - Tokenization is supported for both credit cards and gift cards. Adding a token to the Card Manager using integration is supported only when sending a *Credit Card Number*.
-

Generating Tokens

There are two ways to generate a token:

1. Performing an authorization (`process.do`, `processxml.do`, or `processBatch.do`) either:
 - An authorization with the **Generate Token** indicator of Y by itself will authorize the transaction and generate a token. You must pass either a card number for a hand-keyed transaction, a track data for a swiped transaction or recurring ID for recurring/installment transactions.
 - An authorization with the **Add to Card Manager** Indicator of Y by itself will authorize the transaction, generate a token, and add the token to the Card Manager. Adding to the Card Manager is applicable to credit cards *only*. To add a gift card you must use the user interface.

The following transactions are supported when requesting a token at the authorization time:

| Transaction Types | Description |
|------------------------------|--|
| <code>ccsale</code> | Credit Card Sale |
| <code>ccauthonly</code> | Credit Card Auth Only |
| <code>ccforce</code> | Credit Card Force |
| <code>ccccredit</code> | Credit Card Return/Credit |
| <code>ccbalinquiry</code> | Credit Card Balance Inquiry |
| <code>ccimport</code> | A Batch file of credit card transactions |
| <code>ccrecurringsale</code> | Submit Recurring Payment |
| <code>ccinstallsale</code> | Submit Installment Payment |
| <code>egcactivation</code> | Gift Card Activation |
| <code>egcsale</code> | Gift Card Sale |
| <code>egcbalinquiry</code> | Gift Card Balance Inquiry |
| <code>egcreload</code> | Gift Card Reload |
| <code>egccredit</code> | Gift Card Credit |

All transactions requests sent to Converge must pass the following basic information:

| Field Name | Req? | Description |
|------------------|------|---|
| ssl_track_data | C | Required for swiped transactions. The raw track I and/or II data from the magnetic strip on the credit or gift 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 drack data. Raw track data cannot be stored under any circumstance. Expiration date is included with track data. First and last name of the Cardholder is also included with the Track I data |
| ssl_card_number | C | Required for hand-keyed transactions. Credit Card Number or Gift Card Number as it appears on the card. |
| ssl_exp_date | C | Required for hand-keyed transactions. Expiration date. |
| ssl_recurring_id | C | Required for recurring or installment transactions. The ID number of the recurring record or installment record to be submitted for payment. This value was returned in the recurring ID when the original recurring record was added for recurring or installment ID when the original installment was added for installment. Alphanumeric. |
| ssl_amount | Y | Transaction Amount. |
| ssl_get_token | C | Generate Token indicator must be sent in order to generate tokens only. Valid values: Y (generate a token), N (do not generate token). Defaulted to N. |
| ssl_add_token | C | Add to Card Manager indicator, must be sent in order to indicate that the token needs to generated and added to the Card Manager. Valid values: Y (add token) , N (do not add token). Defaulted to N. Add token to card manager is not supported with ccimport, it is also not supported with Gift Card transactions. |
| ssl_first_name | C | Required with Add to Card Manager indicator. Cardholder first name. |
| ssl_last_name | C | Required with Add to Card Manager indicator. Cardholder last name. |
| ssl_avs_zip | C | Required with Add to Card Manager indicator. Customer ZIP Code. |
| ssl_avs_address | C | Required with Add to Card Manager indicator. Customer Billing Address. |

2. Performing a token generation request (`process.do`, `processxml.do`, or `processBatch.do`).

The following transactions are supported when requesting a token without authorization:

| Transaction Types | Description |
|----------------------------|---|
| <code>ccgettoken</code> | Credit Card Generate Token |
| <code>cctokenimport</code> | A Batch file of Credit Card information |
| <code>egcgettoken</code> | Gift Card Generate Token |

Token generation can request account verification along with the token to be added to the Card Manager.

- A Token generation request by itself will generate a token from a credit or gift card number or recurring ID.
- A Token generation request with card verification flag will validate if the credit card is valid along with CVV and AVS then generate the token.
- A token generation request with the **Add to Card Manager** indicator will generate a token from a card number or recurring ID and add it the token to the card manager, this functionality is not available with gift card transactions using the integration. To add a gift card to the card manager you must use the user interface.
- A token generation request with **Account Verification** indicator will perform a check on the card along with CVV and AVS, if supplied, prior to generating a token from a *Credit Card Number*.
- A token generation request with **Card Verification Flag** and **Add to Card Manager** will validate if the *Credit Card Number* is valid along with CVV and AVS , generate the token, then add it to the card manager.

All transactions with token generation requests sent to Converge must pass the following basic information:

| Field Name | Req? | Description |
|-------------------------------|------|---|
| <code>ssl_card_number</code> | C | Required for hand-keyed transactions. Credit Card Number or Gift Card Number as it appears on the card. |
| <code>ssl_exp_date</code> | C | Required for hand-keyed transactions. Expiration date. |
| <code>ssl_recurring_id</code> | C | Required for recurring or installment transactions. The ID number of the recurring record or installment record to be submitted for payment. This value was returned in the recurring ID when the original recurring record was added for recurring or installment ID when the original installment was added for installment. Alphanumeric. |

| Field Name | Req? | Description |
|-----------------|------|---|
| ssl_verify | N | Verification indicator must be sent in order to verify the card for CVV and AVS checks. Applicable to credit cards only. |
| ssl_add_token | N | Add to Card Manager indicator, must be sent in order to indicate that the token needs to be added to the Card Manager. Applicable to credit cards only. |
| ssl_first_name | C | Required with Add to Card Manager indicator. Cardholder first name. Applicable to credit cards only. |
| ssl_last_name | C | Required with Add to Card Manager indicator. Cardholder last name. Applicable to credit cards only. |
| ssl_avs_zip | C | Required with Add to Card Manager indicator, optional for verification. Customer ZIP Code. Applicable to credit cards only. |
| ssl_avs_address | C | Required with Add to Card Manager indicator, optional for verification. Customer Billing Address. Applicable to credit cards only. |
| ssl_cvv2cvc2 | N | CVV2/CVC/CID in the field. Optional for verification. Applicable to credit cards only. |

All token generation requests with or without authorization will return the following fields:

| Field Name | Description |
|------------------------|--|
| ssl_token | The token value generated from the card number. This value can be stored and used as a substitute for a card number. |
| ssl_token_response | Outcome of the token generation. This value will be a SUCCESS if a token has been generated. Other possible returned values are FAILURE, Action Not Permitted, Invalid Token, Not Permitted, Acct Verification Failed. |
| ssl_add_token_response | Outcome of the Add to Card Manager request, examples: Card Added, Card Updated, Not Permitted, FAILURE - First Name - is required. |

| Field Name | Description |
|--------------------|--|
| ssl_result | Outcome of the transaction if the token generation was requested with verification or with an authorization. A response that contains <code>ssl_result</code> of 0 represents an Approved transaction. A response containing any other value for <code>ssl_result</code> 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_avs_response | The Address Verification Response Code if an AVS was passed (refer to the AVS Response Codes section for a complete list of AVS response codes). |
| ssl_cvv2_response | The Card Verification Response Code if a CVV was passed (refer to the CVV2/CVC2 Response Codes section for a complete list of CVV/CVC2 response codes) |

Processing Transactions Using Tokens

The following transactions are supported when processing transactions using tokens for a *Credit Card Number*:

| Transaction Types | Description |
|-------------------|--|
| ccsale | Credit Card Sale |
| ccautionly | Credit Card Auth Only |
| ccforce | Credit Card Force |
| cccredit | Credit Card Return/Credit |
| ccbalinquiry | Credit Card Balance Inquiry |
| ccimport | A Batch file of credit card transactions |
| egcactivation | Gift Card Activation |
| egcsale | Gift Card Sale |
| egcbalinquiry | Gift Card Inquiry |
| egcreload | Gift Card Reload |
| egccredit | Gift Card Credit |

All credit and gift card transactions sent with tokens to Converge must pass the following basic cardholder information:

| Field Name | Req? | Description |
|--------------|------|---|
| ssl_token | Y | Credit or Gift Card Token previously generated from a card number. |
| ssl_exp_date | C | Expiration date. Required if the token is not in the Card Manager. If the token has been previously stored in the Card Manager, expiration date should not be sent. |
| ssl_amount | Y | Transaction Amount. Number with 2 decimal places. The authorized amount passed on request should not contain the tip amount, tips must be passed separately. The total authorized amount will be calculated during the authorization if tip is provided. For example: 1.00. |

Notes:

- The Card Manager feature is not available with gift card transactions using the integration. To add a gift card to the Card Manager you must use the user interface.
 - When tokens are stored in the Card Manager, the token number can be sent alone , Converge will grab the Billing information and the Cardholder first and last name and send them with the transaction.
 - The more cardholder information passed during authorization, the better. Tokens are substitute to card numbers and adhere to the same rules as a hand keyed transactions. Refer to [Credit Card Sale \(ccsale\)](#), [Credit Card Auth Only \(ccautionly\)](#), or [Credit Card Force \(ccforce\)](#), [Credit Card Return Credit \(ccccredit\)](#), [Credit Card Batch Import \(ccimport\)](#), [Gift Card Activation \(egcativation\)](#), [Gift Card Sale/Redemption \(egcsale\)](#), [Gift Card Balance Inquiry \(egcbalinquiry\)](#), [Gift Card Replenishment/Reload \(egcreload\)](#), or [Gift Card Credit \(egccredit\)](#) sections for more information.
 - Tokens are substitute to card numbers and adhere to the same rules as a hand keyed transactions
-

Managing Stored Tokens

Once stored, tokens from a *Credit Card* can be located, modified or deleted. The following transactions are used to manage stored tokens in the card manager:

| Transaction Types | Description |
|-------------------|--------------------------|
| ccquerytoken | Credit Card Token Query |
| ccupdatetoken | Credit Card Token Update |
| ccdeletetoken | Credit Card Token Delete |

Note: Only those tokens that belong to a credit card number can be managed using the integration. Tokenized gift cards cannot be added, modified, or deleted using integration.

Transaction Flow

1. Submit a transaction request to generate a token or a standard authorization with the **Generate Token** indicator by using HTTPS, either by the HTTP GET or POST with the values shown in the API Reference Chapter in this guide in order to generate a token in the response. If you wish to store the token with billing information in the card manager, you must set the `ssl_add_token` to Y. The request can be submitted on a single transaction or a batch file of transactions.

Shown below are the key value pairs from the header by themselves for a Generate Token request only:

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccgettoken  
ssl_card_number=0000000000000000  
ssl_exp_date=1215  
ssl_result_format=HTML  
ssl_receipt_decl_method=REDG  
ssl_error_url=http://www.url.com/cgi-bin/testtran.cgi  
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
```

Shown below are the key value pairs from the header by themselves for a Generate Token request with card verification:

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccgettoken  
ssl_card_number=0000000000000000  
ssl_exp_date=1215  
ssl_verify=Y  
ssl_cvv2cvc2=123  
ssl_avs_address=123 Main St  
ssl_avs_zip=01234  
ssl_result_format=HTML  
ssl_receipt_decl_method=REDG  
ssl_error_url=http://www.url.com/cgi-bin/testtran.cgi  
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
```

Shown below are the key value pairs from the header by themselves for a Generate Token request with card verification and **Add to Card Manager** Indicator:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_transaction_type=ccgettoken
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_verify=Y
ssl_cvv2cvc2=123
ssl_avs_address=123 Main St
ssl_avs_zip=01234
ssl_add_token=Y
ssl_first_name=John
ssl_last_name=Doe
ssl_result_format=HTML
ssl_receipt_decl_method=REDG
ssl_error_url=http://www.url.com/cgi-bin/testtran.cgi
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
```

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

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_transaction_type=ccsale
ssl_card_number=0000000000000000
ssl_exp_date=1214
ssl_cvv2cvc2=123
ssl_avs_address=123 Main St
ssl_avs_zip=01234
ssl_get_token=Y
ssl_amount=1.00
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
```

Shown below are the key value pairs from the header by themselves for a sale transaction requesting a token and **Add to Card Manager** Indicator:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_transaction_type=ccsale
ssl_card_number=0000000000000000
ssl_exp_date=1214
ssl_cvv2cvc2=123
ssl_avs_address=123 Main St
ssl_avs_zip=01234
ssl_get_token=Y
ssl_add_token=Y
ssl_first_name=John
ssl_last_name=Doe
ssl_amount=1.00
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
```

2. When Converge 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.

Note: User ID (`ssl_user_id`) and PIN (`ssl_pin`) are case sensitive.

3. If the supplied information is invalid, an error is returned that states that the information is invalid. If the data is valid, Converge 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.

4. 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 Converge to send all the errors that are encountered. Any response that is not approved or declined will be sent to the URL specified.

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
```

6. 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:

Shown below are the key value pairs from a successful generate token transaction:

```
ssl_result=0
ssl_token=8848842356063003
ssl_token_response=SUCCESS
ssl card number=00*****0000
```

Shown below are the key value pairs from a successful generate token with card verification transaction:

```
ssl_result=0
ssl_token=8848842356063003
ssl_token_response=SUCCESS
ssl_card_number=00*****0000
ssl_result_message=APPROVAL
ssl_txn_id=AA47AE-4B3FEED3-055E-4D4B-A67B-2AFBC235B925
ssl_approval_code=CVI368
ssl_cvv2_response=A
ssl_avs_response=M
ssl_txn_time=01/14/2014 11:13:51 AM
```

Shown below are the key value pairs from a successful generate token with card verification transaction and Add to Card Manager:

```
ssl_result=0
ssl_add_token_response=Card Added
ssl_token=8848842356063003
ssl_token_response=SUCCESS
ssl_card_number=00*****0000
ssl_result_message=APPROVAL
ssl_txn_id= AA47AE-B183DD7B-209D-4A8E-8CB6-749E1B750944
ssl_approval_code=N46032
ssl_cvv2_response=A
ssl_avs_response=M
ssl_txn_time=02/15/2014 13:12:35 AM
ssl_avs_address=123 Main St
ssl_avs_zip=01234
ssl_first_name=John
ssl_last_name=Doe
```

Shown below are the key value pairs from an approved sale transaction:

```
ssl_card_number=00*****0000
ssl_exp_date=0515
ssl_amount=1.00
ssl_get_token=Y
ssl_token=8138827982163003
ssl_token_response=SUCCESS
ssl_result=0
ssl_result_message=APPROVAL
ssl_txn_id=AA776F-1981E7F3-907D-44F1-93BC-48D52F8CC2DF
ssl_approval_code=N44032
ssl_cvv2_response=A
ssl_avs_response=M
ssl_account_balance=0.00
ssl_txn_time=02/17/2014 08:44:43 AM
```

7. By specifying the `http://www.url.com/cgi-bin/testtran.cgi` url in the `ssl_receipt_decl_get_url` field for the redirect for the transaction above, the following values are returned for when token has failed to be returned:

```
ssl_result=1
ssl_token=
ssl_token_response=FAILURE
```

8. Store the value of `ssl_token` obtained from successful transaction to use for any consecutive requests using HTTPS, either by the HTTP GET or POST method.

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

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccsale  
ssl_token=8128827982163003  
ssl_exp_date=1215  
ssl_avs_zip=70004  
ssl_avs_address=123 Main St  
ssl_first_name=John  
ssl_last_name=Doe  
ssl_amount=1.00  
ssl_result_format=HTML
```

Shown below are the key value pairs from the header by themselves for a credit card sale transaction using a token previously stored in the Card Manager.

```
ssl_merchant_id=xxxxxxx  
ssl_user_id=xxxxxxx  
ssl_pin=xxxxxxx  
ssl_show_form=false  
ssl_transaction_type=ccsale  
ssl_token=8128827982163003  
ssl_amount=1.00  
ssl_result_format=HTML
```

Transaction Examples

Example 1: process.do

In this example, the HTML code demonstrates the initiation of a token generation request with minimal information, the Cardholder will have to enter the card number and expiration date in the payment form and request the card to be added on file for later use:

```
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccgettoken
ssl_show_form=true
ssl_invoice_number=123-ABC
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

Example 2: process.do

The following HTML code show an example of a request to generate a token after performing CVV and AVS check on card. Once the verification is preformed successfully a token will be generated:

```
ssl_merchant_id" value=my_virtualmerchant_id
ssl_user_id" value=my_user_id
ssl_pin" value=my_pin
ssl_transaction_type=ccgettoken
ssl_verify=Y
ssl_card_number=0000000000000000
ssl_exp_date=1215
ssl_cvv2cvc2=123
ssl_avs_address=123 Main St
ssl_avs_zip=01234
ssl_show_form=false
ssl_invoice_number=123-ABC
ssl_result_format" value="HTML
ssl_receipt_decl_method=REDG
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

Approved Receipt

This generates a receipt that includes the following code for a successful token generation with verification:

```
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="M">
  <input type="hidden" name="ssl_avs_response" value="A">
  <input type="hidden" name="ssl_token" value="8138827982163003">
  <input type="hidden" name="ssl_token_response" value="SUCCESS">
</form>
ssl_receipt_decl_get_url=http://www.website.com/decline.asp
ssl_receipt_apprvl_method=REDG
ssl_receipt_apprvl_get_url=http://www.website.com/approval.asp
```

Decline Receipt

The result could be a form that includes the following code for a declined transaction due to verification failure; a token will not be generated:

```
<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_token" value="">
  <input type="hidden" name="ssl_token_response" value="FAILURE">
  <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_email" value=" test@test.com">
  <br>
  <input type="submit" value="Continue" class="smallbutton">
</form>
</form>
```

Approved Receipt

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

```
ssl_card_number=00*****0000
ssl_exp_date=0515
ssl_amount=3.33
ssl_get_token=Y
ssl_customer_code=
ssl_salestax=1.00
ssl_token=8138827982163003
ssl_token_response=SUCCESS
ssl_result=0
ssl_result_message=APPROVAL
ssl_txn_id=AA776F-1981E7F3-907D-44F1-93BC-48D52F8CC2DF
ssl_approval_code=N44032
ssl_cvv2_response=
ssl_avs_response=
ssl_account_balance=0.00
ssl_txn_time=03/22/2013 05:44:43 AM
```

Example 2: processxml.do

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

Initial Request

```
xmldata=<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_transaction_type>ccgettoken</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1222</ssl_exp_date>
  <ssl_avs_address>7300</ssl_avs_address>
  <ssl_avs_zip>12345</ssl_avs_zip>
  <ssl_verify>N</ssl_verify>
</txn>
```

Response Receipt

```
<txn>
  <ssl_result>0</ssl_result>
  <ssl_token_response>SUCCESS</ssl_token_response>
  <ssl_token>7876275006683003</ssl_token>
</txn>
```

Example 3: processxml.do

The following XML code example demonstrates the initiation of a token generation with card verification request and response.

Initial Request

```
xmldata=<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_transaction_type>ccgettoken</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1222</ssl_exp_date>
  <ssl_avs_address>7300</ssl_avs_address>
  <ssl_avs_zip>12345</ssl_avs_zip>
  <ssl_verify>Y</ssl_verify>
</txn>
```

Response Receipt

```
<txn>
  <ssl_transaction_type>ccgettoken</ssl_transaction_type>
  <ssl_result>0</ssl_result>
  <ssl_token_response>SUCCESS</ssl_token_response>
  <ssl_token>7876275006683003</ssl_token>
  <ssl_result_message>SUCCESS</ssl_result_message>
  <ssl_avs_response>A</ssl_avs_response>
  <ssl_cv2_response>M</ssl_cv2_response>
</txn>
```

Example 4: processxml.do

The following XML code example demonstrates the initiation of a sale transaction with a token request:

```

xmldata=<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_transaction_type>ccsale</ssl_transaction_type>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1222</ssl_exp_date>
  <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
  <ssl_cvv2cvc2>321</ssl_cvv2cvc2>
  <ssl_amount>12.00</ssl_amount>
  <ssl_avs_address>7300</ssl_avs_address>
  <ssl_avs_zip>12345</ssl_avs_zip>
  <ssl_get_token>Y</ssl_get_token>
</txn>

```

Response Receipt

```

<txn>
  <ssl_approval_code>CVI194</ssl_approval_code>
  <ssl_dynamic_dba/>
  <ssl_exp_date>1222</ssl_exp_date>
  <ssl_server />
  <ssl_account_balance>12.00</ssl_account_balance>
  <ssl_token>7876275006683003</ssl_token>
  <ssl_get_token>Y</ssl_get_token>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_token_response>SUCCESS</ssl_token_response>
  <ssl_base_amount>12.00</ssl_base_amount>
  <ssl_amount>12.00</ssl_amount>
  <ssl_txn_id>AA4843B-0E98A88D-6923-4A77-B546-
  38CFDDCE4DA4</ssl_txn_id>
  <ssl_result>0</ssl_result>
  <ssl_card_number>00*****0000</ssl_card_number>
  <ssl_txn_time>03/26/2013 02:50:56 PM</ssl_txn_time>
  <ssl_avs_response>S</ssl_avs_response>
  <ssl_cvv2_response>M</ssl_cvv2_response>
</txn>

```

Example 5: processxml.do

The following XML code example demonstrates how to initiate a sale transaction, request a token and have it stored in the Card Manager:

```
xmldata=<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_transaction_type>ccsale</ssl_transaction_type>
  <ssl_amount>10.00</ssl_amount>
  <ssl_card_number>0000000000000000</ssl_card_number>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_invoice_number>1234</ssl_invoice_number>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
  <ssl_cvv2cvc2_indicator>1</ssl_cvv2cvc2_indicator>
  <ssl_cvv2cvc2>123</ssl_cvv2cvc2>
  <ssl_get_token>Y</ssl_get_token>
  <ssl_add_token>Y</ssl_add_token>
</txn>
```

Response Receipt

```
<txn>
<ssl_approval_code>CVI371</ssl_approval_code>
<ssl_cvv2_response>M</ssl_cvv2_response>
<ssl_add_token_response>Card Added</ssl_add_token_response>
<ssl_account_balance>10.00</ssl_account_balance>
<ssl_token>7098165265161885</ssl_token>
<ssl_token_response>SUCCESS</ssl_token_response>
<ssl_result_message>APPROVAL</ssl_result_message>
<ssl_invoice_number>1234</ssl_invoice_number>
<ssl_amount>10.00</ssl_amount>
<ssl_txn_id>AA4843B-0E213944-FEBA-4841-ADEA-
8804C127AB3</ssl_txn_id>
<ssl_result>0</ssl_result>
<ssl_card_number>00*****0000</ssl_card_number>
<ssl_txn_time>02/18/2014 04:06:41 PM</ssl_txn_time>
<ssl_avs_response>D</ssl_avs_response>
<ssl_invoice_number>1234</ssl_invoice_number>
<ssl_first_name>John</ssl_first_name>
<ssl_last_name>Doe</ssl_last_name>
<ssl_avs_address>123 Main</ssl_avs_address>
<ssl_city>Atlanta</ssl_city>
<ssl_state>GA</ssl_state>
<ssl_avs_zip>30123</ssl_avs_zip>
<ssl_country>USA</ssl_country>
</txn>
```

Since the token is now stored in the card manager, any consecutive Sale request can be easily done by sending tokens:

```
xmldata=<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_transaction_type>ccsale</ssl_transaction_type>
  <ssl_amount>10.00</ssl_amount>
  <ssl_token>7098165265161885</ssl_token>
</txn>
```

If the token is not stored in the card manager, you must send expiration data and billing information with the Sale request:

```
xmldata=<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_transaction_type>ccsale</ssl_transaction_type>
  <ssl_amount>10.00</ssl_amount>
  <ssl_token>7098165265161885</ssl_token>
  <ssl_exp_date>1215</ssl_exp_date>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
  <ssl_avs_address>123 Main</ssl_avs_address>
  <ssl_city>Atlanta</ssl_city>
  <ssl_state>GA</ssl_state>
  <ssl_avs_zip>30123</ssl_avs_zip>
  <ssl_country>USA</ssl_country>
</txn>
```

Encryption

Developers can use `process.do` or `processxml.do` to process swiped credit card transactions using an encrypting device. The transaction sent to `processxml.do` is formatted in XML syntax and includes all supported transaction elements nested between one beginning and ending element `<txn>`, the transaction sent to `process.do` is formatted in key value pairs

Swiped transactions should be submitted with encrypted track data using a supported MagneSafe™ secure card authenticator to maximize data protection instead of the standard track data. The encryption is delivered inside the read head closest to the magnetic stripe. This adds an additional layer of protection to the HTTPS POST request. The following swipe devices are supported:

- MagTek BulleT
- Magtek iDynamo
- MagTek uDynamo
- MagTek aDynamo
- MagTek Dynamag

Note: You must download the appropriate SDK from the Magtek site to enable the card reader to communicate with your mobile software. The SDKs can be found at <http://www.magtek.com> by selecting the Programming Tools link under the Support section.

The following transaction types are supported when submitting encrypted swiped transactions:

| Transaction Types | Description |
|----------------------------|---------------------------|
| <code>ccsale</code> | Credit Card Sale |
| <code>ccauthonly</code> | Credit Card Auth Only |
| <code>ccforce</code> | Credit Card Force |
| <code>ccccredit</code> | Credit Card Credit/Refund |
| <code>ccbalinquiry</code> | Credit Card Inquiry |
| <code>egcsale</code> | Gift Card Sale |
| <code>egcactivation</code> | Gift Card Activation |
| <code>egccardrefund</code> | Gift Card Refund |
| <code>egcreload</code> | Gift Card Reload |
| <code>egcbalinquiry</code> | Gift Card Inquiry |
| <code>egcccredit</code> | Gift Card Credit |

All Credit Card authorizations sent from an encrypting device must pass the following required information. For a complete listing of all recommended fields please refer to the [Credit Card Transactions](#) section:

| Field Name | Req? | Description |
|----------------------|------|---|
| ssl_merchant_id | Y | Converge ID as provided by Elavon. |
| ssl_user_id | Y | Converge user ID as configured on Converge (case sensitive). Must be passed, the Merchant Admin (MA) user ID cannot be used. |
| ssl_pin | Y | Converge PIN as generated within Converge (case sensitive). It is strongly recommended that your user PIN is 32 or 64 characters long. |
| ssl_transaction_type | Y | For example, the transaction type of (ccsale) to process a credit card sale or transaction type of (ccauthonly) to process a credit card sale. |
| ssl_vm_mobile_source | Y | Required for all encrypting devices. The mobile source of the transaction, supported values are: <ul style="list-style-type: none"> • BBERRY: For all BlackBerry mobile devices • ADROID: For all Android mobile devices • ITUNES: For all Apple mobile devices (iPhone/iPod/iPad) • WIN8: for all windows WIN8 based devices. • NOMOB: for non-mobile devices (PC or MAC) |
| ssl_ksn | Y | Required for all encrypting devices for payment cards. The Key Serial Number generated from the swiped payment card, this is the value returned by the Magtek device, which was used to encrypt the PAN data, using the Derived Unique Key Per Transaction (DUKPT) method. It is a 10 byte composite field that is transmitted as 20 alphanumeric character fixed length. |
| Card Data | Y | Refer to Card Data Elements table. |
| ssl_amount | Y | Transaction Sale Amount. Number with 2 decimal places. This amount includes the Net amount and Sales Tax. The authorized amount passed on request should not contain the tip amount, tips must be passed separately. The total authorized amount will be calculated during the authorization if tip is provided. For example: 1.00. For those terminals processing <i>Multi-Currency</i> , be sure to submit the correct number of decimal places for the transaction as some currencies have no exponents and some can have three. |
| ssl_vendor_id | Y | Unique vendor identifier assigned by Elavon, up to 08 characters. |

| Field Name | Req? | Description |
|---------------|------|---|
| ssl_mobile_id | N | Optional. Unique assigned mobile identification number of each Mobile device in use as determined by the merchant, required if the Mobile filter is enabled under the fraud prevention rules in Converge, up to 50 characters. This will help the merchant to identify which device is sending transactions and can be used to allow or block transactions generating from a specific device. |

Card Data Elements:

| Field Name | Req? | Description |
|---------------------------|------|--|
| ssl_encrypted_track1_data | C | Required for swiped credit or gift transactions. This is the encrypted Track 1 data extracted from the encrypting device. |
| ssl_encrypted_track2_data | C | Required for swiped credit or gift transactions. This is the encrypted Track 2 data extracted from the secure device. |
| ssl_card_number | C | Required for hand-keyed credit or gift transactions. Credit Card Number as it appears on the credit card. Pass this value when track data is not available. |
| ssl_exp_date | C | Required for hand-keyed credit or gift transactions. Credit Card Expiry Date as it appears on credit card formatted as MMY. Pass this value with a card number. |

Electronic Check ACH ECheck

In order to accept ACH ECheck, you must capture and pass the hand keyed check data during the initial processing including the ABA routing number, full account number, and account type. ACH ECheck represents types: WEB, TEL, PPD and CCD.

The following transactions types are supported when processing ACH transactions:

| Transaction Types | Description |
|-------------------|---------------------------|
| ecspurchase | Electronic Check Purchase |
| ecsvoid | Electronic Check Void |

Notes:

- ACH ECheck is available for region US Only.
 - The terminal must be setup for ACH ECheck.
 - ACH ECheck supports the following service Levels:
 - Conversion with Verification
 - Conversion Only
-

ACH Types

Converge set the ACH type by default based on the following criteria:

- Bank account type : Personal or Business
- Entry Mode: User Interface or Integration
- Market Segment: e-Commerce, card present or MOTO

Converge set the ACH type by default as follow:

- All business check purchases submitted using the user interface or integration will have an ACH type of CCD
- All personal check purchases submitted using the user interface will have an ACH type of TEL
- All personal check purchases submitted using the integration will have an ACH type of WEB

The integration can optionally specify the ACH type in the `ssl_ecs_product_code` field for each transaction to override the default values, the integration can send one of the following entries:

| ACH Type Value | Description |
|----------------|--|
| WEB | An ACH ECheck processing option for either reoccurring or single internet initiated entry processed based on a Customer's input of account information at a payment application website. |
| TEL | An ACH ECheck processing option for either reoccurring or single entry in which an electronic payment Item is created based on a Customer's oral authorization captured via the telephone. |
| PPD | An ACH ECheck processing option for either a reoccurring or single prearranged payment entry to a Customer's account pursuant to a written authorization that is obtained from the Customer. |
| CCD | An ACH-ECheck processing option in which a debit entry is initiated by an organization to another organization. |

To process an ACH ECheck transaction you must provide the following ACH Transaction variables:

| Field Name | Req? | Description |
|--------------------------------------|------|--|
| <code>ssl_aba_number</code> | Y | Bank Routing/Transit Number. |
| <code>ssl_bank_account_number</code> | Y | Bank checking account number. |
| <code>ssl_bank_account_type</code> | Y | The bank account type. One numeric digit value, valid values are: 0 for Personal, 1 for Business. |
| <code>ssl_agree</code> | Y | The agreement flag. One numeric digit value, valid values are: 1 for "I agree", 0 for "I do not Agree". A Declined will be returned when <code>ssl_agree = 0</code> |
| <code>ssl_ecs_product_code</code> | N | Optional on ACH ECheck. The ACH type. A 3 alpha value, valid values are: WEB, TEL, PPD, sent only when account type is personal to indicate the type of ACH transaction to process, this value is automatically set to CCD when the check is business. Optional on ACH ECheck. |
| <code>ssl_first_name</code> | C | Required for personal checks. The cardholder first name. |
| <code>ssl_last_name</code> | C | Required for personal checks. The cardholder last name. |
| <code>ssl_company</code> | C | Required for business checks. Company Name. |
| <code>ssl_recurring_flag</code> | N | The recurring flag can be sent to indicate if a check sale transaction is a recurring or an installment payment. Optional. This option should only be used if maintaining your own recurring and installment database. Valid values: 1=Recurring, 2=Installment. |

Transaction Flow

The following steps outline the process of sending an ACH ECheck transaction with `process.do` or `processxml.do`:

1. Collect the check data information as described in Electronic Check Purchase (`ecspurchase`) section, either by collecting the check data from the Cardholder or presenting the payment page and have the Cardholder enter them:
 - If you are using `process.do(true)`, additional data will be collected from the Cardholders after the payment page displays. Shown below are the key value pairs from the header by themselves for a check purchase request to display the payment page:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=true
ssl_transaction_type=ecspurchase
ssl_amount=10.00
```

- If you are using `process.do(false)` or `processxml.do`, you must collect all necessary information along with the check data. Shown below are the key value pairs from the header by themselves for a check purchase request:

```
ssl_merchant_id=xxxxxxx
ssl_user_id=xxxxxxx
ssl_pin=xxxxxxx
ssl_show_form=false
ssl_transaction_type=ecspurchase
ssl_amount=10.00
ssl_aba_number=123456789
ssl_bank_account_number=123456789011
ssl_bank_account_type=0
ssl_agree=1
ssl_first_name=Jane
ssl_last_name=Doe
ssl_result_format=HTML
ssl_receipt_decl_method=REDG
ssl_error_url=http://www.url.com/cgi-bin/testtran.cgi
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
```

Notes:

- If the check is personal, Cardholder or clerk must enter the first and last name of the account holder
 - If the check is business, Cardholder or clerk must enter the company name.
-

2. Collect the agreement by specifying the agreement indicator: Cardholder agreement is collected either by
 - a) If you are using `process.do(true)`, Cardholder provides agreement by clicking on the Agree check box for the payment page (wording provided at the bottom of the page)
 - b) If you are using `process.do(false)` or `processxml.do`, you must collect the agreement and send the `ssl_agree` indicator, either by phone, previous authorization in writing and signature or directly by the customer. If you are collecting payment using a website, your website must display clear wording to the Cardholder. Refer to the Electronic Check ACH ECheck [Best Practices](#) section for more information.
-

Notes:

- For Internet-Initiated Entries: If the payment is collected in a website, the website must display a clear written agreement stating its terms.
 - For Telephone-Initiated Entries: Clerk must obtain the customer's authorization prior via the telephone or the authorization must be captured using an automated voice response system, orally spoken.
 - For Prearranged Payments: Clerk obtains the customer's authorization by paper and passes the agreement to the point of sale application on behalf of the Cardholder.
-

3. Submit a transaction request using HTTPS, either by the HTTP GET or POST

Transaction Examples

Example 1: process.do

Shown below are the key value pairs from the header by themselves for a personal ACH ECheck transaction, this ACH ECheck type is a WEB transaction:

```
ssl_merchant_id= my_virtualmerchant_id
ssl_user_id= my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_transaction_type=ecspurchase
ssl_amount=20.00
ssl_bank_account_type=0
ssl_aba_number=123456789
ssl_bank_account_number=1234567890
ssl_agree=1
ssl_first_name=John
ssl_last_name=Doe
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
```

Example 2: process.do

Shown below is an example of an ACH ECheck type where the integrated application indicated that the ACH type was a PPD, the ACH type can be specified for personal checks only:

```
ssl_merchant_id= my_virtualmerchant_id
ssl_user_id= my_user_id
ssl_pin=my_pin
ssl_show_form=false
ssl_transaction_type=ecspurchase
ssl_amount=20.00
ssl_bank_account_type=0
ssl_ecs_product_code=PPD
ssl_aba_number=123456789
ssl_bank_account_number=1234567890
ssl_agree=1
ssl_first_name=John
ssl last name=Doe
```

Example 3: processxml.do

Send an ecspurchase request for a personal check:

```
xmldata=<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_transaction_type>ecspurchase</ssl_transaction_type>
  <ssl_amount>20.00</ssl_amount>
  <ssl_bank_account_type>0</ssl_bank_account_type>
  <ssl_aba_number>123456789</ssl_aba_number>
  <ssl_bank_account_number>1234567890</ssl_bank_account_number>
  <ssl_agree>1</ssl_agree>
  <ssl_first_name>John</ssl_first_name>
  <ssl_last_name>Doe</ssl_last_name>
</txn>
```

Receive a response and print receipt

```
<txn>
  <ssl_approval_code>NACREN</ssl_approval_code>
  <ssl_agree>1</ssl_agree>
  <ssl_bank_account_type>0</ssl_bank_account_type>
  <ssl_result_message>APPROVAL</ssl_result_message>
  <ssl_aba_number>123456789</ssl_aba_number>
  <ssl_reference_number>212182533</ssl_reference_number>
  <ssl_base_amount>20.00</ssl_base_amount>
  <ssl_amount>20.00</ssl_amount>
  <ssl_txn_id>AA49315-2C70E99A-F9E0-4C65-9339-
  F8E6AB917B4A</ssl_txn_id>
  <ssl_bank_account_number>12*****7890</ssl_bank_account_num
  ber>
  <ssl_result>0</ssl_result>
  <ssl_txn_time>02/12/2014 11:25:33 AM</ssl_txn_time>
</txn>
```

Best Practices

Authorization information:

Make sure your integrated application meets the following:

- Clear and expressed authorization language:

Example 1:

By Clicking the "I Agree" box below, you authorize (Merchant Name) to use information from your check to initiate a one-time fund transfer from your account or to process the payment as a check transaction or bank drawn draft from your account for the amount of (total amount here). If you payment is returned due to insufficient funds, you authorize us to make a one-time electronic funds transfer or to use a bank draft drawn from your account to collect a fee as allowed by state law.

Example 2:

I authorize (Merchant Name) to use information above to initiate an electronic fund transfer from my account or to process the payment as a check transaction or bank drawn draft from my account for the amount of (total amount here). If my payment is returned due to insufficient funds, I authorize (Merchant Name) to make a one-time electronic funds transfer or to use a bank draft drawn from my account to collect a fee as allowed by state law.

Example 3:

I authorize (Merchant Name) to use information above to initiate an electronic fund transfer from my bank account with account number (Masked Account Number) and routing number of (Masked Routing Number) or to process the payment as a check transaction or bank drawn draft from my account for the amount of (total amount here). If my payment is returned due to insufficient funds, I authorize (Merchant Name) to make a one-time electronic funds transfer or to use a bank draft drawn from my account to collect a fee as allowed by state law.

- Amount of transaction: Example, For a single payment or recurring payment that is for the same amount or a range of payments
- The effective date of transaction
- The Cardholder's Account number
- The Cardholders financial institution routing number
- Revocation language

Authorization Retention:

Retain a copy of the authorization to the customer for all agreement electronically or by paper two years after the termination or revocation of authorization and be prepared to provide records upon request to demonstrate proof of WEB entry, provide details of transaction to show what was exchanged.

To Minimize Potential Fraudulent ACH Transactions:

Use robust authentication methods to verify identity of Cardholder before accepting transactions. Customers with an established business relationship with the Cardholder, whether established online, in person, over the telephone, or by some method, can usually authenticate using shared secrets, such a PIN, password or previous transaction history. Example: checking information against databases, ask a challenge question, sending a specific piece of information, either online or offline, and asking to verify that information as a second step.

TEL Transactions may only be initiated when there is an existing relationship between Customer and Cardholder or when the Cardholder initiated the call.

Solicitations

Merchants are prohibited from making direct payments on behalf of their Cardholders.

Chapter 7: 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 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 |
| ND | INVLD TERM ID 2 | Invalid SE Number Note: AMEX Only |

| Code | Message | Definition |
|------|-------------------|---|
| 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.

| 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 |

| Code | Message | Definition |
|------|--------------------|---|
| 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 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 |

| AVS Response Code | Definition |
|-------------------|--|
| 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/CVC2 Response Codes are returned in the Authorization Response Message when the CVV2/CVC2 data is present in the transaction authorization request.

| AVS Response Code | Definition |
|-------------------|---|
| M | CVV2/CVC2 Match |
| N | CVV2/CVC2 No match |
| P | Not processed |
| S | Issuer indicates that the CVV2/CVC2 data should be present on the card, but the merchant has indicated that the CVV2/CVC2 data is not present on the card |
| U | Issuer has not certified for CVV2/CVC2 or Issuer has not provided Visa with the CVV2/CVC2 encryption keys |

Error Codes

A Converge 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: 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 Converge ID was not supplied in the authorization request. |
| 4001 | VID, UID and PIN Invalid | The Converge 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. |

| Code | Error Name | Default Message |
|------|---|--|
| 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 Converge ID and/or User ID supplied in the authorization request are 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. |
| 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. |
| 4019 | User ID not supplied | The User ID was not supplied in the authorization request. |
| 4022 | The system is unavailable | The system is unavailable. This transaction request has not been approved. Please try again later. |
| 4023 | Settlement is not allowed for this terminal | Settlement is not allowed for this terminal. |
| 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. |

| Code | Error Name | Default Message |
|------|--------------------------------|---|
| 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 PINLess 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. |
| 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. |
| 5018 | Missing MAC value | The MAC value sent appears to be incorrect. |
| 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 value is: T for Triple-DES DUKPT. |
| 5030 | Invalid Billing cycle | The billing cycle specified is not a valid entry. |

| Code | Error Name | Default Message |
|------|--|---|
| 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 | Transaction ID is invalid for this transaction type. |
| 5041 | Exceeded the 10 minutes window | Unable to void transaction, exceeds the 10mn window. |
| 5042 | Swipe data is not allowed for this market segment | Swipe data is not allowed for this market segment. Please rekey the card data. |
| 5043 | End Of Month and Next Payment Date combination mismatch | The end of the Month value provided [value] doesn't correspond with the next payment date of [value]. |
| 5050 | Invalid tip | Tip Amount is invalid. |
| 5055 | Missing response file name | The response file name is missing. Please provide a response file name and try again. |
| 5056 | Invalid response file name | The response file name is invalid. Please make sure the response file name doesn't contain any of the characters : \ / : * ? " < > |
| 5057 | Missing batch file | The batch file is missing. Please make sure the file exists and try again. |
| 5058 | Invalid batch file name | The batch file name you provided is too long. The file name cannot exceed 30 characters. |
| 5059 | Invalid batch file format | The batch file you uploaded is invalid. Please make sure that the file is properly formatted. |
| 5060 | Invalid batch file extension | The batch file you uploaded has an incorrect extension. Please make sure the file is in either CSV or XML format and try again. |
| 5061 | Import Batch in Progress, retry later | Import Batch in Progress, retry later. |
| 5062 | File Exceeds Max Number of Transactions | File Exceeds Max Number of Transactions. |

| Code | Error Name | Default Message |
|------|--|--|
| 5063 | File already imported | File already imported. |
| 5064 | Invalid fields in the batch file | The batch file you uploaded is invalid. Please make sure that the file is properly formatted and file doesn't contain any (fieldname) field or values. |
| 5065 | Invalid response file length | The response file name you provided is too long. The response file name cannot exceed 30 characters. |
| 5066 | Batch Import Limit exceeded | The number of import batch files has exceeded the limit allowed within 24 hours. |
| 5067 | Invalid transaction type present in batch file | The batch file you uploaded is invalid. Please make sure that the file contains valid and supported transaction types. For a complete list of all supported transaction types please consult manual. |
| 5068 | Error processing batch | There was an error in processing your batch. If this issue persist please contact Technical Support at 1-800-377-3962, option 2 then option 2 (in Canada you are asked to choose either English or French for you language). |
| 5069 | Invalid Batch Import Request | Only Multipart Form Data request are accepted. |
| 5070 | Signature already in System | 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 10K size quota. |
| 5074 | Signature is not allowed for this market segment | Signature is not allowed for this market segment. |
| 5078 | The MasterPass wallet is unavailable | The MasterPass wallet is currently unavailable, please retry at later time. |
| 5079 | MasterPass is not setup for this terminal | You must setup your terminal to accept MasterPass before sending this value. |
| 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. |
| 5083 | Invalid DBA name | The DBA name cannot contain special characters and based on the DBA prefix setup for your terminal, this value cannot exceed 21, 17 or 12 characters long. |
| 5085 | Invalid Token | The token supplied in the authorization request appears to be invalid. |

| Code | Error Name | Default Message |
|------|--|---|
| 5086 | Invalid card information | The card information supplied in the authorization request appears to be invalid. You cannot provide both token and card number for processing. |
| 5087 | Transaction currency is not allowed for this terminal | Transaction currency is not allowed for this terminal. You must be setup with <i>Multi-Currency</i> to specify a transaction currency. |
| 5088 | Transaction currency is not allowed for this Card Type | In order to process <i>Multi-Currency</i> you must use either a Visa or a MasterCard, you can also process this card type in the merchant local currency. |
| 5089 | Invalid Transaction Currency | The transaction currency sent is not correct. Please provide the correct 3-digit ISO code. |
| 5090 | Invalid BIN Override Value | Invalid BIN Override Value. |
| 5091 | Invalid Amount | The amount exceeds the original transaction amount. |
| 5092 | Invalid country code | Invalid country code. |
| 5093 | Transaction Time Exceeded | Transaction has timed out, you may retry at later time. |
| 5094 | Invalid Travel information | The travel departure or completion dates specified are not valid. Dates should be formatted as MM/DD/YYYY and cannot be in the past. |
| 5095 | Invalid Search Criteria | Too many search criteria were entered for this query. Modify the search and try again. |
| 5096 | Invalid Transaction Currency | The currency sent doesn't match the original transaction currency. |
| 5097 | Missing Travel information | The Departure Date and the Completion Date must be sent together. |
| 5098 | Invalid Transaction Currency | The transaction currency sent is not supported for this terminal. |
| 5099 | Invalid Search Date | Search dates must be formatted as MM/DD/YYYY. The end date must be greater than the start date and the range cannot be greater than 31 days. |
| 5100 | Account Closed | This account has been closed by Account Updater. |
| 5101 | MICR/Image Data is not allowed for this terminal | MICR read or check image is not allowed for this terminal. Your terminal must be setup for ECS Paper Check. |
| 5102 | Keyed Check is not allowed for this terminal | Key entered check is not allowed for this terminal. Your terminal must be setup for ACH ECheck. |
| 5103 | Invalid bank account type value | The value for the check type (<i>ssl_bank_account_type</i>) field should be 1 Numeric Character only. Valid values are: 0 for Personal, 1 for Business. |
| 5104 | Invalid agreement indicator value | The value for the agreement indicator (<i>ssl_agree</i>) field should be 1 Numeric Character only. Valid values are: 1 for Agree, 0 for Do not agree. |

| Code | Error Name | Default Message |
|-------------|--------------------------------------|---|
| 5105 | Invalid Bank Account Number | The bank account number supplied in the authorization request appears to be invalid. The bank account number must numeric and up to 16 characters long. |
| 5106 | Invalid Bank Routing Number | The bank routing number supplied in the authorization request appears to be invalid. The bank routing number must be numeric fixed value of 9 characters. |
| 5107 | Invalid Check Number | The Check number supplied in the authorization request appears to be invalid. The check number must be numeric. |
| 5108 | Missing check holder information | The check information supplied in the request appears to be missing. You must provide first and last names for a personal check or a company name for a business check. |
| 5121 | Invalid Mobile indicator | The mobile indicator appears to be missing or invalid. |
| 5122 | Invalid Merchant IP Address | The IP Address appears to be missing or invalid. |
| 5123 | Email is not setup for this terminal | Email is not setup for this terminal. You must enable the email options under the email setup form. |
| 6001 | Manual Transaction Declined | 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. |
| 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. |

| Code | Error Name | Default Message |
|------|-----------------------------|---|
| 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. |
| 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. |

| Code | Error Name | Default Message |
|-------------|---|---|
| 6025 | Request Exceeds Balance | 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 8: Code Samples

The following sections show code samples used by Converge.

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.d
o?
ssl_merchant_id=xxxxxx&ssl_user_id=xxxxxx&ssl_pin=xxxxxx&ssl_show
_form=false&ssl_result_format=ascii&ssl_card_number=00000000000000
000&ssl_exp_date=1208&ssl_amount=1.02&ssl_transaction_type=ccsale
&ssl_cvv2cvc2_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=00*****0000
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=M
ssl_avs_response=A
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 Converge 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 Converge. 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 Converge 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

//Uncomment the endpoint desired.
//Production URL
//$url =
'https://www.myvirtualmerchant.com/VirtualMerchant/process.do';
//Demo URL
//$url =
'https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do';

//Configuration parameters.
$ssl_merchant_id = 'xxxxxx';
$ssl_user_id = 'xxxxxx';
$ssl_pin = 'xxxxxx';
$ssl_show_form = 'true';
$ssl_result_format = 'HTML';
$ssl_test_mode = 'false';
$ssl_receipt_link_method = 'REDG';
$ssl_receipt_link_url = 'your_receipt_page_url_here';
$ssl_transaction_type = 'CCSALE';
$ssl cvv2cvc2 indicator = '1';
```

(continued)

```
//Declares base URL in the event that you are using the VM
payment form.
if($ssl_show_form == 'true')
{
    echo "<html><head><base href='" . $url .
    "'></base></head>";
}

//Dynamically builds POST request based on the information
being passed into the script.
$queryString = "";
foreach($_REQUEST as $key => $value)
{
    if($queryString != "")
    {
        $queryString .= "&";
    }
    $queryString .= $key . "=" . urlencode($value);
}

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, 1);
```

(continued)

```
curl_setopt($ch, CURLOPT_POSTFIELDS, $queryString .
    "&ssl_merchant_id=$ssl_merchant_id".
    "&ssl_user_id=$ssl_user_id".
    "&ssl_pin=$ssl_pin".

    "&ssl_transaction_type=$ssl_transaction_type".

    "&ssl_cvv2cvc2_indicator=$ssl_cvv2cvc2_indicator".
    "&ssl_show_form=$ssl_show_form".
    "&ssl_result_format=$ssl_result_format".
    "&ssl_test_mode=$ssl_test_mode".
    "&ssl_receipt_link_method=$ssl_receipt_link_method".

    "&ssl_receipt_link_url=$ssl_receipt_link_url");
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
curl_setopt($ch, CURLOPT_VERBOSE, true);

$result = curl_exec($ch);
curl_close($ch);

?>
```

(end)

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;}
?>
```

PHP Batch Import Script

```
<?php
$url = [Insert URL Here];
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_VERBOSE, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
curl_setopt($ch, CURLOPT_POST, 1);
// replace xxxxxx values with your own credentials
// value for ssl_import_file should be the path to the batch file
// on your web server
$post = array(
    "ssl_merchant_id"=>"xxxxxxx",
    "ssl_user_id"=>"xxxxxxx",
    "ssl_pin"=>"xxxxxxx",
    "ssl_test_mode"=>"false",
    "ssl_show_form"=>"false",
    "ssl_transaction_type"=>"CCIMPORT",
    "ssl_result_format"=>"HTML",
    "ssl_response_file"=>"curltest",
    "ssl_merchant_email"=>"xxxxxxx",
    "ssl_do_merchant_email"=>"T",
    "ssl_import_file"=>"@CC_Batch.csv"
);
curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
$response = curl_exec($ch);
curl_close($ch);
?>
```

Python Sample

PYTHON INPUT

```
import sys, urllib2, urllib
url =
'https://demo.myvirtualmerchant.com/VirtualMerchantDemo/process.do?
ssl_merchant_id=xxxxxxx&ssl_user_id=xxxxxxx&ssl_pin=xxxxxxx&ssl_show_f
orm=false&ssl_result_format=ascii&ssl_card_number=0000000000000000&
ssl_exp_date=1208&ssl_amount=1.02&ssl_transaction_type=ccsale&ssl_c
vv2cvc2_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 Converge:

```
ssl_card_number=00*****0000
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=M
ssl_avs_response=A
ssl_account_balance=0.00
ssl_txn_time=08/17/2008 10:20:27 AM
```

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.

HTML Sample

Batch Import

```

<html>
  <head>
    <title>Batch Import</title>
  </head>
  <body>
    <table width="80%" cellspacing="1" cellpadding="1" >
      <tr><td><b>Batch Import</b></td></tr>
    </table>
    <form action= [Insert URL Here] method="post" id="transactionForm"
    enctype="multipart/form-data">
      <table border="0" cellspacing="0" cellpadding="2" width="100%">
        <td width="150">Input File Name </td>
        <td> <input type="file" name="ssl_import_file" value="File Location"></td>
        <td width="150">Info:</td>
          <td>Account ID: <input type="text" name="ssl_merchant_id"> </td>
          <td>User ID: <input type="text" name="ssl_user_id"> </td>
          <td>PIN: <input type="text" name="ssl_pin"> </td>
        </tr>
        <tr >
          <td width="150">Transaction Type</td>
          <td nowrap="true">
            <select name='ssl_transaction_type'>
              <option value=''>Other</option>
              <option value='CCRECIMPORT'>Recurring</option>
              <option value='CCIMPORT'>Credit Card</option>
            </select></td></tr><tr >
          <td width="150">Result Format</td>
          <td nowrap="true">
            <select name='ssl_result_format'>
              <option value='XML'>XML</option>
              <option value='HTML'>HTML</option>
              <option value='ASCII'>ASCII</option>
            </select>
          </td></tr><tr>
          <td>Response File:</td>
          <td><input type="text" name="ssl_response_file" size="40"></td></tr>
          <tr><td>
            <input type="text" name="ssl_merchant_email" size="20"></td></tr>
            <input type="hidden" name="ssl_do_merchant_email" value="T"><tr>
            <td width="150">
            <input type="submit" value = "Upload File"/></td></tr>
          </table>
        </form>
      </body>
    </html>

```

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 |
|--------------------------|------------|------|-------------------|------------|------|---------|-------|--|
| <input type="checkbox"/> | MERCH_ID | text | latin1_swedish_ci | | No | | |       |
| <input type="checkbox"/> | MERCH_USER | text | latin1_swedish_ci | | No | | |       |
| <input type="checkbox"/> | 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 (continued):

```
<?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;
}
```

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

```
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;
}
```

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

```
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: #FFF0CF;
    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>
```

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

```
<?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>
```

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

```
<?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>
```

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

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

(continued)

Below is the php source code for ELAVON_CFG.php (continued):

```
<?php if ($page < $pagecount) { ?>
<td>&nbsp;<a href="ELAVON_CFG.php?page=<?php echo $page + 1
?>">Next&nbsp;&gt;&gt;</a>&nbsp;</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);
} ?>
```

(end)

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>
```

(continued)

Simple .Net DB Configuration Script (continued):

```
<center>
  <%
  while(!recordSet.EOF)
  {
  %>
  <form action=" [Insert URL Here] 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>
```

(end)

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 Converge payment form.
- **Auto-Settle**
An option that automatically settles all un-pended 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.
- **Converge Account**
The Converge Account your company has with Elavon.
- **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 Converge account. The Merchant Admin User ID (MA) is the same as the Converge 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 "Un-pended."
- **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 (for example: 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 Converge. See the chapter on **User Management** for details on how to make or edit Terminal Associations in the *Converge User Guide*.
- **Terminal Friendly Name**

Terminals are referred to in Converge 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 Converge, this is a virtual ID. You may have more than one terminal for use within your Converge 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 *Converge 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 Converge. The user ID is case sensitive.

- **User Rights**

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