
GoMerchant, LLC. GoMerchant.com

XML Gateway API Documentation With CVV2 Support

White Paper Version 1.0

GoMerchant, LLC.
523 - 525 Station Ave
Haddon Heights, NJ 08035

[Note: This information is provided under confidentiality and should not be discussed outside GoMerchant, LLC. without express written permission.]

Purpose

The purpose of this document is to outline the technology and processes that drive the XML Gateway API. The API provides an interface into the gateway transaction processing network via secure sockets. The end user, after reading this document, should have sufficient knowledge based on the documentation and working samples provided to successfully integrate their ecommerce application into the XML Gateway API.

What is the Gateway API

The XML gateway API is a programming interface that resides on transaction servers which communicate directly to credit card processing networks. The programming interface requires that the merchant, or their web programming staff, be sufficiently knowledgeable in programming skills in any programming or object oriented scripting language. Several programming samples in a variety of languages are provided for reference to coding into the XML Gateway API.

The XML Gateway API accepts the card purchasers information, including credit card information, billing address, total charge amount and order id and produces an authorization or decline directly from the merchant bank. The XML Gateway API provides methods to perform the following operations: AUTH, SALE, SETTLE, CREDIT, VOID and QUERY. The information is passed via 128bit SSL https post in XML format. That post occurs in the background from the merchant's server. Thus the purchaser never leaves the merchant's website. The authorization information is returned in XML format with full error trapping and reporting to indicate the success or failure of the transaction.

For merchant's running Windows NT, the only requirement is that IE 5.01 be installed on the system. IE 5.01 contains the object required to deliver the transaction data via secure SSL post. Alternatively, several other programming languages such as Java, C++, Perl, Visual Basic etc., provide libraries into the Open SSL/Crypt functions.

An example in Perl on Windows platform is available in this documentation.

Operation Types

The XML Gateway allows the merchant to perform an AUTH, SALE, SETTLE, CREDIT, VOID, QUERY operations.

An AUTH operation is nothing more than a transaction that reserves the funds on a customer's credit card. The AUTH transaction must be accompanied by a SETTLE transaction in order for the merchant to receive funds.

A SALE operation produces a transaction that authorizes and captures the funds of the customer's credit card. The settle will automatically take place at 2:00am EST and funds will be available within 24-48 hours. AUTH/SALE transactions are produced by the card issuing bank and passed to the gateway transaction servers in typically 5 seconds or less.

SETTLE operations allow the merchant to submit the reference numbers of corresponding authorized transactions for batch settlement. The settlement occurs at 2:00am each day.

CREDIT operations are performed against settled authorizations or sale transactions. Credits may only be performed against settled transactions and may not exceed the settled amount regardless of the original authorized amount. Multiple credits may be performed against a single settled authorization/sale up to the total settled amount.

VOID operations are performed against AUTH transactions that have not settled. Voids will prevent the authorized transaction from ever settling.

QUERY operations allow the merchant to query the transaction database for all transaction types over a specified query range. The transaction data is returned with reference numbers in order for further operations to be performed against each transaction. NOTE – the full card number will not be exposed in a query transaction as per Visa/Mastercard requirements.

To utilize the XML gateway API and obtain a credit card authorization or any of the other operation types, you must have obtained an active merchant credit card processing account through one of the accepted GoMerchant processing networks. Your Web server or transaction server MUST be capable of performing an https (SSL) silent POST or GET to GoMerchant's transaction servers and receive that data, not in the form of a URL but as an XML document that can be parsed for results.

Operational process:

1. The merchant's website produces the XML with all the required information depending on the operation type and performs an https (silent POST) or GET to gateway transaction servers.
2. The gateway's transaction servers determine the operation type and execute it.
3. Once the correct operation is carried out, the gateway assembles the response in XML.
4. The gateway's transactions servers send the XML response to the merchant's web or transaction servers indicating a success, failure or error.
5. The merchant parses the XML and notifies the customer or user of a successful request, failure or error.

Secure posting to the gateway transaction servers:

The URL used in the https post or get is:

https://www.gomerchant4.com/trans_center/gateway/xmlgateway.cgi

The XML format for all possible operations are illustrated in the next sections.

*NOTE – All keys and XML tags **MUST** be named exactly as below.

Auth/Sale Operations

AUTH/SALE (process initial credit card charge)

Send to XMLGateway:

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">merchant id</FIELD> # assigned by gateway
    <FIELD KEY="password">pwd</FIELD> # assigned by gateway
    <FIELD KEY="operation_type">auth/sale</FIELD>
    <FIELD KEY="order_id">MUST BE UNIQUE ASSIGNED</FIELD>
    <FIELD KEY="total">(In US Dollars. Example 59.99 No $ signs.)</FIELD>
    <FIELD KEY="card_name">(Visa, Amex, Discover or MasterCard)</FIELD>
    <FIELD KEY="card_number">15|16 digit credit card number</FIELD>
    <FIELD KEY="card_exp">MMYY</FIELD>
    <FIELD KEY="cvv2">3 digit cvv2 code</FIELD> #not required
    <FIELD KEY="owner_name">name</FIELD>
    <FIELD KEY="owner_street">address</FIELD>
    <FIELD KEY="owner_city">city</FIELD>
    <FIELD KEY="owner_state">state AA</FIELD>
    <FIELD KEY="owner_zip">zip</FIELD>
    <FIELD KEY="owner_country">country AA</FIELD>
    <FIELD KEY="owner_email">Email of customer </FIELD> #not required
    <FIELD KEY="owner_phone">Phone of customer</FIELD> #not required
    <FIELD KEY="recurring">0-no 1-yes</FIELD>
    <FIELD KEY="recurring_type">Null if recurring =0</FIELD>
    <FIELD KEY="remote_ip_address">Customer IP Address</FIELD> #not required
  </FIELDS>
</TRANSACTION>
```

NOTE

recurring_type options: daily, weekly, biweekly, monthly, quarterly, semiannually, annually

order_id options: only these characters are accepted: a-z, A-Z, 0-9, @, hyphen (-), space, comma (,), and a period (.). There is also a length restriction with a low limit of 1 and a max of 50 characters.

operation types:

- auth** – just gets the card authorized it does not settle the transaction.
- sale** – authorizes the card and settles the transaction.

Response from XMLGateway for operation_type auth or sale:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="status">0-error 1-success 2-declined</FIELD>
    <FIELD KEY="auth_code">character code sent by the bank </FIELD>
    <FIELD KEY="auth_response">message from the bank </FIELD>
    <FIELD KEY="avs_code">avs code from the bank</FIELD>
    <FIELD KEY="cvv2_code">cvv2 code from the bank</FIELD>
    <FIELD KEY="order_id">echoed back from original post</FIELD>
    <FIELD KEY="reference_number">returned for use with credits/voids/settles</FIELD>
    <FIELD KEY="error">error text</FIELD>
  </FIELDS>
</RESPONSE>
```

****NOTE****

status –

if it is 0 the response will look like this:

```
<FIELD KEY="status">0</FIELD>
<FIELD KEY="auth_code"></FIELD>
<FIELD KEY="auth_response"></FIELD>
<FIELD KEY="avs_code"></FIELD>
<FIELD KEY="cvv2_code"></FIELD>
<FIELD KEY="order_id">'WHATEVER WAS SENT IN</FIELD>
<FIELD KEY="reference_number"></FIELD>
<FIELD KEY="error">DESCRIPTIVE ERROR MESSAGE</FIELD>
```

Notice most fields are blank when **status** = 0! If the **status** is a 1 or 2 then the fields will be field in with the appropriate responses

reference_number – this is the number used in later operations to reference specific transactions for credits, voids and settles.

Credit Operations

CREDIT (credit money back to a settled charge)

The Credit operation allows the merchant to credit a previously settled transaction. The credit amount may not exceed the amount of the settlement even if the authorization was a greater amount than the settlement amount. Example, an auth operation was approved for \$10.00. The settlement was only authorized for \$5.00. The ensuing credit may only be performed for \$5.00.

Send to XMLGateway:

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">merchant id</FIELD>
    <FIELD KEY="password">pwd</FIELD>
    <FIELD KEY="operation_type">credit</FIELD>
    <FIELD KEY="total_number_transactions">number of transactions submitted</FIELD>
    <FIELD KEY="reference_number1..n">reference number returned from Auth/Sale</FIELD>
    <FIELD KEY="credit_amount1..n">amount to credit from corresponding reference number</FIELD>
  </FIELDS>
</TRANSACTION>
```

NOTE

reference_number – the number that was originally returned by the auth/sale command for each transaction or the number received from the xml gateway when a query is performed.

1..n – this means that if the **total_number_transactions = 3** then there should be a **reference_number** and a **credit_amount** for each of the **three** transactions.

Ex:

```
<FIELD KEY="total_number_transactions">3</FIELD>
<FIELD KEY="reference_number1">125</FIELD>
<FIELD KEY="credit_amount1">8.00</FIELD>
<FIELD KEY="reference_number2">128</FIELD>
<FIELD KEY="credit_amount2">5.00</FIELD>
<FIELD KEY="reference_number3">130</FIELD>
<FIELD KEY="credit_amount3">2.00</FIELD>
```

Returned from XMLGateway for operation_type credit:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="total_transactions_credited">Total number of transactions credited</FIELD>
    <FIELD KEY="status1..n"> 0-error 1-success 2-Rejected</FIELD>
    <FIELD KEY="response1..n">Response text returned by Transaction Center</FIELD>
    <FIELD KEY="reference_number1..n">reference number of credited transaction</FIELD>
    <FIELD KEY="credit_amount1..n">amount credited</FIELD>
    <FIELD KEY="error1..n">error text</FIELD>
  </FIELDS>
```

</RESPONSE>

****NOTE****

total_transactions_credited - the number of successful credits performed on the list of credits attempted. This value can be used to check the successful statuses of the returned credits and make sure the number of successful statuses is the same as the **total_transactions_credited**.

1..n – this means that if the **total_number_transactions = 3** which was passed in then there should be a **status, response, reference_number, credit_amount,** and **error** associated with each credit attempted. Each field would end in the number

example:

if **total_number_transactions = 3** then in the response there would be a **status1 , status2** and a **status3** field.

Void Operations

VOID (nullify a non-settled authorization, this will make an auth impossible to ever settle)

The Void operation allows the merchant to void an AUTH that has not been settled. Any auth transaction that has settled may no longer be voided. The transaction must be credited if it has settled. Sale transactions may not be voided since they automatically settle.

Send to XMLGateway:

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">merchant id</FIELD>
    <FIELD KEY="password">pwd</FIELD>
    <FIELD KEY="operation_type">void</FIELD>
    <FIELD KEY="total_number_transactions">number of transactions ubmitted</FIELD>
    <FIELD KEY="reference_number1..n">reference number returned from Auth/Sale</FIELD>
  </FIELDS>
</TRANSACTION>
```

****NOTE****

reference_number – the number that was originally returned by the auth/sale command for each transaction or the number received from the xml gateway when a query is performed.

1..n – this means that if the **total_number_transactions = 3** then there should be a **reference_number** for each of the **three** transactions.

example:

```
<FIELD KEY="total_number_transactions">3</FIELD>
<FIELD KEY="reference_number1">125</FIELD>
<FIELD KEY="reference_number2">128</FIELD>
<FIELD KEY="reference_number3">130</FIELD>
```

Returned from XMLGateway for operation_type void:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="total_transactions_voided">Total number of transactions voided</FIELD>
    <FIELD KEY="status1..n"> 0-error 1-success 2-rejected</FIELD>
    <FIELD KEY="response1..n">Response text returned by Transaction Center</FIELD>
    <FIELD KEY="reference_number1..n">reference number of credited transaction</FIELD>
    <FIELD KEY="error1..n">error text</FIELD>
  </FIELDS>
</RESPONSE>
```

****NOTE****

total_transactions_voided - the number of successful voids performed on the list of voids attempted. This value can be used to check the successful statuses of the

returned voids and make sure the number of successful statuses is the same as the **total_transactions_voided**.

1..n – this means that if the **total_number_transactions = 3** which was passed in then there should be a **status, response, reference_number**, and **error** associated with each void attempted. Each field would end in the number

example:

if **total_number_transactions = 3** then in the response there would be a **status1** , **status2** and a **status3** field.

Settle Operations

SETTLE (finalize an auth. This will put the money into your account.)

The Settle operation allows the merchant to batch settle any successful authorizations. Authorizations will not transfer money until settled. An authorization merely reserves the funds on the customer's credit card. The funds will not transfer to the merchant's bank account until a settle is performed against the authorization. Sale transactions do not need nor can they be settled.

Send to XMLGateway:

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">merchant id</FIELD>
    <FIELD KEY="password">pwd</FIELD>
    <FIELD KEY="operation_type">settle</FIELD>
    <FIELD KEY="total_number_transactions">number of transactions submitted</FIELD>
    <FIELD KEY="reference_number1..n">reference number returned from Auth/Sale</FIELD>
    <FIELD KEY="settle_amount1..n">amount to settle w/o $</FIELD>
  </FIELDS>
</TRANSACTION>
```

NOTE

reference_number – the number that was originally returned by the auth/sale command for each transaction or the number received from the xml gateway when a query is performed.

1..n – this means that if the **total_number_transactions = 3** then there should be a **reference_number** and a **settle_amount** for each of the **three** transactions.

example:

```
<FIELD KEY="total_number_transactions">3</FIELD>
<FIELD KEY="reference_number1">125</FIELD>
<FIELD KEY="settle_amount1">8.00</FIELD>
<FIELD KEY="reference_number2">128</FIELD>
<FIELD KEY="settle_amount2">5.00</FIELD>
<FIELD KEY="reference_number3">130</FIELD>
<FIELD KEY="settle_amount3">2.00</FIELD>
```

Returned from XMLGateway for operation_type settle:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="total_transactions_settled">Total number of transactions settled</FIELD>
    <FIELD KEY="total_amount_settled">amount that was successful set to settle pending</FIELD>
    <FIELD KEY="status1..n"> 0-error 1-success 2-rejected</FIELD>
    <FIELD KEY="response1..n">Response text returned by Transaction Center</FIELD>
    <FIELD KEY="reference_number1..n">reference number of credited transaction</FIELD>
    <FIELD KEY="settle_amount1..n">amount settled w/o $</FIELD>
    <FIELD KEY="batch_number1..n">Batch Number for corresponding transaction</FIELD>
    <FIELD KEY="error1..n">error text</FIELD>
  </FIELDS>
```

</RESPONSE>

****NOTE****

total_transactions_settled - the number of successful settles performed on the list of settles attempted. This value can be used to check the successful statuses of the returned settles and make sure the number of successful statuses is the same as the **total_transactions_settled**.

total_amount_settled – the total value of the transactions marked for settle pending which, if all approved will be deposited into your account.

1..n – this means that if the **total_number_transactions = 3** which was passed in then there should be a **status, response, reference_number, batch_number, settle_amount,** and **error** associated with each settle attempted. Each field would end in the number

ex:

if **total_number_transactions = 3** then in the response there would be a **status1 , status2** and a **status3** field.

Query Operations

5. QUERY

The Query operation allows the merchant to directly query the gateway transaction database for all transaction information based on the search criteria specified in the XML sent. The response includes a reference number with each transaction row for further operations against the transaction. Full card numbers will not be disclosed in the query operation as per Visa/Mastercard regulations. Only partial numbers will be made available, however no card number information is required to perform further operations against a transaction such as credit, void or settle. Only the reference number is required.

Send to XMLGateway:

```
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">merchant id</FIELD>
    <FIELD KEY="password">pwd</FIELD>
    <FIELD KEY="operation_type">query </FIELD>
    <FIELD KEY="card_type">( Visa, Amex, Discover or MasterCard )</FIELD> # not required
    <FIELD KEY="trans_type">SALE, AUTH, VOID, CREDIT, SETTLE</FIELD>
    <FIELD KEY="trans_status">1(successful) or 0(failed)</FIELD> # not required
    <FIELD KEY="begin_date">MMDDYY </FIELD>
    <FIELD KEY="begin_time">HHMMAM (HHMMPM)</FIELD>
    <FIELD KEY="end_date">MMDDYY</FIELD>
    <FIELD KEY="end_time">HHMMAM (HHMMPM)</FIELD>
    <FIELD KEY="order_id">unique oid</FIELD> # not required
    <FIELD KEY="card_number">15-16 digits</FIELD> # not required
    <FIELD KEY="low_amount">price without $ </FIELD> # not required
    <FIELD KEY="high_amount">price without $ </FIELD> # not required
  </FIELDS>
</TRANSACTION>
```

****NOTE****

for any of the not required fields if you leave them blank they query will search for all possible options and ignore the blank fields.

Returned from XMLGateway for operation_type query:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="records_found">total number of records returned </FIELD>
    <FIELD KEY="status"> 0-error 1-success 2-rejected</FIELD>
    <FIELD KEY="trans_type1..n">type of trans </FIELD>
    <FIELD KEY="trans_status1..n">status</FIELD>
    <FIELD KEY="settled1..n">1 (yes) or 0 (no)</FIELD>
    <FIELD KEY="credit_void1..n"> None, Full Credit, Partial Credit, Void </FIELD>
    <FIELD KEY="order_id1..n">unique id </FIELD>
    <FIELD KEY="reference_number1..n">unique id </FIELD>
    <FIELD KEY="trans_time1..n">MM/DD/YYYY HH:MM:SS AM/PM</FIELD>
```

```
<FIELD KEY="card_type1..n">( Visa, Amex, Discover or MasterCard) </FIELD>
<FIELD KEY="amount1..n">auth or sale amount with no $</FIELD>
<FIELD KEY="amount_settled1..n">settled amount with no $</FIELD>
<FIELD KEY="amount_credited1..n">credited amount with no $</FIELD>
<FIELD KEY="error1..n">error text</FIELD>
</FIELDS>
</RESPONSE>
```

****NOTE****

if **records_found** is 0 then only **error** is returned no **error1...n**

Error XML Responses

****GENERAL NOTES****

Malformed fields or bad xml in all cases will be returned in the xml response defined for the operation type sent in with the **error** field defined. For operation types that cannot be induced from incoming fields a generic xml response will come back with simply an **error** field defined with the reason for the failure.

Example Error:

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
  <FIELDS>
    <FIELD KEY="error">DESCRIPTIVE ERROR MESSAGE</FIELD>
  </FIELDS>
</RESPONSE>
```

Test Authorization Account Information

In order to test your implementation of the XML Gateway, you may supply test transaction data that will either produce an authorization or decline or error. By using the test data, you may ensure that you are properly communicating with the transaction gateway and also properly parsing the return values.

To test a successful response from the gateway, please provide the values for the following variables:

merchant = 1264
password = password
order_id = any unique invoice number.

To get an approved response, use the following information:

Visa: 4111111111111111 CVV2 = 123
Mastercard: 5000300020003003 CVC2 = 123
Discover: 6011111111111117 CVV2 = 123
Amex: 374255312721002 CVV2 = 1234

Address: 123 Test St
Zip: 12345-6789

The remaining field values you can make whatever you want to test all the various aspects of the API, just make sure to fulfill the defined criteria listed above for the different operation types.

* As this is a generic test account, try to use a formula that will generate unique order_id's based on your merchant name.

To test a decline response from the gateway, please provide the values for the following variables:

Use above information and any other card number other than those listed above. Legitimate card numbers will decline since this is a test account.

The remaining field values you can make whatever you want to test all the various aspects of the API, just make sure to fulfill the defined criteria listed above for the different operation types.

To test an error response from the gateway, please provide the values for the following variables:

To generate an error, use the above account information and provide bad XML or bad data in the XML fields and an error response will be generated.

The remaining field values you can make whatever you want to test all the various aspects of the API, just make sure to fulfill the defined criteria listed above for the different operation types.

NOTE – It is strongly recommended that all procedures be tested to ensure that your gateway integration is complete and correct.

Sample Code

The sample code is provided as is. The sample provides an XML interface to each of the operation_types in one process. This code can be cut out and saved on any windows platform with perl 5 installed and the XML::Simple and Win32::OLE modules installed. Only order_id and reference fields need necessarily be changed. The sample uses the test gateway account.

Windows NT/2000 Perl 5.6 – IE 5.01 Object Method

```
##### XMLgatewaytest.cgi #####
#
#This program is used to connect to the xml gateway api
#you will notice 6 variables named $xml_1 - $xml_6
#each variable contains the correctly formatted xml request required to send to the api
#for each of the 6 operation types that can be performed.
#Each request can be altered by you to test the api and validate the responses you receive
#back.
#
#You will also notice the two code snippets that handle seding the XML to the api
#and receiving the response back. Simply change which XML variable you want to
#either POST or GET to test the different operation types and responses
##### Explanation Over #####
#c:\perl5\bin\perl.exe

use strict;
use warnings;

$|=1;

# Perl library modules
use WIN32::OLE;
use XML::Simple;

print "Content-type:text/html\n\n";

my $xml_1 = '
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
```

```

    <FIELD KEY="merchant">1264</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">sale</FIELD>
    <FIELD KEY="order_id">YOURID_NUMBER</FIELD>
    <FIELD KEY="total">5.00</FIELD>
    <FIELD KEY="card_name">Visa</FIELD>
    <FIELD KEY="card_number">4111111111111111</FIELD>
    <FIELD KEY="card_exp">1106</FIELD>
    <FIELD KEY="cvv2">123</FIELD>
    <FIELD KEY="owner_name">Bob Auth</FIELD>
    <FIELD KEY="owner_street">123 Test St</FIELD>
    <FIELD KEY="owner_city">city</FIELD>
    <FIELD KEY="owner_state">PA</FIELD>
    <FIELD KEY="owner_zip">12345-6789</FIELD>
    <FIELD KEY="owner_country">US</FIELD>
    <FIELD KEY="recurring">0</FIELD>
    <FIELD KEY="recurring_type">annually</FIELD>
  </FIELDS>
</TRANSACTION>;

```

```

my $xml_2 = '<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">12364</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">credit</FIELD>
    <FIELD KEY="total_number_transactions">1</FIELD>
    <FIELD KEY="reference_number1">REF ID FROM SALE </FIELD>
    <FIELD KEY="credit_amount1">1.00</FIELD>
  </FIELDS>
</TRANSACTION>;

```

```

my $xml_3 = '<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">1264</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">void</FIELD>
    <FIELD KEY="total_number_transactions">3</FIELD>
    <FIELD KEY="reference_number1">REF ID FROM AUTH</FIELD>
    <FIELD KEY="reference_number2">REF ID FROM AUTH</FIELD>
    <FIELD KEY="reference_number3">REF ID FROM AUTH</FIELD>
  </FIELDS>
</TRANSACTION>;

```

```

my $xml_4 = '<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">1264</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">settle</FIELD>
    <FIELD KEY="total_number_transactions">2</FIELD>
    <FIELD KEY="reference_number1">REF ID FROM AUTH</FIELD>
    <FIELD KEY="settle_amount1">5.00</FIELD>
    <FIELD KEY="reference_number2">REF ID FROM AUTH</FIELD>
    <FIELD KEY="settle_amount2">2.00</FIELD>
  </FIELDS>
</TRANSACTION>;

```

```

my $xml_5 = '<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">1264</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">query</FIELD>
    <FIELD KEY="card_type"></FIELD>
    <FIELD KEY="trans_type">SALE</FIELD>
  </FIELDS>
</TRANSACTION>

```

```

                <FIELD KEY="trans_status">0</FIELD>
                <FIELD KEY="begin_date">100103</FIELD>
                <FIELD KEY="begin_time">1222AM</FIELD>
                <FIELD KEY="end_date">123103</FIELD>
                <FIELD KEY="end_time">1159PM</FIELD>
                <FIELD KEY="order_id"></FIELD>
                <FIELD KEY="card_number"></FIELD>
                <FIELD KEY="low_amount"></FIELD>
                <FIELD KEY="high_amount"></FIELD>
</FIELDS>
</TRANSACTION>;

my $xml_6 = '
<?xml version="1.0" encoding="UTF-8"?>
<TRANSACTION>
  <FIELDS>
    <FIELD KEY="merchant">1264</FIELD>
    <FIELD KEY="password">password </FIELD>
    <FIELD KEY="operation_type">auth</FIELD>
    <FIELD KEY="order_id">YOURID_NUMBER</FIELD>
    <FIELD KEY="total">5.00</FIELD>
    <FIELD KEY="card_name">Visa</FIELD>
    <FIELD KEY="card_number">4111111111111111</FIELD>
    <FIELD KEY="card_exp">1006</FIELD>
    <FIELD KEY="cvv2">123</FIELD>
    <FIELD KEY="owner_name">Bob Recurring_Sale</FIELD>
    <FIELD KEY="owner_street">123 test st</FIELD>
    <FIELD KEY="owner_city">city</FIELD>
    <FIELD KEY="owner_state">PA</FIELD>
    <FIELD KEY="owner_zip">12345-6789</FIELD>
    <FIELD KEY="owner_country">US</FIELD>
    <FIELD KEY="recurring">0</FIELD>
    <FIELD KEY="recurring_type"></FIELD>
  </FIELDS>
</TRANSACTION>;

my $SendObject = Win32::OLE->new('microsoft.XMLhttp');

$SendObject->open("POST", "https://www.gomerchant4.com/trans_center/gateway/xmlgateway.cgi", "false");
$SendObject->setRequestHeader("Content-type", "text/xml");
$SendObject->send();
my $response = $SendObject->responseText;

print "<html><head><title>testing xml gateway</title><head><body>";
print "GET RESPONSE: $response";

#contains key value pairs of xml returned
my %xml_pairs_get = &ParseXml($response);

$SendObject->open("POST", "https://securewww.gomerchant4.com/trans_center/gateway/xmlgateway.cgi",
"false");
$SendObject->setRequestHeader("Content-type", "text/xml");

$SendObject->send($xml_2);
$response = $SendObject->responseText;

print "<br><br>POST RESPONSE: $response";

#contains key value pairs of xml returned
my %xml_pairs_post = &ParseXml($response);

print "</body></html>";

#this method will parse the xml and return a hash of key value pairs
#which can be manipulated in any way you need them to be
sub ParseXml {
  my $xml = $_[0];
  my %RESULT;
  my $ref = eval { XMLin($xml) };
  if ($@) {

```

```

$RESULT{error} = $@;
return(%RESULT);
}

foreach my $key(@{$ref->{'FIELDS'}{'FIELD'}}) {
    my ($val1, $val2) = each %$key;
    my ($val3, $val4) = each %$key;
    my ($val5, $val6) = each %$key;
    if($val5 && $val6){
        $RESULT{$val6} = $val4;
    }
    else{
        $RESULT{$val4} = $val2;
    }
}
return(%RESULT);
}

#EOF

```

Conclusions

The XML Gateway API is an intuitive interface to the transaction servers that produce credit card transactions and a number of operations on those initial transactions. The gateway is capable of being programmed into by any programming language, from any platform as long as an SSL https silent post or get can be performed.