CGI Transaction Gateway API v2.18.0

Production / Live URLs to connect to:

https://www. secure.completegateway.com/gate

https://www. secure.completegateway.com/secure/gate

Port: 443 - HTTPS

API Overview

The gateway API can be accessed in two ways, server-side or client-side, depending on the capabilities of your scripting/cgi platform. The server-side method is safer but requires that your CGI make an SSL connection to the CGI in the background.
The client method is easier to implement but does not offer as strong security as the server method. Most notably it opens your cart to cross-site scripting attacks and could reveal sensitive data via server log files. Before filling any orders placed through a client-side system it is recommended that you double check authorizations against your batch reports.

It is highly recommended that you use the server method if possible. Please check http://cmsonline.com/pages/index.php?p=Products_and_Services/Complete_Gateway/Developers/index.html, which provides libraries and sample code for further assistance. Even with all of the security features built into the Complete Gateway, we can only protect a transaction so far. It is essential that web developers take security seriously and check all of their scripts for possible weaknesses.
Client certificate use is also recommended. To require a client certificate use https://www. secure.completegateway.com/secure/gate.php. If you do not want to use client certificates, then you should use https://www.secure.completegateway.com/gate.php.

Integration Methods

Server-Side Method

The server-side method uses the following steps and requires programming and web development knowledge:

1. First, the client's browser submits the checkout form securely to https://www.yourcompany.com/makeApayment.xxx
2. makeApayment.xxx validates the info and preforms any data correction (removing spaces from credit card numbers, etc.)
3. makeApayment.xxx opens a connection to https://www.secure.completegateway.com/gate.php and posts the required processing fields (listed below). This connection is created by your server (www.yourcompany.com) not by the client's browser (hence server-side). You can do this several ways which are listed under Language Specific Libraries or with our web services SOAP API.
4. yourscript.cgi reads the processing result returned by posting to https://www.secure.completegateway.com.gate.php
5. yourscript.cgi then interprets the data and returns the desired result back to the client browser.

Client-Side Method

The client-side method provides an easy integration into existing forms without the need for extensive CGI programming experience. It does not provide as clean an interface and may cause concern for the end user if they see what appears to be an unfamiliar site handling their credit card information (Complete Gateway). The following basic steps are used in a client-side transaction:

1. Client browser submits check out form (with required hidden fields) securely to https://www.secure.completegateway.com.gate.php
2. The card is then processed and the results are passed back to your script by redirecting the client browser to http://www.yoursite.com/yourscript.cgi?resultfields
3. yourscript.cgi then interprets the data and returns the desired result back to the client browser.

The following is an example of a client-side method form. (You would have to also write myorderform.cgi.)

```
<form action="https://www.secure.completegatway.com/gate.php" method="POST">
<input type="hidden" name="UMkey" value="Your_source_key_here">
<input type="hidden" name="UMredir" value="http://www.mycompany.com/cgi-
bin/myorderform.cgi">
<input type="hidden" name="UMinvoice" value="1234">
<input type="hidden" name="UMamount" value="1.00">
Full Name: <input type="text" name="UMname"><br>
Street: <input type="text" name="UMstreet"><br>
City: <input type="text" name="city"><br>
State: <input type="text" name="state"><br>
Zip: <input type="text" name="UMzip"><br>
<br>
<br>
Credit Card Info:<br>
Card Type: <select name=cardtype>
<option>Visa <option>Mastercard <option>Amex</select><br>
Card Number: <input type=text name=UMcard><br>
Expiration: <input type=text name=UMexpir><br>
Name On Card: <input type=text name=UMname><br>
<br>
<br>
<input type=submit value="Place Order">
</form>
```

Client-Side Method With Simple Redirection

Also known as “Direct Post Method”. If you do not have the programming knowledge needed to write a CGI script, you can use the gateway to manage your results. When a customer's credit card is approved, the gateway will redirect the customer to a specified URL. If the card is declined, the gateway can be set to redirect to the URL of your choice, or display a customizable template. To upload a template, log in at https://www.completegateway.com, click on Settings, and edit the source key for your form. On the source page is a box for “Declined Template.” Paste your HTML into this box.

The following is an example of a client-side method form. This differs from the above example in that you do not need to write a CGI to manage the server response.

```
<form action="https://www.secure.completegatway.com/gate.php " method="POST">
<input type="hidden" name="UMkey" value="Your_source_key_here">
<input type="hidden" name="UMredirApproved"
value="http://www.mycompany.com/orderapproved.html">
<input type="hidden" name="UMinvoice" value="123456">
<input type="hidden" name="UMamount" value="1.00">
Full Name: <input type="text" name="UMname"><br>
Street: <input type="text" name="UMstreet"><br>
City: <input type="text" name="city"><br>
State: <input type="text" name="state"><br>
Zip: <input type="text" name="UMzip"><br>
<br>
<br>
Credit Card Info:<br>
Card Type: <select name=cardtype>
<option>Visa
<option>Mastercard
<option>Amex</select><br>
Card Number: <input type="text" name="UMcard"><br>
Expiration: <input type="text" name="UMexpir"><br>
Name On Card: <input type="text" name="UMname"><br>
<br>
<br>
<input type="submit" value="Place Order">
</form>
```

ePayment Form / Check-Out Form

The payment form allows you to send the customer to Complete Gateway for the collection of secure payment information. This eliminates the need for individual merchants to maintain their ownSSL certificates. To implement the payment form, the merchant's website needs to redirect the customer to a specially formatted URL or display a form that will post to the Complete Gateway site. For more information on using the payment form, please see the ePayment Form Manual.

Processing Commands

The transaction api is typically used to process credit card sales but it can also accept many other transaction types by changing the UMcommand variable. The following section describes the different commands that are available and the data required for each.

Command	Alternate/Legacy Equivalents	Pin Required
cc:sale	sale	
cc:authonly	preauth, authonly	
cc:capture	capture	
cc:adjust	adjust	
cc:credit	credit	
cc:postauth	postauth	
check:sale	check	
check:credit	checkcredit, reverseach	Yes
void	cc:void, check:void	
refund	cc:refund, check:refund	

creditvoid		Yes

cc:sale - Credit Card Sale

```
UMcommand=cc:sale
```

The 'cc:sale' command is the default processing command. If UMcommand is left blank or omitted, the system will use 'cc:sale'. The 'cc:sale' command runs a standard credit card sale. It will charge (debit) the customer’s credit card for the amount specified. If the charge is successful the transaction will be placed in the merchant's currently open batch for settlement. As long as the merchant has their batch set to autoclose, no further action is required to capture these funds.

cc:authonly - Credit Card Authorization

```
UMcommand=cc:authonly
```

The 'cc:authonly' command runs a credit card authorization. It approved, the funds will be held in the customers account. The funds will remain held until either the transaction is captured and settled, or until the authorization code expires. The length of time before an authorization expires varies from bank to bank but generally it is recommended that the authorization be captured within 24-48 hours. Merchants should consult their merchant service provider for the information specific to their account. If a merchant does not capture the transaction, no funds will be received by the merchant.

cc:capture- Capture an AuthOnly Transaction

```
UMcommand=cc:capture
```

The 'cc:capture' command moves previously authorization only transaction into the batch for settlement. The original Transaction ID (refnum) must be passed in UMrefNum field. Additionally, the amount of originally authorized may be adjusted by passing the UMamount field. The tolerances for the settle amount vary depending on the type of Merchant Account and the merchant service provider. The transaction will be placed in the merchant's currently open batch for settlement. As long as the merchant has their batch set to autoclose, no further action is required to capture these funds.

cc:adjust- Adjust a Credit Card Transaction

```
UMcommand=cc:adjust
```

The 'cc:adjust' command allows you to make changes to an existing (unsettled) sale. The authorization amount can be increased (incremental authorization) or decreased (partial reversal) or not changed. Additional data elements such as tax amount and po number can be added. The original Transaction ID (refnum) must be passed in UMrefNum field. The tolerances for the settle amount vary depending on the type of Merchant Account and the merchant service provider. The adjust and capture commands function identically except that the adjust command does not place the transaction in the batch.

cc:postauth - Offline Credit Card Transaction

```
UMcommand=cc:postauth
```

The 'cc:postauth' command adds a transaction that was authorized outside of the gateway to the current batch. Typically this is used when a merchant obtains a Voice Authorization via the phone. To send a postauth transaction, the merchant must pass in the authorization code that was provided by bank in the UMauthCode field. The transaction will be placed in the merchant's currently open batch for settlement. As long as the merchant has their batch set to autoclose, no further action is required to capture these funds.

cc:credit - Credit Card Refund

```
UMcommand=cc:credit
```

The 'cc:credit' command is used to refund money to a credit card. It requires the credit card number and expiration date as well as the amount being refunded. This transaction is also referred to as an “Open Credit”. It does not associate the credit with an existing sale in the system. Some credit card processors do not support open credits. Merchants should verify with their provider before using this command. A safer alternative to the credit command is the “Refund” command. (see below). The credit is placed in the currently open batch. When the batch is closed the credit will be sent to the bank.

Check:Sale - ACH/EFT Check Sale

```
UMcommand=check:sale
```

The 'check:sale' command is used to debit money from a customer’s checking/savings account via ACH/EFT. To use this feature the merchant must have an account with a support check processor. To process a check:sale, the customer's account number and ABA Routing number must be sent in the UMaccount and UMrouting fields.

Check:Credit - ACH/EFT Check Credit

```
UMcommand=Check:Credit
```

The 'check:credit' command is used to send money to a customer’s checking/savings account via ACH/EFT. Funds will be transferred from the merchant's account and deposited into the customer's account. To use this feature the merchant must verify with their check processor that they can support this type of transaction. Check:Credit transactions are not designed to be refunds on previous sales but rather to pay a 3rd party. Example uses include paying commissions to resellers or paying vendors/contractors. To refund an existing “Check:Sale” transaction, the “Refund” command (see below) should be used. Due to the risk associated with processing Check:Credit transactions, this command requires the use of a pin on the source key.

void: Cancel Previous Transaction

```
UMcommand=void
```

The 'void' command cancels a pending transaction. For credit card transactions, this command removes the transaction from the current batch. For ACH check transactions, the transaction is removed from the file that is sent to the bank. In both cases, there is a limited amount of time that a void may be run. For credit cards, a transaction can no longer be voided once the batch has been closed. For checks, a transaction can no longer be voided once the file has been sent to bank. This typically happens at the end of each business day. The void requires that the original transaction reference number be passed in the UMrefNum field.

refund: Refund Previous sale

```
UMcommand=refund
```

The 'refund' command allows the merchant to refund some or all of a previous sale transaction. It can be used with both credit card and check sales. It requires that the Transaction ID (refnum) of the original sale be submitted in the UMrefNum field along with the amount to be refunded. If the amount is not submitted, then the entire amount of the original sale will be refunded. The refund command will work for both credit card and check transactions. Not all check processors support refunds on checks so Merchants should verify with their provider that they can use this command.

Optional Features

Merchant Email Receipts

(primarily used in conjunction with client-side method with simple redirection) You can enable automatic emailing of Merchant Receipts for the API gateway by editing your source on www.completegateway.com. Once you have logged into your account, click on “Settings.” Edit the source by clicking on the pencil next to its name and enter the desired email address(es) into the Merchant Receipt field. Separate multiple addresses with commas.

Electronic Checks

In order to process checks though the API, the merchant must have an account with one of our supported check processors. Please contact technical support if you have any questions about adding check processing capabilities to your account. The fields required for processing checks are: UMkey, UMrouting, UMaccount, UMamount, UMname and identification information. Identification information can either be UMssn (social security number) or UMdlnum and UMdlstate (driver's license number and state of issue).

Recurring Billing

The API allows you to add customers to a recurring billing cycle through client-side or server-side methods. In order for a customer to be added to a recurring billing cycle the UMaddcustomer command must be set to yes and the initial charge must be approved. Once the initial charge is approved and the UMaddcustomer has been set to yes, the customer will automatically be added to a recurring billing cycle. To set the initial amount and the recurring charge to different amounts, use UMamount to the initial charge and UMbillamount to the recurring charge. (If UMbillamount is left empty, UMamount will be the recurring charge.)If you do not want to enable recurring billing for this customer, set UMschedule=disabled.

VPAS (Verified by Visa) and UCAF (Mastercard Secure Code)

The gateway supports VPAS (Verified by Visa) and UCAF (Mastercard Securecode) with both an integrated authentication system and support for third party verification. Using the integrated system provides a quick, easy method for developers to support Verified by Visa and Mastercard Secure Code without requiring complicated XML messaging formats.

To use the integrated solution, the merchant must first have an account with Cardinal Commerce. (If the merchant does not have an account, but requests authentication, the transaction will be handled as if the cardholder is not enrolled in VPAS and/or UCAF.) The following process is required to use the integrated solution:

1. Merchant's site collects cardholder information
2. Merchant's site sends an authorization to gate.php with the UMcardauth flag set to true
3. Gateway checks to see if the cardholder is enrolled in the VPAS and/or UCAF program. If the cardholder has not set a password, the transaction is processed normally (gate.php will return UMstatus=Approved or UMstatus=Declined). If the cardholder does have a password then gate.php will return UMstatus=Verification indicating that the merchant's site needs to prompt the user for a password. In the response, gate.php will also send back UMacsurl and UMpayload.
4. Merchant's site must send the customer's browser to the URL contained in UMacsurl with three get values: PAReq set to the value in UMpayload, TermUrl set to the URL on the merchant's site that will continue the transaction, MD set to some identifying information (such as the order number) that will allow the order to proceed.
5. Customer enters their username and password. If authentication is successful, they will be sent back to TermUrl with the variable PaRes set.
6. Merchant's set sends a second authentication request to the gateway, identical to the one sent in step 2, except that UMpares is set to the value of PaRes.
7. Then gate.php returns Approved or Declined as usual.

If you are using a third party verification system, or implementing the Cardinal Commerce API on the merchant's side, simply pass UMcavv and UMeci with the authentication request. (Please note: UMxid is obsolete and will be ignored.)

Source Pin Code

To validate transaction authenticity, the merchant can set a pin code for a source. The pin is stored in the merchant's software, or entered manually when the transaction is placed. The pin is not sent to the gateway, but is instead used to create a hash (also known as a fingerprint or message digest) for a transaction. The transaction hash is created by combining the command, the pin, the transaction amount, the invoice, and an optional random seed value. This information is separated by colons and run through an md5 or sha1 algorithm. The algorithm produces a hash, which is then sent to the gateway in the UMhash field where it is matched against the hash from the pin on file. If the two hashes do not match, the transaction is rejected.

Format of UMhash

Field	Value	Description
Algorithm	'm' or 's'	A one character code indicating the algorithm used. 'm' = MD5 and 's' = SHA1
(separator)	'/'	
Seed	(up to 256 chars)	Optional random seed value. Must match seed value used inside hash value. Also must be upper case letter and/or numbers.
(separator)	'/'	
Hash	(32 or 40 Char Hex)	MD5 or Sha1 Hash value in hex format.
(separator)	'/'	
Response	'y' or 'n'	Request a response hash. If omitted, defaults to 'n'

Example UMhash value:

```
UMhash=m/2123123/8827380daee8c6f935d3eaecf63e646c/y
```

Calculating Hash Value

The data used to calculate the hash value are the command, the source key pin, the transaction amount, the invoice and the random seed value separated by colons. The resulting data is sent to MD5 or SHA1 function which will produce the hash value.

For example:

Command	sale
Pin	sd*s3j002jd
Amount	53.21
Invoice	34576721
Seed	1234

Would result in the following data:

Data String	sale:sd*s3j002jd:53.21:34576721:1234
MD5 Hash String	52d534dd45388432ac0a44c9174ffb3f
UMhash Value	m/1223/52d534dd45388432ac0a44c9174ffb3f/

The following is a sample implementation in PHP:

```
<?php
$umcommand = "sale" ;
$umkey = "Your_source_key_here" ;
$card = "4444111122223337" ;
$expir = "0209";
$pin = "hs4rk";
$amount = "29.30" ;
$invoice = "45671" ;

$hashseed = mktime ();   // mktime returns the current time in seconds since epoch.
$hashdata = $umcommand . ":" . $pin . ":" . $amount . ":" . $invoice . ":" . $hashseed
;

$hash = md5 ( $hashdata );   // php includes a built-in md5 function that will create
the hash

$request =
"UMkey=$umkey&UMhash=m/$hashseed/$hash/y&UMamount=$amount&UMinvoice=$invoice" ;
$request .= "&UMcard=$card&UMexpir=$expir" ;
?>
```

Verify Response Hash Value

If requested in the UMhash, the server will generate a response hash value. This response hash value can be used to verify that the response value was generated by the gateway. This is useful when using a response landing page with the client side or payment form integrations.

Field	Value	Description

Algorithm	'm' or 's'	A one character code indicating the algorithm used. 'm' = MD5 and 's' = SHA1
(separator)	'/'	
Seed	(up to 256 chars)	Seed value used to create
(separator)	'/'	
Hash	(32 or 40 Char Hex)	MD5 or Sha1 Hash value in hex format.

The hash value is the combination of the Pin, UMresult, UMrefNum and the Seed, separated by colons.

The following is a sample implementation in PHP:

```php
<?php
// Pin assigned to source key
$pin='1234';

// break apart response hash
if(!$_REQUEST['UMresponseHash']) die('Gateway did not return a response hash');
$tmp = explode('/', $_REQUEST['UMresponseHash']);
$gatewaymethod = $tmp[0];
$gatewayseed = $tmp[1];
$gatewayhash = $tmp[2];

// assembly prehash data
$prehash = $pin . ':' . $_REQUEST["UMresult"] . ':' . $_REQUEST["UMrefNum"] . ':' .
$gatewayseed;

// calculate what we think the hash should be
if($gatewaymethod=='m') $myhash=md5($prehash);
else if($gatewaymethod=='s') $myhash=sha1($prehash);
else die('Unknown hash method');

// Compare our hash to gateway's hash
if($myhash == $gatewayhash)
{
   echo "Transaction response validated";
} else {
   echo "Invalid transaction response";
}

?>
```

Split Payments

Requirements

- There will be 2 or more authorizations obtained, depending on how many “splits”
- Each merchant will have their respective split accessible in the reports section
- The cardholder sees two or more separate charges with the DBA names of the respective merchants on the statement.
- At least 2 MIDs (Merchant IDs) are required, with a source key generated in each account.

Usage

The transaction API can process multiple payments to multiple merchants/source keys in single call. This is useful in cases where a base sale is processed by the merchant and a separate handling fee is processed by another merchant. In the following examples we will use the scenario of a concert where the ticket price is

$25 dollars collected by the venue plus a $2 handling fee that is collected by the servicing company. The venue and the servicing company have two separate merchant accounts and two separate source keys.

```
<form action="https://www.secure.completegatway.com/secure/gate" method="POST">
<!-- Base Information,  Source key for concert venue -->
<input type="hidden" name="UMkey" value="Your_source_key_here">
<input type="hidden" name="UMname" value="John Smith">
<input type="hidden" name="UMamount" value="25.00">
<input type="hidden" name="UMdescription" value="Ticket to Concert">
<input type="hidden" name="UMcard" value="4444555566667779">
<input type="hidden" name="UMexpir" value="0909">
<input type="hidden" name="UMstreet" value="1234 Main Street">
<input type="hidden" name="UMzip" value="01029">

<!-- Additional Information for Service Fee,  Source key for Service Company -->
<input type="hidden" name="UM02key" value="Your_source_key_here">
<input type="hidden" name="UM02amount" value="2.00">
<input type="hidden" name="UM02description" value="Servicing Fee">

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

Each additional transaction is defined by adding a numerical index to the base transaction fields. For example, UMkey is the sourcekey for the first transaction and UM02key is the source key for the second transaction. If a third transaction is also going to be run, it can be specified using UM03key. Any fields that are not populated for the second transaction will be copied from the base transaction. For example if UMdescription is set to “Testing” but UM02description is omitted, the system will copy the UMdescription to UM02description. To prevent this from happening, make sure to specify a blank value:

```
<input type="hidden" name="UMdescription" value="First description">
<input type="hidden" name="UM02description" value="">
```

While typically this feature will be used to split the payment into just two transactions, it is possible to run up to 100 transactions. Then index number must always be two digits: UM02key not UM2key.

Handling errors

By default the gateway will stop if it encounters an error in one of the transactions. For example if the ticket fee of $25 is declined, processing stops and the $2 fee is not processed. If you would like to override this behavior, set UMonError=continue. This will cause all transactions to be processed, even if the first one is declined. You may also set UMonError=void, which will void any approvals if subsequent transactions are declined. For example, if the $25 is approved but the $2 service fee is declined, the system would go back and void the $25 approval.

Result Codes

The gateway will return indexed error codes that match the index on the input variables. For example UMauthCode will contain the authcode for the $25 ticket sale and UM02authCode will contain the auth code for the $2 service fee.

Examples

CreditVoid

The CreditVoid command allows you to “credit back” or “void out” a transaction based on the original transaction reference number. The command automatically checks the status of the transaction, if the transaction has been settled then a credit (for all or part of the initial transaction) is entered into the current batch. If the transaction has not been settled then it will be voided (removed from the current settlement batch). The only required property for this command is the refnum property. The amount specified must be equal to or less than the original transaction. If no amount is specified, the full amount will be refunded. Note: for security reasons, this command requires that a pin be configured on the source key.

```
<form action="https://www.secure.completegatway.com/gate">

<input type="hidden" name="UMkey" value="Your_source_key_here">

<!-- Enter your source key, above is a dummy key -->

<input type="hidden" name="UMcommand" value="creditvoid"> <!-- Credit / Void command -
->

<input type="hidden" name="UMrefNum" value="55001467">

<!-- Reference number (transaction ID) of the original transaction -->

<input type="hidden" name="UMinvoice" value="123456">

<input type="hidden" name="UMhash" value="cea9cc2f86f666edda4d855ce523f6ae">

<input type="submit" value="Process CreditVoid">

</form>
```

Use of a pin is required for this command. In order to complete the process your md5hash value must include the command, the PIN, and the invoice. The data string for this particular example is as follows: creditvoid:1234:123456: resulting in the md5 hash to look like: cea9cc2f86f666edda4d855ce523f6ae Due to the process referencing an older transaction the UMamount field does not need to be passed or included in the MD5 hash.

Developer Reference

Test Mode

Setting UMtestmode to "true" puts the transaction into "Test Mode". For more information on what test mode does and credit card numbers to use with test mode, please see the test mode reference page. **PLEASE NOTE:** It is recommended that developers use the sandbox server instead of test mode. Test mode is a simple echo test and is only useful for checking that you are send and receiving data correctly. **Transaction data is not stored when in test mode** which will prevent you from: viewing transaction data in the console, testing customer/merchant receipts, capturing/voiding transactions.

Sandbox Server

The sandbox server provides a full simulation of the production environment. It allows developers to test all aspects of processing before putting their project into production. The system is self contained and credit cards will not actually be charged. For more information on obtaining and using a sandbox account see the sandbox documentation. To use the transaction api on the sandbox server change the url from https://secure.completegateway.com/gate to https://test.completegateway.com/gate You will also need to use a UMkey generated on the sandbox server. Keys created in production will not work on the sandbox server and vice versa.

Address Verification System

These AVS codes are returned in the UMavsResultCode variable, and serve to provide developers with greater control over the AVS system. Many developers may choose to capture and display the UMavsResult variable instead. The UMavsResult variable contains the meaning of the code, rather than the code itself.
The following is a list of result codes for the Address Verification System (AVS) and what each one indicates. The codes listed in the Code column are the most common responses, but depending on the platform being used, codes listed in the Alternates column may be received.

Code	Alternates	Meaning
YYY	Y, YYA, YYD	Address: Match & 5 Digit Zip: Match
NYZ	Z	Address: No Match & 5 Digit Zip: Match
YNA	A, YNY	Address: Match & 5 Digit Zip: No Match
NNN	N, NN	Address: No Match & 5 Digit Zip: No Match
YYX	X	Address: Match & 9 Digit Zip: Match
NYW	W	Address: No Match & 9 Digit Zip: Match
XXW		Card Number Not On File
XXU		Address Information not verified for domestic transaction
XXR	R, U, E	Retry / System Unavailable
XXS	S	Service Not Supported
XXE		Address Verification Not Allowed For Card Type
XXG	G,C,I	Global Non-AVS participant
YYG	B, M	International Address: Match & Zip: Not Compatible
GGG	D	International Address: Match & Zip: Match
YGG	P	International Address: No Compatible & Zip: Match

Card ID Result Codes

The following is a list of result codes for the CVV2/CVC2/CID verification system and what each one indicates. These codes are returned in the UMcvv2ResultCode variable and provide developers with greater control over the CVV2 system. Many developers choose to capture and display the UMcvv2Result variable instead. The UMcvv2Result variable contains the meaning of the code rather than the code itself.

The card security codes are 3 or 4 digit codes printed or embossed on Visa, Mastercard, American Express and Discover cards. These codes are also referred to as CVV2, CVC, CSC or CCID. Their purpose is to provide additional protection against fraudulent card use. Below is a list of possible result codes.

Code	Meaning
M	Match
N	No Match
P	Not Processed
S	Should be on card but not so indicated
U	Issuer Not Certified
X	No response from association
(blank)	No CVV2/CVC data available for transaction.

Card Level Result Codes

The following is a list of result codes for the card level results and what each one indicates. These codes are returned in the UMcardLevelResult variable and provide developers with extended information about the type of Visa Card that is being processed.

Code	Description
A	Visa Traditional
B	Visa Traditional Rewards
C	Visa Signature
D	Visa Signature Preferred
G	Visa Business
H	Visa Consumer Check Card
I	Visa Commerce
K	Visa Corporate
M	Mastercard/EuroCard and Diners
S	Visa Purchasing
U	Visa TravelMoney
G1	Visa Signature Business
G2	Visa Business Check Card
J1	Visa General Prepaid
J2	Visa Prepaid Gift Card
J3	Visa Prepaid Healthcare
J4	Visa Prepaid Commercial
K1	Visa GSA Corporate T&E
S1	Visa Purchasing with Fleet
S2	Visa GSA Purchasing
S3	Visa GSA Purchasing with Fleet
DI	Discover
AX	American Express

API Version Number

It is essential that developers pay attention to the version number of the API. The version number is broken up into three positions: compatibility.features.fixes. For example: v2.4.1 The first position (2 in this example) represents the compatibility level. Any application developed with any version 2 of the API will always work with any other revision of version 2. But applications developed with version 3, will not be compatible with version 2.

When a new version of the gateway API is released, the URL will change to keep scripts written on the old version from breaking. The second position (4 in this example) refers to the addition of new fields/features. These features provide additional functionality and are not mandatory. The third position (1 in this example) indicates bug fixes, when nothing structural has changed within the API.

CGI Post Variables

The following is a list of CGI variables that need to be passed to the gateway url. If you are using the client-side method some can be embedded as hidden tags, and others much match your field. An example of a client-side form can be found above.

Note: CC stands for real-time credit card transactions, Checks stands for real-time check processing.

Base Fields

Field	Required	Description
UMcommand		Processing Command. Possible values are: cc:sale, cc:authonly, cc:capture, cc:credit, cc:postauth, check:sale, check:credit, void, refund and creditvoid. Default is sale.
UMkey	ALL	The source key (assigned by the server when you created a source in your virtual terminal).
UMhash	Check:Credit, CreditVoid	If the source key (UMkey) has a pin assigned a validation hash is required (See Source Pin Code section).
UMignoreDuplicate		Set to yes if you would like to override the duplicate transaction handling.
UMauthCode	CC:PostAuth	Authorization Code obtained “offline” (ie telephone authorization). Only required for Post Auth.
UMrefNum	CC:Capture,Void,Refund, CreditVoid	The UMrefNum received when a transaction was authorized via either the “sale” or “authonly” commands. Required for void and capture commands.
UMcard	CC:Sale, CC:AuthOnly, CC:Credit, CC:PostAuth	Credit Card Number with no spaces or dashes.
UMexpir	CC:Sale, CC:AuthOnly, CC:Credit, CC:PostAuth	Expiration Date in the form of MMYY with no spaces or punctuation.
UMrouting	Check:Sale, Check:Credit	Bank Routing number. Required when UMcommand is set to check or checkcredit.
UMaccount	Check:Sale, Check:Credit	Checking Account Number. Required when UMcommand is set to check or checkcredit.
UMaccounttype		The type of account to debit/credit, either “Checking” or “Savings”. If omitted or left blank, it will default to “Checking” Only applies to the Check:Sale and Check:Credit commands.
UMcheckformat		Record type of electronic check transaction. Not supported by all check processors. List of Check Record Types Also known as SEC code.
UMamount	CC:Sale, CC:AuthOnly, CC:PostAuth, CC:Credit, Check:Sale, Check:Credit	Charge amount without $. This is the grand total including tax, shipping and any discounts.
UMallowPartialAuth		Yes/No. Indicates whether to allow a partial authorization if the full UMamount is not available (for debit, prepaid and gift cards). If left blank or set to No, the transaction will be automatically canceled and reversed if full UMamount is not available
UMcurrency	*	Currency of UMamount. **Required if using a MCP based account.** Must be one of the 3 digit numeric codes found here.
UMtax		Portion of above charge amount (UMamount) that is sales tax.
UMnontaxable		Set to yes if transaction is not taxable. (optional, platform dependant)
UMtip		Portion of charge amount (UMamount) for gratuity (tip).
UMshipping		Portion of above charge amount (UMamount) that is for shipping charges.
UMdiscount		The amount of any discounts that were applied.
UMsubtotal		The amount of the order before tip, shipping, discount and tax were applied. UMsubtotal+UMtip+UMshipping-UMdiscount+UMtax must equal UMamount. If UMsubtotal is not specified, it will be ignored.
UMcustid		Unique customer id.
UMinvoice	*	Unique Invoice or order number. 10 digits. While not required, it is strongly recommended that this field be populated for CC:Sale, CC:AuthOnly, CC:PostAuth, CC:Credit, Check:Sale and Check:Credit
UMorderid		Order identifier. This field can be used to reference the order to which this transaction corresponds. This field can contain up to 64 characters and should be

		used instead of UMinvoice when orderid is longer that 10 digits.
UMponum		Purchase Order number. Only required for corporate purchasing cards.
UMticket		Obsolete, included for backward compatibility. Use UMinvoice instead
UMdescription		Description of transaction.
UMcomments		Optional transaction comments field. Comments are displayed on the transaction details page.
UMname	*	Name on card or checking account. While not required, it is strongly recommended that this field be populated for CC:Sale, CC:AuthOnly, CC:PostAuth, CC:Credit, Check:Sale and Check:Credit
UMstreet	*	Billing Street Address for credit cards. Used for Address Verification System. While not required, this field should be populated for Fraud Prevention and to obtain the best rate for Ecommerce credit card transactions. It should be populated for CC:Sale and CC:AuthOnly
UMzip	*	Billing Zip Code for credit cards. Used for Address Verification System. While not required, this field should be populated for Fraud Prevention and to obtain the best rate for Ecommerce credit card transactions. It should be populated for CC:Sale and CC:AuthOnly
UMcvv2	*	CVV2 data for Visa. Set to -2 if the code is not legible, -9 if the code is not on the card. While not required, this field should be populated for Fraud Prevention and to obtain the best rate for Ecommerce credit card transactions. It should be populated for CC:Sale and CC:AuthOnly
UMcustemail		Customer's Email Address.
UMcustreceipt		Yes/No. Sends the Customer a Receipt. This overrides the setting for the merchant and source.
UMcustreceiptname		Name of the custom receipt template to use. If omitted or if receipt not found, default customer receipt template will be used
UMdlnum		Driver's license number.
UMdlstate		Driver's license state of issue.
UMchecknum		Check number.
UMcheckimagefront		JPG image of front side of check. (optional) If data is encoded, encoding must match UMcheckimageencoding
UMcheckimageback		JPG image of back side of check. (optional) If data is encoded, encoding must match UMcheckimageencoding
UMcheckimageencoding		Encoding method used for check images. (Either base64 or raw). Be careful to avoid double encoding the data! Many check scanners output the image files already in base64 format. No additional encoding is required.
UMclerk		Indicates the clerk/person processing transaction, for reporting purposes. (optional)
UMtranterm		Indiactes the terminal used to process transaction, for reporting purposes. (optional)
UMresttable		Indicates the restaurant table, for reporting purposes. (optional)
UMip		The IP address of the client requesting the transaction. This is used in many of the fraud modules.
UMsoftware		Allows application developers to stamp their application name and version number onto each transaction. Used to assist customers with trouble shooting. (optional)
UMredir		
UMredirApproved		Redirection URL - If the card is approved, redirect to this URL. No fields are passed back. Typically merchants should enable merchant receipts with this option. To enable merchant receipts for a source, see Merchant Email Receipts

		above.
UMredirDeclined		Redirection URL - If the card is not approved, redirect to this URL. No fields are passed back. If UMredirApproved is set but UMredirDeclined is not, the gateway will display the template entered in the sources area. This feature overrides the "Declined Template" feature that is available in the Source Key settings screen. If both are set, the declined template will be ignored and the user will be redirect to the UMredirDeclined url.
UMechofields		Echo input fields in response. If UMechofields is set to yes then all fields included in the form (except for the credit card number, key, expiration and cvc) will be included in the redirection URL as GET variables. This is only useful with the client-side method. For example if you post a form with the field "comments" to the gateway and set UMredir to "http://mysite.com/handler.cgi" then the gateway will redirect to http://mysite.com/handler.cgi?comments=... Please Note: Use of this feature is not recommended for security reasons. These fields will typically be logged in your web server log files. On many hosting companies these log files are world-readable which means that anyone would be able to read the information.
UMonError		Instructs the gateway what to do when a decline is received when multiple transactions are being processed (see "Split Payments" above). Can be set to Stop, Continue or Void. Defaults to "Stop"
UMtestmode		If UMtestmode is set to 1 the gateway will simulate a transaction without actually processing the card. No transaction data is stored when testmode is enabled. You will not see the transaction on reports or in the batch. Use of testmode is discouraged for anything more than rudimentary integration testing. For a better simulation of a transaction, it is recommended that you use the Sandbox.

Billing and Shipping Address Fields

Billing Address	Shipping Address
UMbillfname	UMshipfname
UMbilllname	UMshiplname
UMbillcompany	UMshipcompany
UMbillstreet	UMshipstreet
UMbillstreet2	UMshipsreet2
UMbillcity	UMshipcity
UMbillstate	UMshipstate
UMbillzip	UMshipzip
UMbillcountry	UMshipcountry
UMbillphone	UMshipphone
UMemail	
UMfax	
UMwebsite	

Stored Customers and Recurring Billing

Field	Description
UMaddcustomer	If set to yes and the transaction has been approved, the system will create a new customer record and store the payment information for future use.
UMschedule	Determines the recurring billing schedule. Possible values are daily, weekly, biweekly, monthly, bimonthly, quarterly, biannually, or annually. (defaults to monthly) If you do not want to enable recurring billing for this customer, set UMschedule=disabled
UMbillsourcekey	If set to yes, the customer will be stored under the source key specified by UMkey. Otherwise the customer will be stored under the default 'Recurring' key.
UMbillamount	Sets the monetary amount to charge on each cycle. If this field is left blank the UMamount will be used instead. This

	is NOT the "initial" charge or "setup" charge, this is only the "recurring" charge. UMamount determines the initial charge.
UMnumleft	Number of transactions remaining in recurring billing cycle. If the recurring billing should go on indefinitely, set this to *. (defaults to *)
UMstart	Start date. Default is tomorrow. Must be entered in YYYYMMDD format. If set to UMstart=next the date of the next billing cycle will be used. For example if today is 1/10/2004 and UMschedule=monthly then UMstart will be set to 2/10/2004.
UMrecurringreceipt	Yes/No. Sends the Customer a Recurring Billing Receipt.

Merchant Defined Custom Fields

The merchant can create up to 20 fields for storing custom data. This data is stored for both transactions and customers. For a transaction this data is visible on the transaction details screen. For customers the data can be viewed by editing a customer in the customer database.

Field	Max Length	Description
UMcustom1	255	Optional fields for storing custom data.
UMcustom2	255	
UMcustom3	255	
...		
UMcustom19	255	
UMcustom20	255	

Line Item Details

Merchants can pass information about the individual line items that make up an order. This data is visible on the transaction details page. Up to 100 lines may be stored per transaction.

Field	Max Length	Description
UMline*productrefnum	12	(optional) Gateway assigned product RefNum, used for inventory control.
UMline*sku	32	Product id, code or SKU
UMline*name	255	Item name or short description
UMline*description	64k	Long description
UMline*cost	00000000.00	Cost of item per unit of measure (before tax or discount)
UMline*qty	00000000.0000	Quantity
UMline*taxable	1	Y = Taxable, N = Non-taxable
UMline*taxrate	00.000	Tax rate for line (only required for level 3 processing)
UMline*taxamount	00000000.00	Amount of tax charge for line (if left blank will be calculated from taxrate)
UMline*um	12	Unit of measure. If left blank or an invalid code is sent, EA (Each) will be used. See list of valid unit of measure codes
UMline*commoditycode	12	Commodity code (only required for level 3 processing). See http://www.unspsc.org/ for valid list of codes.
UMline*discountrate	000.000	Discount percentage for line (only required for level 3 processing)
UMline*discountamount	00000000.00	Discount amount for line (if left blank will be calculated from discountrate)

* replace the '*' with the line number. For example UMline1sku.

Visa VPAS (Verified By Visa) and UCAF (Mastercard Secure Code) Fields

The following fields are only required in participating in the Verified by Visa or Mastercard Secure Code programs:

Field	Description
UMcardauth	Set UMcardauth=true to enable cardholder authentication using the integrated gateway authentication system.
UMpares	Cardholder authentication payload. (When using integrated authentication system.)
UMxid	This field is obsolete.
UMcavv	CAVV value received from third part authenticator. If you support CAVV but customer did not participate you must send UMcavv=-1. This flags our system that you support VbV/MC Secure Code. (Only needed if not using the gateway authentication system.)
UMeci	ECI value received from third party authenticator. (Only needed if not using the gateway authentication system.)

Card Present Fields

The following fields are only required in a card present environment such as a retail store, restaurant or hotel.

Field	Description
UMcardpresent	Set UMcardpresent=true to enable card present mode. If UMmagstripe is sent, UMcardpresent will be automatically set to true.
UMmagstripe	Mag stripe data read from card. Can include Track 1, Track 2 or both. For encrypted track data, base64 encode the entire block (including masked track data) and then prepend with "enc : / /". See end to end encryption
UMdukpt	DUK/PT key for Pin-Debit transactions. The first 16 characters are the encrypted pin block, followed by the 6 character long Key Set Identifier (KSID). The remaining characters are the Pin Pad serial number and transaction counter.
UMtermtype	The type of terminal being used: POS (cash register), StandAlone (self service terminal), Unattended (ie gas pump) Unkown (defaults to Unknown)
UMmagsupport	Describes the magstripe reading capabilities of your POS hardware: yes, no, contactless, unknown (default is unknown unless magstripe has been sent).
UMcontactless	UMmagstripe was read via contactless swiper: yes or no (default is no).
UMsignature	Cardholder signature captured by pos device. Base64 encoded.

Lodging Industry

The following fields are only applicable in the hotel/lodging environment. The fields will be ignored if the merchant is not configured for this industry. Not all fields are applicable for all platforms but may be provided for reporting purposes. Contact integration support for specific platform requirements.

Field	Description
UMfolio	Folio Number
UMroomRate	Nightly Room Rate
UMnights	Number of nights
UMcheckInDate	Guest Check In Date
UMcheckOutDate	Guest Check Out Date
UMextraChargeReasons	
UMroomNumber	
UMcustomerCode	
UMlengthOfStay	
UMroomTaxRate	
UMnonRoomCharges	
UMroomTaxAmount	
UMpreferredCustomer	
UMchargeType	

UMdepartureTime	
UMarrivalTime	

CGI Result Variables

The following CGI variables are returned to your script along with any other fields you have passed to gate.php. For example, if you pass the variable FavoriteColor, gate.php will echo FavoriteColor back in the response.

Field	Description
UMstatus	Status of the transaction. The possible values are: Approved, Declined, Verification and Error.
UMauthCode	Authorization number.
UMauthAmount	Amount authorized. May be less than amount requested if UMallowPartialAuth=true
UMrefNum	Transaction reference number
UMbatch	Batch reference number. This will only be returned for sale and auth commands. Warning: The batch number returned is for the batch that was open when the transaction was initiated. It is possible that the batch was closed while the transaction was processing. In this case the transaction will get queued for the next batch to open.
UMremainingBalance	Returns the balance remaining on some prepaid and stored value cards
UMavsResult	AVS result in readable format
UMavsResultCode	AVS result code.
UMcvv2Result	CVV2 result in readable format.
UMcvv2ResultCode	CVV2 result code.
UMvpasResultCode	Verified by Visa (VPAS) or Mastercard SecureCode (UCAF) result code.
UMresult	Transaction result - A, D, E, or V (for Verification)
UMerror	Error description if UMstatus is Declined or Error.
UMerrorcode	A numerical error code.
UMacsurl	Verification URL to send cardholder to. Sent when UMstatus is verification (cardholder authentication is required).
UMpayload	Authentication data to pass to verification url. Sent when UMstatus is verification (cardholder authentication is required).
UMisDuplicate	Indicates whether this transaction is a folded duplicate or not. 'Y' means that this transaction was flagged as duplicate and has already been processed. The details returned are from the original transaction. Send UMignoreDuplicate to override the duplicate folding.
UMconvertedAmount	Amount converted to merchant's currency, when using a multi-currency processor.
UMconvertedAmountCurrency	Merchant's currency, when using a multi-currency processor.
UMconversionRate	Conversion rate used to convert currency, when using a multi-currency processor.
UMcustnum	Customer reference number assigned by gateway. Returned only if UMaddcustomer=yes.
UMresponseHash	Response verification hash. Only present if response hash was requested in the UMhash. (See Source Pin Code section for further details)
UMprocRefNum	Transaction Reference number provided by backend processor (platform), blank if not available)
UMcardLevelResult	Card level results (for Visa cards only), blank if no results provided

Revision History

```
ver. 2.17.0 -> 2.17.1:

Added UMrecurringreceipt

11/30/10
```

```
  ver. 2.16.0 -> 2.17.0:

  Added UMallowPartialAuth, UMauthAmount and UMremainingBalance for
[[developer:partialauth|Partial Authorizations]]

  Added UMauxonus and UMepcCode for check 21 transactions

  08/10/10

  ver. 2.15.0 -> 2.16.0:

  Added UMcheckimagefront, UMcheckimageback, and UMcheckimageencoding  for check
images (check 21)

  04/28/09

  ver. 2.14.0 -> 2.15.0:

  Added UMcustreceiptname

  03/31/09

  ver. 2.13.7 -> 2.14.0:

  UMprocRefNum and UMcardLevelResult

  02/18/09

  ver. 2.13.6 -> 2.13.7:

  Renamed the UMrecurring field to UMaddcustomer to make functionality clearer.

  UMrecurring will be maintained for backward compatibility

  Update UMschedule to include "disabled"  option.  This allows for customers to be
stored without enabling recurring billing

  01/27/09

  ver. 2.13.5 -> 2.13.6:

  Added UMline fields for line item detail

  Updated UMcustom fields to reflect storage for both customers and transactions

  11/10/08

  ver. 2.13.3 -> 2.13.5:

  UMhash replaces UMmd5Hash and UMmd5key variables. UMmd5Hash & UMmd5key remain under
legacy support.

  Added UMresponseHash variable

  Added UMdukpt  for debit transactions
  Added UMcomments  for extended transaction notes

  Added UMsignature  for signature capture support

  05/21/08

  ver. 2.13.2 -> 2.13.3:

  Fixed bug that caused duplicate customers to be created when duplicate folding is
enabled

  05/21/08

  ver. 2.13.1 -> 2.13.2:

  Internal fixes and additional logging/tracing capabilites

  Record customer creation date,  set created by user to (tranapi)

  04/16/08

  ver. 2.13.0 -> 2.13.1:

  Added support for check:credit transactions

  Reworked the UMcommand variable to make transaction types clearer.

  02/20/08

  ver. 2.12.1 -> 2.13.0:

  Added support for split payments

  Added UMonError for split payments

  01/15/08

  ver. 2.12.0 -> 2.12.1:

  Added UMaccounttype

  Added UMcheckformat

  Fixed storage of initial transaction when customer is created

  01/11/08

  ver. 2.11.2 -> 2.12.0:

  Added UMcustnum

  Added UMcontactless

  Updated UMmagsupport to include contactless option
```

```
05/16/07

ver. 2.11.1 -> 2.11.2:

Documented UMchecknum

05/08/07

ver. 2.11.0 -> 2.11.1:

Documented the CAVV=-1 flag for VbV/MC Secure code transactions

Documented the UMcustom fields

03/06/07

ver. 2.10.1 -> 2.11.0:

Added UMconvertedAmount, UMconvertedAmountCurrency and UMconversionRate

03/06/07

ver. 2.10.0 -> 2.10.1:

Added creditvoid command

01/29/07

ver. 2.9.6 -> 2.10.0:

Added UMcurrency

01/18/07

ver. 2.9.5 -> 2.9.6:

Added UMignoreDuplicate and UMisDuplicate

11/28/06

ver. 2.9.4 -> 2.9.5:

Added custom fields for recurring billing

Added UMclerk, UMtranterm and UMresttable

9/12/06

ver. 2.9.3 -> 2.9.4:

Copy cardholder from magstrip if not provided in UMname

5/20/06

ver. 2.9.2 -> 2.9.3:
```

```
Added UMnontaxable flag for indicating nontaxable instruction

Added UMvpasResultCode result field with value of vpas aut (platform dependant)

3/28/05

ver. 2.9.1 -> 2.9.2:

UMcustid now saves to the recurring billing customer number field

12/27/04

ver. 2.9.0 -> 2.9.1:

Added next option to UMstart

Corrected UMversion response variable to return 2.9

11/23/04

ver. 2.8.3 -> 2.9.0:

Added input fields for UMtip, UMshipping, UMdiscount, UMorderid

Added response field for UMerrorcode

11/10/04

ver. 2.8.2 -> 2.8.3:

Added support for loading check based recurring billing entries

09/02/04

ver. 2.8.0 -> 2.8.2:

Commas are now stripped from amount (ie 4,500.09 will now be handled correctly)

UMredirDeclined url now gets appended with the result fields.

04/28/04

ver. 2.7.1 -> 2.8.0:

Card present support added with new fields: UMcardpresent, UMmagstripe,

 UMtermtype, UMmagsupport

Added UMsoftware field.

Added UMbatch result field

03/26/04

ver. 2.7.0 -> 2.7.1:
  VBV can now be rerun using only UMrefNum

  02/14/04

  ver. 2.6.1 -> 2.7.0:

  Added UMmd5hash and UMmd5key, implementing new source pin feature

  Updated UMversion to 2.7.

  02/07/04

  ver. 2.6.0 -> 2.6.1:

  Fixed api to return RefNum for credits (blank refnum was being returned).

  Fixed void command to void transaction via refnum (this is the recommended

   way to void a transaction).

  Updated UMversion to 2.6.

  02/07/04

  ver. 2.5.3 -> 2.6.0:

  Fixed API to return AVS and CVV2 results even when error is returned (ie a

   fraud module block).

  Added new UMavsResultCode and UMcvv2ResultCode fields.

  01/27/04

  ver. 2.5.2 -> 2.5.3:

  Added support for biannually in Recurring Billing schedule method.

  10/22/03

  ver. 2.5.1 -> 2.5.2:

  Added Mastercard UCAF to the Verified by Visa Section. Both are handled
identicatlly.

  08/08/03

  ver. 2.5.0 -> 2.5.1:

  Added UMtax and UMponum fields for corporate purchasing cards.

  08/08/03

  ver. 2.4.0 -> 2.5.0:
```

```
  Added support for VPAS (Verified by Visa) fields: UMcardauth,UMpares, UMcavv, UMxid
and UMeci

  07/26/03

  ver. 2.3.2 -> 2.4.0:

  Added new command "capture" which can be used to settle a transaction that was
authorized

   using the "authonly" command. This differs from the "postauth" command which
settles a

   transaction authorized "offline" (ie telephone authorization). To use capture, the
UMrefNum

   that was obtained from authonly must be passed back.

  The "preauth" command has been rename "authonly" to avoid confusion. "preauth" is
still

   supported for backward compatibility.

  06/25/03

  ver. 2.3.1 -> 2.3.2 :

  UMbillamount added to the Recurring Billing functionality allowing the merchant to
separate

   things like "setup" charge amount from the recurring charge amount.

  03/28/03

  ver. 2.3.0 -> 2.3.1 :

  Recurring Capabilities added via API. Please refer to the Recurring Billing section
of the API

   to the newly added fields.

  11/22/02

* Unordered List Item
```