Braintree Payment Solutions API Documentation

Table of Contents

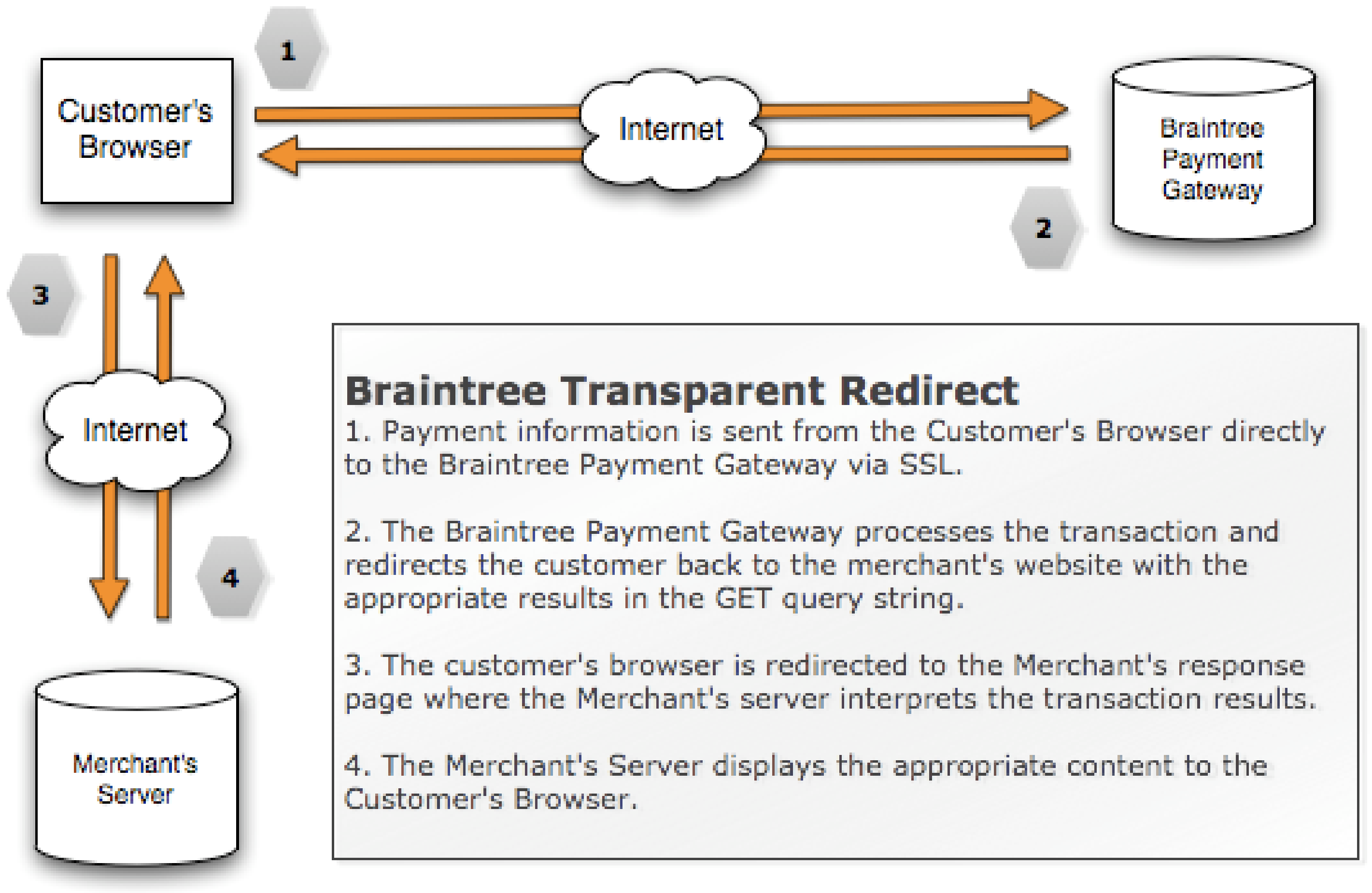
Transparent Redirect (TR) Payment API

The TR API was designed to eliminate the handling and processing of credit card data. When integrated, no credit card data will enter a merchant’s environment which will greatly reduce the scope of PCI Compliance and increase security. Using TR, merchants have the same look and feel of a normal check out process and have complete control over credit card data. TR is not a hosted page solution. It’s entirely hidden from the end user.

Note, only sale, authorization and capture transactions can be done via Transparent Redirect. Refunds, voids and credits should be done via the Direct Post method.

Merchants may also use our Vault to securely store the credit card data for subsequent transactions, see our Vault API documentation.

How Transparent Redirect Works



Braintree Transparent Redirect

1. Payment information is sent from the Customer's Browser directly to the Braintree Payment Gateway via SSL.

2. The Braintree Payment Gateway processes the transaction and redirects the customer back to the merchant's website with the appropriate results in the GET query string.

3. The customer's browser is redirected to the Merchant's response page where the Merchant's server interprets the transaction results.

4. The Merchant's Server displays the appropriate content to the Customer's Browser.

Please note that the merchant may be required to use the **Transparent Redirect** *method if they are enrolled in 3D Secure Technology (Verified by Visa / MasterCard SecureCode).*

Upon the cardholder's request to checkout, the merchant displays a form to the cardholder that contains or collects the appropriately named variables (as defined below). The additional redirect variable (as defined below) indicates to the Payment Gateway where the cardholder should be directed and essentially what file will parse the transaction results. This will typically be passed as a hidden field. This form should also have an action of the Payment Gateway's URL.

When the cardholder submits the form, the data is posted to the Payment Gateway (**Step 1**). The Payment Gateway processes the transaction and sends a header redirect back to the cardholder (**Step 2**).

In **Step 3**, the cardholder requests the URL defined by the previously posted redirect variable and the results of the transaction are included in the GET query string. The merchant must create a script that parses these response variables, updates their database/ordering system appropriately, and displays the receipt or decline message to the cardholder. (**Step 4**)

Generating the Hash

To completely protect the authenticity of a merchant's API request and response, security keys must be used in the Transparent Redirect Method. The key and the key Id can be found in the gateway under Options/Security Keys. If you are just testing, you can use the test key: 844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7 and key id: 1247307. You may not use a username and password for Transparent Redirect API Authentication.

The inbound hash protects the authenticity of the data in the request. The `key` and `key_id` replace a standard username and password for best-in-class security authentication.

Field Requirements

`redirect`—The URL the customer will be redirected to after a transaction is processed by the Payment Gateway. This URL must also parse and respond to the response variables included in the GET query upon the redirect. Format: https://www.domain.tld/

`hash`—MD5 variable hash generated to protect the authenticity of a request. The hash is composed of the string value of the following variables delimited by pipes and hashed with an *MD5 algorithm*.

```
orderid|amount|customer_vault_id|time|Key
```

The `key` can be obtained under the Security Keys section within the Gateway Options and is associated with the passed `key_id` above.

`customer_vault_id` should be included in the hash only when you are passing the variable `customer_vault_id` in your post. If you are not referencing a vault id in your transaction, do not include this variable in your hash.

Vault Transaction Example

```
123456-xfj|10.00|123456|20080516190549|844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7
```

Non-Vault Transaction Example

```
123456-xfj|10.00|20080516190549|844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7
```

- `orderid`—This is the order id from your application, you can generate it in any manner you choose. It can be in any format as long as it does not contain any non-ASCII characters. For this example, it is `123456-xfj`. If you choose to leave `orderid` blank, you may, but remember to include an extra pipe(|) at the beginning.

- `amount`—Total amount to be charged. Format: x.xx. In this example, it is $10.00.

- `customer_vault_id`—Required only when passing `customer_vault_id`. In the example, the Vault ID is 123456. When not passing variable `customer_vault_id`, do not include it in the hash.

- `time`—This value is associated with the hash to protect them from being used over an extended period of time. The gateway will reject times that are over 15 minutes old. The timezone used must by UTC (Universal Coordinated Time). Format: YYYYMMDDHHMMSS. In this example, the time is 20080516190549.

- `key`—This is the value obtained from the gateway. Make sure this is the `key`, and not the `key_id`. For this example, I used the test key, 844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7.

Security Recommendations

You can always use the Default Key that gets generated by Braintree when your account is created, but we'd recommend creating your own key and name it 'api'. For extra security, change your key and key id once a month. Your `key_id` could be seen if the customer looks at the generated HTML of the form, but that's not really a concern as long as you don't allow anyone to see your `key`. There is no way to completely hide your user key completely as it will be stored on your system. You can obscure it with key encrypting or create a password to your webserver. A compromised key will mean that someone could potentially send money from your account (issue credits). Credits are turned off by default, but it is something to be aware of if they are enabled.

You may also want to setup IP Restrictions in the Gateway. This means that even if someone gets your key, they will have to make requests from your server to make any transactions. IP Restrictions can be set in the gateway under Options -> IP Restrictions.

Generating Form HTML

Now that we know that our hash is correct we can start to create our form. We're going to start with an empty form with the action set to the Braintree Payment Gateway and a submit button.

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

From there we're going to add the hidden fields that are necessary to help authenticate the hash. We need to add the `orderid`, `amount`, `customer_vault_id` (for vault transactions only), `time`, `hash`, and `key_id`. We need to pass these values as hidden inputs so we can generate a matching hash based upon the values that you send. `orderid` and `amount` can be left blank depending on the transaction, but the variables must be passed.

NOTE We use the `key_id` that gets sent to look up your `key` value to hash so make sure your `key` is never visible. From we've got above and our hidden inputs will make the form look like this:

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="27cc9375a73a8046707b18112dcaff6c"
name="hash"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

The next thing to add is a transaction type, in this example we will include `type=sale`.

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="27cc9375a73a8046707b18112dcaff6c"
name="hash"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Now add the fields for the customer to input their credit card information into. The bare minimum required for a sale transaction is `ccnumber` and `ccexp`.

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="27cc9375a73a8046707b18112dcaff6c" 
name="hash"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value="1010" name="ccexp"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

This is a basic transaction example. View the required and optional variables for each transaction type below. You can also create custom variables.

Variables

Note: Only ASCII characters are accepted for the values.

Sale

Sales can be done via Transparent Redirect or Direct Post.

Transaction sales are submitted and immediately flagged for settlement.
[Required Variables]

`type=sale`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`zip`

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing Total tax amount. Format: x.xx

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Authorization

Authorizations can be done via Transparent Redirect or Direct Post Methods.

Authorizations are authorized immediately, but are not flagged for settlement. After a successful authorization, funds will be held in the customer’s account for 3-7 business days. In order to submit these transactions for settlement, the merchant must submit a “type=capture” request. The merchant also has the option to void these authorizations before a capture is submitted. If the merchant does not run a subsequent capture or void, the authorization will fall of the customer’s account after 3-7 business days. Visa assesses a small processing fee for allowing this to happen.

When part of an authorization is captured, the rest of the original authorization is automatically voided. [Required Variables]

`type=auth`

`ccnumber`

Valid credit card number.

`ccexp` Format: MMYY

Credit card expiration date for the credit card passed. (1010 = October, 2010)

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state Format: CC (2 Character abbrev.)

`zip`

`country` Format:

Card billing country CC (ISO-3166) (ie. US)

`phone`

`email`

[Optional Variables]

`orderdescription`

`orderid`

Capture

Captures can be done via Transparent Redirect or Direct Post Methods.

Transaction captures flag existing authorizations for settlement. Only authorizations can be captured. Captures can be submitted for an amount equal to or less than the original authorization. When an authorization has been partially captured, the rest of the original authorization is automatically voided.
[Required Variables]

`type=capture`

`amount`

Total amount to be charged. Format: x.xx

`transactionid`

This value was passed in the response to the previous gateway authorization transaction.

[Optional Variables]

`descriptor`

Set payment descriptor (On supported processors)

`descriptor_phone`

Set payment descriptor phone (On supported processors)

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Validate

Validate can be done via Transparent Redirect or Direct Post methods.

A “type=validate” will check for card validity without ever holding funds on the cardholder’s account. A validate request cannot not determine whether the customer has enough funds to complete a transaction or not, nor will it return AVS or CVV data. Currently, Visa and MasterCard are the only two credit cards that accept validate requests. For other card types, an authorization must be used to check for validity.

[Required Variables]

`type=validate`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`processor_id`

Required if account has multiple processor ids. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`address1`

Cardholder’s billing address.

`zip`

[Optional Variables]

`amount`

If passed, the amount must be 0.00.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

`product_sku_1`

Required for recurring billing plans only.

`orderid`

`orderdescription`

Void

Voids can only be done using the Direct Post method, not Transparent Redirect.

Voiding a transaction will cancel an existing sale or captured authorization from actually charging the card. In addition, non-captured authorizations can be voided to prevent any future capture. Note however, that the amount is still reserved on the card and will take a few days to expire. You will have to call the issuing bank to request that the authorization be removed if the customer does not want to wait for it to expire on its own.

Voids can only occur if the transaction has not been settled; settled transactions should be refunded.
[Required Variables]

`type=void`

`transactionid`

This value was passed in the response of the previous gateway transaction that needs to be void.

Refund

Refunds can only be done using the Direct Post method, not Transparent Redirect.

Transaction refunds will reverse a previously settled transaction. If the transaction has not been settled, it must be voided instead of refunded.
[Required Variables]

`type=refund`

`amount`

Total amount to be refunded. Format: x.xx

`transactionid`

This value was passed in the response of the previous Gateway Transaction that needs to be refunded.

Credit

Credits can only be done using the Direct Post method, not Transparent Redirect.

Transaction credits apply a negative amount to the cardholder's card. In most situations, credits are disabled as transaction refunds should be used instead.
[Required Variables]

`type=credit`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be refunded. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder's billing address.

`city`

`state`

Card billing state. Format: CC (2 Character abbrev.)

`zip`

`country`

Card billing country Format: CC (ISO-3166)(ie. US)

`phone`

`email`

[Optional Variables]

Update

Updates can only be done using the Direct Post method, not Transparent Redirect.

Transaction updates can be used to update previous transactions with specific order information, such as a tracking number and shipping carrier.
[Required Variables]

`type=update`

`transactionid`

This value was passed in the response of the previous Gateway Transaction.

[Optional Variables]

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

Response URL

The last item we’re going to add is the response URL. This hidden value should be the URL in your application that is capable of accepting and parsing a response from the gateway. You need to include the full domain for this value.

NOTE You can append any other values such as a session id or some such other identifier. For this example, I’m going to use the response url of http://example.tld/gateway_response.

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="27cc9375a73a8046707b18112dcaff6c"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value="1010" name="ccexp"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Receiving a Response

Return Hash

After an API call has been made, a unique hash will immediately be returned in the response. The response hash protects the authenticity of the data returned to the merchant’s servers. Performing an additional md5 check on the response hash will ensure that the response is authentic. Specifically, the following fields are protected by the response hash:

- `orderid`
- `amount`
- `response`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `customer_vault_id` (when customer_vault= is used in the originating transaction)
- `time` (The merchant’s application can use this time to determine the staleness of the response.)
- `key`

The hash should be the values of the following variables delimited by pipes and hashed with an md5 algorithm.

Examples

Vault Transactions
orderid|amount|response|transactionid|avsresponse|cvvresponse|customer_vault_id|time|key

Non-Vault Transactions **orderid|amount|response|transactionid|avsresponse|cvvresponse|time|key**

Customizing Your Response Variables

You can choose to customize the variables you receive in the response to an API transaction. This functionality is particularly useful in pre-populating fields when a transaction fails, so a customer does not need to re-enter all of the information.

To do this, you must first log in to the control panel, then click “Options” in the left sidebar. Next, click on “API Configuration.” Here, you will be able to add variables to your response. You cannot remove any of the default variables from the response to a transaction. By default, the variables returned include:

- `response`
- `responsetext`
- `response_code`
- `authcode`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `orderid`
- `type`
- `customer_vault_id` (only returned by default when adding a customer to the vault)

Variables that cannot be added to the response include:

- `ccnumber`
- `ccexp`
- `cvv`

Default Response Variables

Variable Name	Description
response	"1" = Transaction Approved "2" = Transaction Declined "3" = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: "AUTH DECLINED" (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) "LOST/STOLEN CARD" (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and "CALL VOICE OPER" (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree's transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer vault id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

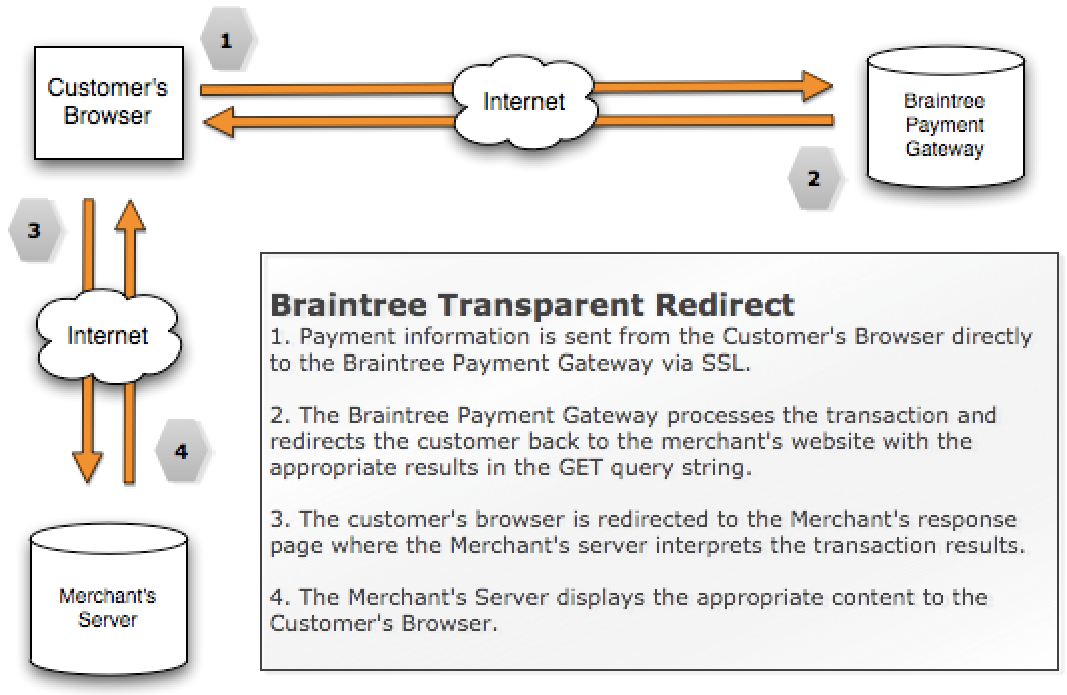
Transparent Redirect (TR) Vault API

How Transparent Redirect Works

The Vault works with either Direct Post or Transparent Redirect and enables the remote storage of credit card data. Credit card information can be submitted to the Vault with or without completing a transaction and receive a ‘token’ in response. Merchants can then use the token for all subsequent transactions including sale, auth, void and returns. Tokens can be 36 alphanumeric characters. Any sensitive information can be stored in the Vault including name, address, bank account information, social security and driver license numbers, image documents, media files, etc.

Note, only sale, authorization and capture transactions can be done via Transparent Redirect. Refunds, voids and credits should be done via the Direct Post method.

When integrated, no credit card data will enter or be stored a merchant’s environment, which will greatly reduce the scope of PCI Compliance and increase security.



Braintree Transparent Redirect

1. Payment information is sent from the Customer's Browser directly to the Braintree Payment Gateway via SSL.

2. The Braintree Payment Gateway processes the transaction and redirects the customer back to the merchant's website with the appropriate results in the GET query string.

3. The customer's browser is redirected to the Merchant's response page where the Merchant's server interprets the transaction results.

4. The Merchant's Server displays the appropriate content to the Customer's Browser.

Please note that the merchant may be required to use the **Transparent Redirect** *method if they are enrolled in 3D Secure Technology (Verified by Visa / MasterCard SecureCode).*

Upon the cardholder’s request to checkout, the merchant displays a form to the cardholder that contains or collects the appropriately named variables (as defined below). The additional redirect variable (as defined below) indicates to the Payment Gateway where the cardholder should be directed and essentially what file will parse the transaction results. This will typically be passed as a hidden field. This form should also have an action of the Payment Gateway’s URL.

When the cardholder submits the form, the data is posted to the Payment Gateway (**Step 1**). The

Payment Gateway processes the transaction and sends a header redirect back to the cardholder (**Step 2**).

In **Step 3**, the cardholder requests the URL defined by the previously posted redirect variable and the results of the transaction are included in the GET query string. The merchant must create a script that parses these response variables, updates their database/ordering system appropriately, and displays the receipt or decline message to the cardholder. (**Step 4**)

Making a Request

Transaction details should be delivered to the Braintree Payment Gateway using a POST HTTP method with the appropriate variables with the request. All credit card or account information you pass with it will be stored in the Vault and you will be returned a token which you can use for future transactions.

URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
</form>
```

Generating the Hash

To completely protect the authenticity of a merchant's API request and response, security keys must be used in the Transparent Redirect Method. The key and the key Id can be found in the gateway under Options/Security Keys. If you are just testing, you can use the test key: 844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7 and key id: 1247307. You may not use a username and password for Transparent Redirect API Authentication.

The inbound hash protects the authenticity of the data in the request. The `key` and `key_id` replace a standard username and password for best-in-class security authentication.

Field Requirements

`redirect`—The URL the customer will be redirected to after a transaction is processed by the Payment Gateway. This URL must also parse and respond to the response variables included in the GET query upon the redirect. Format: https://www.domain.tld/

`hash`—MD5 variable hash generated to protect the authenticity of a request. The hash is composed of the string value of the following variables delimited by pipes and hashed with an *MD5 algorithm*.

```
orderid|amount|customer_vault_id|time|Key
```

The `key` can be obtained under the Security Keys section within the Gateway Options and is associated with the passed `key_id` above.

`customer_vault_id` should be included in the hash only when you are passing the variable `customer_vault_id` in your post. If you are not referencing a vault id in your transaction, do not include this variable in your hash.

Vault Transaction Example

```
123456-xfj|10.00|123456|20080516190549|844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7
```

Non-Vault Transaction Example

```
123456-xfj|10.00|20080516190549|844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7
```

- `orderid`—This is the order id from your application, you can generate it in any manner you choose. It can be in any format as long as it does not contain any non-ASCII characters. For this example, it is `123456-xfj`. If you choose to leave `orderid` blank, you may, but remember to include an extra pipe(|) at the beginning.

- `amount`—Total amount to be charged. Format: x.xx. In this example, it is $10.00.

- `customer_vault_id`—Required only when passing `customer_vault_id`. In the example, the Vault ID is 123456. When not passing variable `customer_vault_id`, do not include it in the hash.

- `time`—This value is associated with the hash to protect them from being used over an extended period of time. The gateway will reject times that are over 15 minutes old. The timezone used must by UTC (Universal Coordinated Time). Format: YYYYMMDDHHMMSS. In this example, the time is 20080516190549.

- `key`—This is the value obtained from the gateway. Make sure this is the `key`, and not the `key_id`. For this example, I used the test key, 844wfNN5FGuGS7wtKfQsY6k6ZxAv6Ff7.

Security Recommendations

You can always use the Default Key that gets generated by Braintree when your account is created, but we'd recommend creating your own key and name it 'api'. For extra security, change your key and key id once a month. Your `key_id` could be seen if the customer looks at the generated HTML of the form, but that's not really a concern as long as you don't allow anyone to see your `key`. There is no way to completely hide your user key completely as it will be stored on your system. You can obscure it with key encrypting or create a password to your webserver. A compromised key will mean that someone could potentially send money from your account (issue credits). Credits are turned off by default, but it is something to be aware of if they are enabled.

You may also want to setup IP Restrictions in the Gateway. This means that even if someone gets your key, they will have to make requests from your server to make any transactions. IP Restrictions can be set in the gateway under Options -> IP Restrictions.

Example Authentication:

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="010c018537d89147d10fd0c65eb11e2e"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Adding a customer to the Vault

Pass the required variables:

`customer_vault=add_customer`

`orderid` (leave blank)

`amount` (leave blank)

`ccnumber` Valid credit card number. Pass if adding a credit card.

`ccexp` Credit card expiration date if adding a credit card. Format: MMYY (1010 = October, 2010)

`customer_vault_id` Specifies the name of the vault ID to be created. Values can be up to 32 alphanumeric characters. If you do not pass a customer_vault_id, the Vault will randomly generate one and return it in the response.

[Recommended Variables]

`firstname`

`lastname`

`address1`

`city`

`state` Card billing state Format: NY (2 Character abbrev.)

`zip`

`country` Card billing country. Format: CC (ISO-3166). Example country=US for the United States

`phone`

`email`

[Optional Variables]

`company`

`address2`

`fax`

`website`

See additional optional variables here.

Example

In this example, we are adding the customer to the vault and specifying the vault ID of 123456. Since we are not running a transaction, leave `orderid` and `amount` blank. They must be included for the hash:

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="" name="orderid"/>
  <input id="amount" type="hidden" value="" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="bccb8777b7f94d20f28bf1850df05847"
name="hash"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value="1010" name="ccexp"/>
  <input id="customer_vault" type="hidden" value="add_customer"
name="customer_vault"/>
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Verifying AVS/CVV Upon Adding a Customer to the Vault

If the merchant is issuing a transaction while simultaneously creating a vault id, the customer will automatically be subjected to the AVS and CVV settings enabled on the account.

If no transaction is being issued when a customer is added to the vault, the merchant can issue a “type=auth” and “amount=1.00” request in the same string containing “customer_vault=add_customer”. If the customer’s information passes the AVS and CVV rejection settings, a vault id will be created. If the information does not, no vault id will be created.

It is important to note that Visa assesses a small processing fee for running an authorization on a card without a subsequent capture or void.

Issuing transactions using a Vault id

Pass `customer_vault_id=` along with the required transaction variables for the transaction type.

Note: Only ASCII characters are accepted for the values.

Sale

Transaction sales are submitted and immediately flagged for settlement.
[Required Variables]

`type=sale`

`customer_vault_id`

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder's billing address.

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`zip`

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

[Optional Variables]

`orderid`

`orderdescription`

`product_sku_1`

Required for recurring billing plans only.

`billing_method`

Required only when issuing a recurring billing transaction. Value should be set to “recurring”

`tax`

Required for Level 2 processing. Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value is either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Authorization

Authorizations are authorized immediately, but are not flagged for settlement. After a successful authorization, funds will be held in the customer’s account for 3-7 business days. In order to submit these transactions for settlement, the merchant must submit a “type=capture” request. The merchant also has the option to void these authorizations before a capture is submitted. If the merchant does not run a subsequent capture or void, the authorization will fall of the customer’s account after 3-7 business days. Visa assesses a small processing fee for allowing this to happen.

When part of an authorization is captured, the rest of the original authorization is automatically voided.
[Required Variables]

`type=auth`

`customer_vault_id`

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state Format: CC (2 Character abbrev.)

`zip`

`country` Format:

Card billing country CC (ISO-3166) (ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in format: ups / fedex / dhl / usps

See additional optional variables here.

Capture

Transaction captures flag existing authorizations for settlement. Only authorizations can be captured. Captures can be submitted for an amount equal to or less than the original authorization. When an authorization has been partially captured, the rest of the original authorization is automatically voided.
[Required Variables]

`type=capture`

`amount`

Total amount to be charged. Format: x.xx

`transactionid`

This value was passed in the response to the previous gateway authorization transaction.

[Optional Variables]

`descriptor`

Set payment descriptor (On supported processors)

`descriptor_phone`

Set payment descriptor phone (On supported processors)

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Validate

A “type=validate” will check for card validity without ever holding funds on the cardholder’s account. A validate request cannot not determine whether the customer has enough funds to complete a transaction or not, nor will it return AVS or CVV data. Currently, Visa and MasterCard are the only two credit cards that accept validate requests. For other card types, an authorization must be used to check for validity.

[Required Variables]

`type=validate`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`processor_id`

Required if account has multiple processor ids. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`address1`

Cardholder’s billing address.

`zip`

[Optional Variables]

`amount`

If passed, the amount must be 0.00.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

`product_sku_1`

Required for recurring billing plans only.

`orderid`

`orderdescription`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

Void

Transaction voids will cancel an existing sale or captured authorization. In addition, non-captured authorizations can be voided to prevent any future capture. Voids can only occur if the transaction has not been settled. Voids can only be done via Direct Post.

[Required Variables]

`type=void`

`transactionid`

This value was passed in the response of the previous gateway transaction that needs to be void.

Refund

Transaction refunds will reverse a previously settled transaction. If the transaction has not been settled, it can be voided instead of refunded. Refunds can only be done via Direct Post.

[Required Variables]

`type=refund`

`amount`

Total amount to be refunded. Format: x.xx

`transactionid`

This value was passed in the response of the previous Gateway Transaction that needs to be refunded.

Credit

Transaction credits apply a negative amount to the cardholder’s card. In most situations, credits are disabled as transaction refunds should be used instead. Credits can only be done via Direct Post.
[Required Variables]

`type=credit`

`customer_vault_id`

`amount`

Total amount to be refunded. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state. Format: CC (2 Character abbrev.)

`zip`

`country`

Card billing country Format: CC (ISO-3166)(ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

See additional optional variables here.

Update

Transaction updates can be used to update previous transactions with specific order information, such as a tracking number and shipping carrier.
[Required Variables]

`type=update`

`transactionid`

This value was passed in the response of the previous Gateway Transaction.

[Optional Variables]

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Example

In this example, we are issuing a sale transaction for Vault id 123456 for 10.00.

Since no sensitive data is being passed, this action can be done using Direct Post:

```
username=testapi&password=password1&type=sale&amount=10.00&customer_vault_id=123456
```

Transparent Redirect Example:

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="010c018537d89147d10fd0c65eb11e2e"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

If successful, the response code of 1 will be returned along with customer_vault_id. If a response of 2 or 3 is returned, the transaction failed and the customer has not been added to the vault.

Updating an existing Vault record

“Updating” refers to both adding information on an existing Vault record and updating information already stored on a Vault record. It is important to note that the gateway will erase data for any variable that is passed with a null value on an update customer request. For this reason, we strongly recommend you only pass the variables you wish to update when issuing this type of request.

Pass the required variables:

`customer_vault=update_customer`

`customer_vault_id` Specifies the specific customer record to update

Plus, the variable you would like to add/update.

[Recommended Variables]

`firstname`

`lastname`

`address1`

`city`

`state` Card billing state Format: NY (2 Character abbrev.)

`zip`

`country` Card billing country. Format: CC (ISO-3166). Example country=US for the United States

`phone`

`email`

[Optional Variables]

`orderdescription`

`orderid`

`company`

`address2`

`fax`

`website`

See additional optional variables here.

Example

In this example, we are updating the expiration date of vault ID 123456.

Since no sensitive data is being passed, this action can be done using Direct Post:

```
username=testapi&password=password1&customer_vault=update_customer&customer_vault_i
d=123456&ccexp=1012
```

Using Transparent Redirect, leave `orderid` and `amount` blank, but include them as they are required for the hash:

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="" name="orderid"/>
  <input id="amount" type="hidden" value="" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="bccb8777b7f94d20f28bf1850df05847"
name="hash"/>
  <input id="ccexp" type="text" value="1012" name="ccexp"/>
  <input id="customer_vault" type="hidden" value="update_customer"
name="customer_vault"/>
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Deleting a Vault record

Pass the required variables:

`customer_vault=delete_customer`

`customer_vault_id` Specifies the specific customer record to delete.

Example

You can delete a customer via the Direct Post method only. In this example, we are deleting customer vault ID 123456:

```
username=testapi&password=password1&customer_vault=delete_customer&customer_vault_i
d=123456
```

Adding a customer and issuing a transaction

Pass the required variables:

`customer_vault=add_customer`

`amount` Format x.xx

`type` Format “sale” or “auth”

Example

In this example, a $10.00 sale is being submitted and the customer is being added to the vault. The vault ID is being specified as 123456.

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php">
  <input id="customer_vault" type="hidden" value="add_customer"
name="customer_vault"/>
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="1247307" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="10.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="010c018537d89147d10fd0c65eb11e2e"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value="1010" name="ccexp"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

If successful, the response code of 1 will be returned along with `customer_vault_id`. If a response of 2 or 3 is returned, the transaction failed and the customer has not been added to the vault.

Receiving a Response

Return Hash

After an API call has been made, a unique hash will immediately be returned in the response. The response hash protects the authenticity of the data returned to the merchant's servers. Performing an additional md5 check on the response hash will ensure that the response is authentic. Specifically, the following fields are protected by the response hash:

- `orderid`
- `amount`
- `response`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `customer_vault_id` (when customer_vault= is used in the originating transaction)
- `time` (The merchant's application can use this time to determine the staleness of the response.)
- `key`

The hash should be the values of the following variables delimited by pipes and hashed with an md5 algorithm.

Examples

Vault Transactions
orderid|amount|response|transactionid|avsresponse|cvvresponse|customer_vault_id|time|key

Non-Vault Transactions **orderid|amount|response|transactionid|avsresponse|cvvresponse|time|key**

Customizing Your Response Variables

You can choose to customize the variables you receive in the response to an API transaction. This functionality is particularly useful in pre-populating fields when a transaction fails, so a customer does not need to re-enter all of the information.

To do this, you must first log in to the control panel, then click “Options” in the left sidebar. Next, click on “API Configuration.” Here, you will be able to add variables to your response. You cannot remove any of the default variables from the response to a transaction. By default, the variables returned include:

- `response`
- `responsetext`
- `response_code`
- `authcode`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `orderid`
- `type`
- `customer_vault_id` (only returned by default when adding a customer to the vault)

Variables that cannot be added to the response include:

- `ccnumber`
- `ccexp`
- `cvv`

Default Response Variables

Variable Name	Description
response	“1” = Transaction Approved “2” = Transaction Declined “3” = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: “AUTH DECLINED” (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) “LOST/STOLEN CARD” (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and “CALL VOICE OPER” (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree’s transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer vault id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

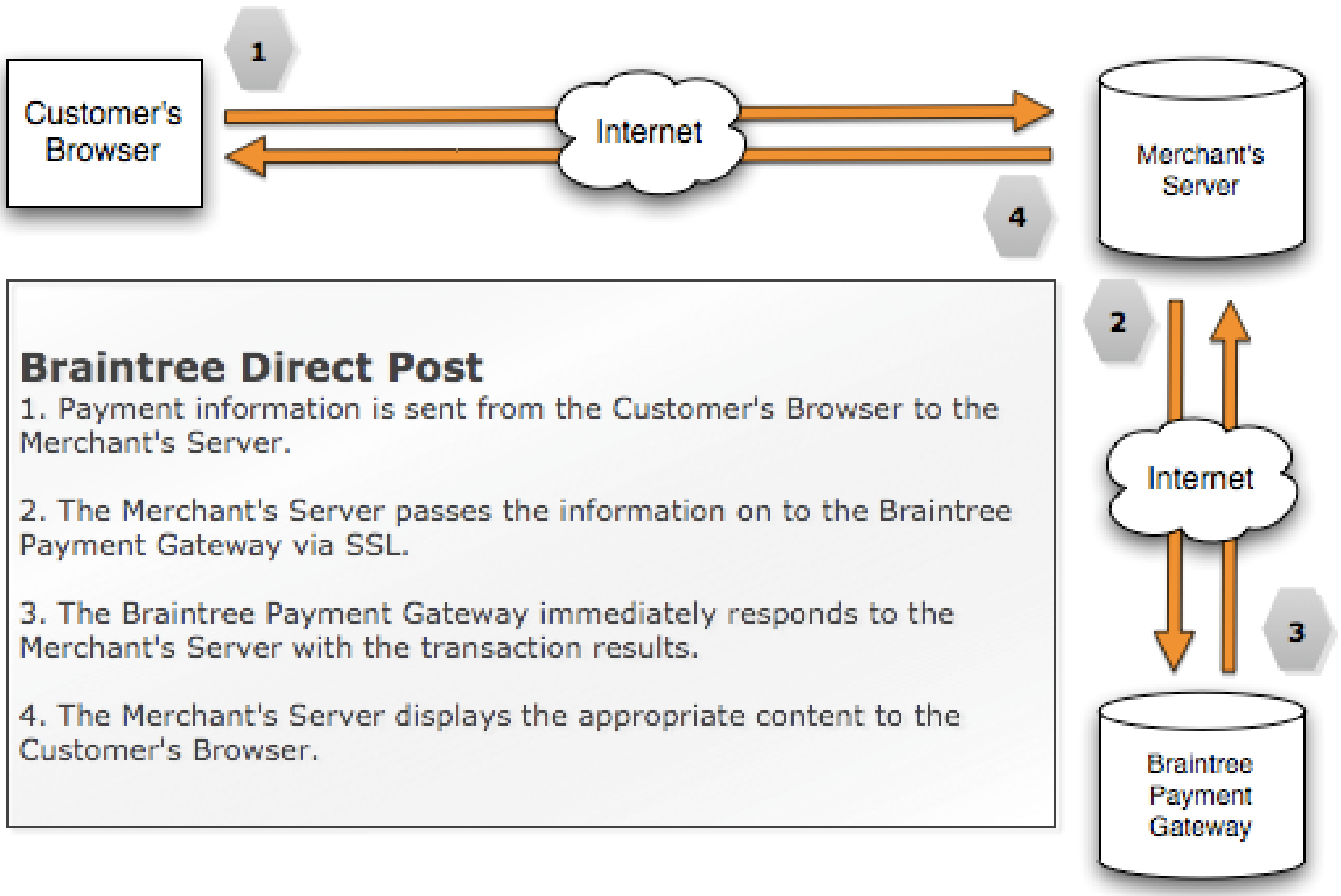
CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

Direct Post (DP) Payment API

The Direct Post method sends messages to the Payment Gateway's server using the standard HTTP protocol over an SSL connection. Please note that the full credit card data is passing through the merchant's server. Braintree has developed the Transparent Redirect method to eliminate this security risk.

How Direct Post Works



Braintree Direct Post

1. Payment information is sent from the Customer's Browser to the Merchant's Server.

2. The Merchant's Server passes the information on to the Braintree Payment Gateway via SSL.

3. The Braintree Payment Gateway immediately responds to the Merchant's Server with the transaction results.

4. The Merchant's Server displays the appropriate content to the Customer's Browser.

In the Direct Post method, the communications with the cardholder (**Steps 1 and 4**) are developed completely by the merchant and therefore are not defined by the Payment Gateway. **Step 1** should simply collect the payment data from the cardholder and **Step 4** should display the appropriate transaction receipt or declined message.

In **Step 2**, transaction details should be delivered to the Payment Gateway using the POST method with the appropriate variables defined below posted along with the request.

In **Step 3**, the transaction responses are returned in the body of the HTTP response in a query string name/value format delimited by ampersands. For example:

```
variable1=value1&variable2=value&variable3=value3
```

Making A Request

To make a request, determine transaction type (ie. sale, authorization, void, etc.) and pass the credit card number, expiration date and amount, using POST HTTP in the format `variable1=value1&variable2=value2&variable3=value3`.

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

To authenticate against the Direct Post API, simply pass your username and password as two of the parameter variables in the request.

`username` (required)— Username assigned to the merchant account. This username is also a username to access the Braintree Virtual Terminal. Username should be hidden from customers in all situations.

`password` (required)— Password for the specified username. This password is also the password to access the Virtual Terminal. Password should be hidden from the customer in all situations.

For example, to do a $10.00 sale:

```
username=demoapi&password=password1&ccnumber=4111111111111111&ccexp=1010
&type=sale&amount=10.00
```

Variables

Note: Only ASCII characters are accepted for the values.

Sale

Sales can be done via Transparent Redirect or Direct Post.

Transaction sales are submitted and immediately flagged for settlement.
[Required Variables]

`type=sale`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`zip`

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Authorization

Authorizations can be done via Transparent Redirect or Direct Post Methods.

Authorizations are authorized immediately, but are not flagged for settlement. After a successful authorization, funds will be held in the customer’s account for 3-7 business days. In order to submit these transactions for settlement, the merchant must submit a “type=capture” request. The merchant also has the option to void these authorizations before a capture is submitted. If the merchant does not run a subsequent capture or void, the authorization will fall of the customer’s account after 3-7 business days. Visa assesses a small processing fee for allowing this to happen.

When part of an authorization is captured, the rest of the original authorization is automatically voided. [Required Variables]

`type=auth`

`ccnumber`

Valid credit card number.

`ccexp` Format: MMYY

Credit card expiration date for the credit card passed. (1010 = October, 2010)

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state Format: CC (2 Character abbrev.)

`zip`

`country` Format:

Card billing country CC (ISO-3166) (ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`tax`

Required for Level 2 Processing. Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in format: ups / fedex / dhl / usps

See additional optional variables here.

Capture

Captures can be done via Transparent Redirect or Direct Post Methods.

Transaction captures flag existing authorizations for settlement. Only authorizations can be captured. Captures can be submitted for an amount equal to or less than the original authorization. When an authorization has been partially captured, the rest of the original authorization is automatically voided. [Required Variables]

`type=capture`

`amount`

Total amount to be charged. Format: x.xx

`transactionid`

This value was passed in the response to the previous gateway authorization transaction.

[Optional Variables]

`descriptor`

Set payment descriptor (On supported processors)

`descriptor_phone`

Set payment descriptor phone (On supported processors)

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Validate

Validate can be done via Transparent Redirect or Direct Post methods.

A “type=validate” will check for card validity without ever holding funds on the cardholder’s account. A validate request cannot not determine whether the customer has enough funds to complete a transaction or not, nor will it return AVS or CVV data. Currently, Visa and MasterCard are the only two credit cards that accept validate requests. For other card types, an authorization must be used to check for validity.

[Required Variables]

`type=validate`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`processor_id`

Required if account has multiple processor ids. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`address1`

Cardholder’s billing address.

`zip`

[Optional Variables]

`amount`

If passed, the amount must be 0.00.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

`product_sku_1`

Required for recurring billing plans only.

`orderid`

`orderdescription`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

Void

Voids can only be done using the Direct Post method, not Transparent Redirect.

Voiding a transaction will cancel an existing sale or captured authorization from actually charging the card. In addition, non-captured authorizations can be voided to prevent any future capture. Note however, that the amount is still reserved on the card and will take a few days to expire. You will have to call the issuing bank to request that the authorization be removed if the customer does not want to wait for it to expire on its own.

Voids can only occur if the transaction has not been settled; settled transactions should be refunded.
[Required Variables]

`type=void`

`transactionid`

This value was passed in the response of the previous gateway transaction that needs to be void.

Refund

Refunds can only be done using the Direct Post method, not Transparent Redirect.

Transaction refunds will reverse a previously settled transaction. If the transaction has not been settled, it must be voided instead of refunded.
[Required Variables]

`type=refund`

`amount`

Total amount to be refunded. Format: x.xx

`transactionid`

This value was passed in the response of the previous Gateway Transaction that needs to be refunded.

Credit

Credits can only be done using the Direct Post method, not Transparent Redirect.

Transaction credits apply a negative amount to the cardholder's card. In most situations, credits are disabled as transaction refunds should be used instead.
[Required Variables]

`type=credit`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be refunded. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder's billing address.

`city`

`state`

Card billing state. Format: CC (2 Character abbrev.)

`zip`

`country`

Card billing country Format: CC (ISO-3166)(ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

See additional optional variables here.

Update

Updates can only be done using the Direct Post method, not Transparent Redirect.

Transaction updates can be used to update previous transactions with specific order information, such as a tracking number and shipping carrier.

[Required Variables]

`type=update`

`transactionid`

This value was passed in the response of the previous Gateway Transaction.

[Optional Variables]

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

Receiving a Response

The transaction responses are returned in the body of the HTTP response in a query string name/value format delimited by ampersands.

For example: `variable1=value1&variable2=value&variable3=value3`

Customizing Your Response Variables

You can choose to customize the variables you receive in the response to an API transaction. This functionality is particularly useful in pre-populating fields when a transaction fails, so a customer does not need to re-enter all of the information.

To do this, you must first log in to the control panel, then click “Options” in the left sidebar. Next, click on “API Configuration.” Here, you will be able to add variables to your response. You cannot remove any of the default variables from the response to a transaction. By default, the variables returned include:

- `response`
- `responsetext`
- `response_code`
- `authcode`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `orderid`
- `type`
- `customer_vault_id` (only returned by default when adding a customer to the vault)

Variables that cannot be added to the response include:

- `ccnumber`
- `ccexp`
- `cvv`

Default Response Variables

Variable Name	Description
response	“1” = Transaction Approved “2” = Transaction Declined “3” = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: “AUTH DECLINED” (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) “LOST/STOLEN CARD” (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and “CALL VOICE OPER” (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree’s transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer vault id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

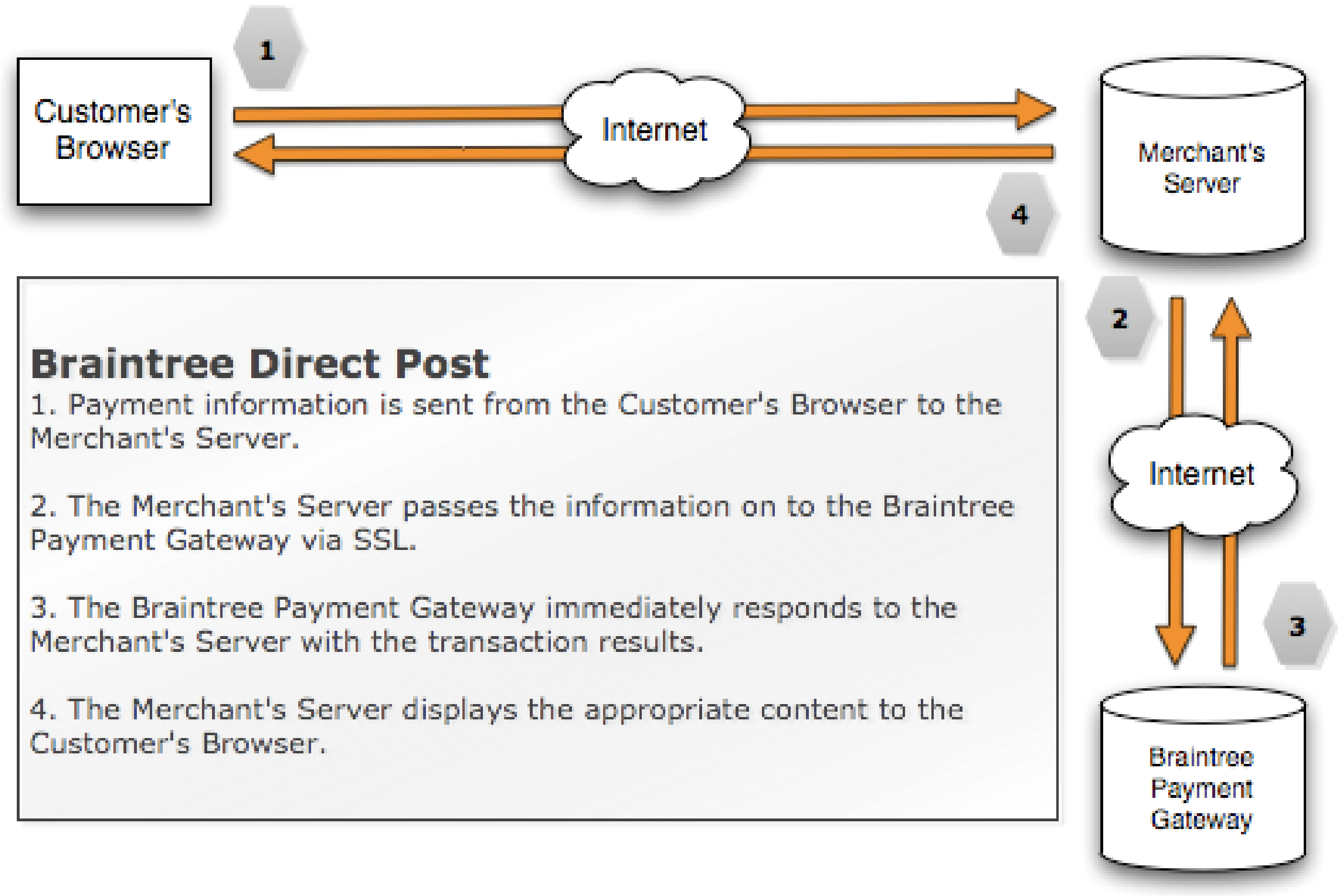
M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

Direct Post (DP) Vault API

The Vault works with either Direct Post or Transparent Redirect and enables the remote storage of credit card data. Credit card information can be submitted to the Vault with or without completing a transaction and receive a ‘token’ in response. Merchants can then use the token for all subsequent transactions including sale, auth, void and returns. Tokens can be 36 alphanumeric characters. Any sensitive information can be stored in the Vault including name, address, bank account information, social security and driver license numbers, image documents, media files, etc.

The Direct Post method sends messages to the Payment Gateway’s server using the standard HTTP protocol over an SSL connection. Please note that the full credit card data is passing through the merchant’s server. Braintree has developed the Transparent Redirect method to eliminate this security risk.

How Direct Post Works



Braintree Direct Post

1. Payment information is sent from the Customer's Browser to the Merchant's Server.

2. The Merchant's Server passes the information on to the Braintree Payment Gateway via SSL.

3. The Braintree Payment Gateway immediately responds to the Merchant's Server with the transaction results.

4. The Merchant's Server displays the appropriate content to the Customer's Browser.

In the Direct Post method, the communications with the cardholder (**Steps 1 and 4**) are developed completely by the merchant and therefore are not defined by the Payment Gateway. **Step 1** should simply collect the payment data from the cardholder and **Step 4** should display the appropriate transaction receipt or declined message.

In **Step 2**, transaction details should be delivered to the Payment Gateway using the POST method with the appropriate variables defined below posted along with the request.

In **Step 3**, the transaction responses are returned in the body of the HTTP response in a query string name/value format delimited by ampersands. For example:
`variable1=value1&variable2=value&variable3=value3`

Making A request

To make a request, determine transaction type (ie. sale, authorization, void, etc.) and pass the credit card number, expiration date and amount, using POST HTTP in the format
`variable1=value1&variable2=value2&variable3=value3`.

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

To authenticate against the Direct Post API, simply pass your username and password as two of the parameter variables in the request.

`username` (required)— Username assigned to the merchant account. This username is also a username to access the Braintree Virtual Terminal. Username should be hidden from customers in all situations.

`password` (required)— Password for the specified username. This password is also the password to access the Virtual Terminal. Password should be hidden from the customer in all situations.

Adding a customer to the Vault

Pass the required variables:

`customer_vault=add_customer`

`ccnumber` Valid credit card number. Pass if adding a credit card.

`ccexp` Credit card expiration date if adding a credit card. Format: MMYY (1010 = October, 2010)

`customer_vault_id` Specifies the name of the vault ID to be created. Values can be up to 32 alphanumeric characters. If you do not pass a customer_vault_id, the Vault will randomly generate one and return it in the response.

[Recommended Variables]

`firstname`

`lastname`

`address1`

`city`

`state` Card billing state Format: NY (2 Character abbrev.)

`zip`

`country` Card billing country. Format: CC (ISO-3166). Example country=US for the United States

`phone`

`email`

[Optional Variables]

`company`

`address2`

`fax`

`website`

In this example, we are adding the customer to the vault and specifying the vault ID of 123456:

```
username=demoapi&password=password1&ccnumber=4111111111111111&ccexp=1010
&customer_vault=add_customer&customer_vault_id=123456
```

Verifying AVS/CVV Upon Adding a Customer to the Vault

If the merchant is issuing a transaction while simultaneously creating a vault id, the customer will automatically be subjected to the AVS and CVV settings enabled on the account.

If no transaction is being issued when a customer is added to the vault, the merchant can issue a “type=auth” and “amount=1.00” request in the same string containing “customer_vault=add_customer”. If the customer’s information passes the AVS and CVV rejection settings, a vault id will be created. If the information does not, no vault id will be created.

It is important to note that Visa assesses a small processing fee for running an authorization on a card without a subsequent capture or void.

Issuing transactions using a Vault id

Pass the required variables:

`type=`

`amount=`

`customer_vault_id=`

In this example, we are issuing a sale transaction for Vault id 123456 for 10.00.

```
username=demoapi&password=password1&type=sale&amount=10.00&customer_vault_id=123456
```

Additional variables can be added:

Note: Only ASCII characters are accepted for the values.

Sale

Transaction sales are submitted and immediately flagged for settlement.
[Required Variables]

`type=sale`

`customer_vault_id`

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`zip`

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`billing_method`

Required only when issuing a recurring billing transaction. Value should be set to “recurring”

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing. Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Authorization

Authorizations are authorized immediately, but are not flagged for settlement. After a successful authorization, funds will be held in the customer’s account for 3-7 business days. In order to submit these transactions for settlement, the merchant must submit a “type=capture” request. The merchant also has the option to void these authorizations before a capture is submitted. If the merchant does not run a subsequent capture or void, the authorization will fall of the customer’s account after 3-7 business days. Visa assesses a small processing fee for allowing this to happen.

When part of an authorization is captured, the rest of the original authorization is automatically voided. [Required Variables]

`type=auth`

`customer_vault_id`

`amount`

Total amount to be authorized. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state Format: CC (2 Character abbrev.)

`zip`

`country` Format:

Card billing country CC (ISO-3166) (ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`tax`

Required for Level 2 processing. Total tax amount, format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in format: ups / fedex / dhl / usps

See additional optional variables here.

Capture

Transaction captures flag existing authorizations for settlement. Only authorizations can be captured. Captures can be submitted for an amount equal to or less than the original authorization. When an authorization has been partially captured, the rest of the original authorization is automatically voided.

[Required Variables]

`type=capture`

`amount`

Total amount to be charged. Format: x.xx

`transactionid`

This value was passed in the response to the previous gateway authorization transaction.

[Optional Variables]

`descriptor`

Set payment descriptor (On supported processors)

`descriptor_phone`

Set payment descriptor phone (On supported processors)

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Validate

A “type=validate” will check for card validity without ever holding funds on the cardholder’s account. A validate request cannot not determine whether the customer has enough funds to complete a transaction or not, nor will it return AVS or CVV data. Currently, Visa and MasterCard are the only two credit cards that accept validate requests. For other card types, an authorization must be used to check for validity.

[Required Variables]

`type=validate`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`processor_id`

Required if account has multiple processor ids. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`address1`

Cardholder’s billing address.

`zip`

[Optional Variables]

`amount`

If passed, the amount must be 0.00.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

`product_sku_1`

Required for recurring billing plans only.

`orderid`

`orderdescription`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Void

Transaction voids will cancel an existing sale or captured authorization. In addition, non-captured authorizations can be voided to prevent any future capture. Voids can only occur if the transaction has not been settled.

[Required Variables]

`type=void`

`transactionid`

This value was passed in the response of the previous gateway transaction that needs to be void.

Refund

Transaction refunds will reverse a previously settled transaction. If the transaction has not been settled, it must be voided instead of refunded.
[Required Variables]

`type=refund`

`amount`

Total amount to be refunded. Format: x.xx

`transactionid`

This value was passed in the response of the previous Gateway Transaction that needs to be refunded.

Credit

Transaction credits apply a negative amount to the cardholder's card. In most situations, credits are disabled as transaction refunds should be used instead.
[Required Variables]

`type=credit`

`customer_vault_id`

`amount`

Total amount to be refunded. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder's billing address.

`city`

`state`

Card billing state. Format: CC (2 Character abbrev.)

`zip`

`country`

Card billing country Format: CC (ISO-3166)(ie. US)

`phone`

`email`

[Optional Variables]

`product_sku_1`

Required for recurring billing plans only.

`orderdescription`

`orderid`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

See additional optional variables here.

Update

Transaction updates can be used to update previous transactions with specific order information, such as a tracking number and shipping carrier.
[Required Variables]

`type=update`

`transactionid`

This value was passed in the response of the previous Gateway Transaction.

[Optional Variables]

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

If successful, the response code of 1 will be returned along with customer_vault_id. If a response of 2 or 3 is returned, the transaction failed and the customer has not been added to the vault.

Updating an existing Vault record

"Updating" refers to both adding information on an existing Vault record and updating information already stored on a Vault record. It is important to note that the gateway will erase data for any variable that is passed with a null value on an update customer request. For this reason, we strongly recommend you only pass the variables you wish to update when issuing this type of request.

Pass the required variables:

`customer_vault=update_customer`

`customer_vault_id` Specifies the specific customer record to update

Plus, the variable you would like to add/update.

[Recommended Variables]

`firstname`

`lastname`

`address1`

`city`

`state` Card billing state Format: NY (2 Character abbrev.)

`zip`

`country` Card billing country. Format: CC (ISO-3166). Example country=US for the United States

`phone`

`email`

[Optional Variables]

`company`

`address2`

`fax`

`website`

In this example, we are updating the expiration date of vault ID 123456

```
username=demoapi&password=password1&ccexp=1012
&customer_vault=update_customer&customer_vault_id=123456
```

Deleting a Vault record

Pass the required variables:

`customer_vault=delete_customer`

`customer_vault_id` Specifies the specific customer record to delete.

In this example, we are deleting customer vault ID 123456:

```
username=demoapi&password=password1&customer_vault=delete_customer&customer_vault_i
d=123456
```

Adding a customer and issuing a transaction

Pass the required fields:

`customer_vault=add_customer`

`customer_vault_id` Specifies the name of the vault ID to be created. If you do not pass a customer_vault_id, the Vault will randomly generate one and return it in the response.

`amount` Format x.xx

`type` Format `Sale` / `Auth`

In this example, we are adding vault ID 123456 and performing a sale of $10.00:

```
username=demoapi&password=password1&ccnumber=4111111111111111&ccexp=1010
&type=sale&amount=10.00&customer_vault=add_customer&customer_vault_id=123456
```

Receiving a Response

The transaction responses are returned in the body of the HTTP response in a query string name/value format delimited by ampersands.

For example: `variable1=value1&variable2=value&variable3=value3`

Customizing Your Response Variables

You can choose to customize the variables you receive in the response to an API transaction. This functionality is particularly useful in pre-populating fields when a transaction fails, so a customer does not need to re-enter all of the information.

To do this, you must first log in to the control panel, then click “Options” in the left sidebar. Next, click on “API Configuration.” Here, you will be able to add variables to your response. You cannot remove any of the default variables from the response to a transaction. By default, the variables returned include:

- `response`
- `responsetext`
- `response_code`
- `authcode`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `orderid`
- `type`
- `customer_vault_id` (only returned by default when adding a customer to the vault)

Variables that cannot be added to the response include:

- `ccnumber`
- `ccexp`
- `cvv`

Default Response Variables

Variable Name	Description
response	“1” = Transaction Approved “2” = Transaction Declined “3” = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: “AUTH DECLINED” (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) “LOST/STOLEN CARD” (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and “CALL VOICE OPER” (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree’s transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer_vault_id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

Batch Processing

Batch processing is an additional method for processing transactions. It is most commonly used to process a large batch of transactions at a single time for merchants doing recurring billing or mail and catalog orders. If real time processing is needed for applications such as ecommerce, the Direct Post or Transparent Redirect APIs should be used instead.

Our batch process is extremely fast, capable of processing thousands of orders every few minutes. Merchants can pass a batch file via the control panel or FTP over SSL in csv, xls, xlsx, or txt format.

Merchants may utilize our Vault API to eliminate the handling, processing and storage of credit card data. You can add customers to the vault via batch upload or use existing vault ids in place of credit card numbers to issue transactions.

Files can be uploaded in one of two ways: the Control Panel, or FTPS (FTP over SSL) and can only contain ASCII characters.

Batch Processing via Control Panel

1. Configure user permissions

The user must first be given the user right to do batch processing in the control panel.

2. Set format

Merchants use our auto detect utility or pre-configure preferred file formats in the control panel. Each column represents a variable and each row is a transaction/record with the corresponding variable value. See example below.

Format: **Column A:** Vault ID, **Column B:** type, **Column C:** Amount, **Column D,E,etc.**

Row 1 (first transaction): **Column A:** 123456, **Column B:** sale, **Column C:** 10.00, **Column D,E,etc.**

See below for all available fields.

3. Create file and upload

Once the .csv, .xls, .xlsx, or .txt file is created and populated with the request data, upload it in the control panel.

Receiving a Response

All transaction details can be accessed in the control panel by clicking on the report icon next to the batch summary or via API using the Query API. Transactions type=sale will automatically be settled at the specified time (Options/Settlement Schedule) while type=auth needs to be captured before settlement occurs.

See Response Codes below

Batch Processing via Uploading FTP over SSL

1. Ask us to turn the functionality on for your account and ensure that the user has been given the permission to do batch processing in the control panel.

Email us at Support@getbraintree.com.

2. Set up your batch file format in the control panel.

You can tell our system what order you will be passing the fields by creating a format and setting it as the default. You can either include the header row, or delete it. If you include the header row, you will get one error and the reason text will read “Header Line”. The available fields are listed below.

You can also tell our system to auto detect the format by including the exact field names in the first row and the values in the corresponding columns. You will automatically get one error for the first row. The reason text will read “Header Line”. The available fields are listed below.

3. Create and upload the file.

The host uses Explicit SSL or SSL-TLS. Using curl for example, a connection string would look like this:

```
curl -k --ftp-ssl -3 -u username:password ftp://123.45.678.90:0000
```

Curl is a command line tool for transferring files which is freely available for most platforms. Many unix/linux distributions have it installed by default. The option **-k** auto-validates the certificate, **-ftp-ssl** tells it to use FTPS for the following FTP address, **-3** says to use SSLv3, **-u** gives the user/password combination.

4. Check for the response

When responses are delivered to the /response directory, they are prepended with response. If “transactions.csv” was uploaded to /upload, the response would be saved to /response as “response_transactions.csv”

The fields in the original file will be returned as well as the response codes (see below) for each line of the file.

Additionally, transaction details may also be obtained using the Query API or via reporting in the control panel.

Fields

Request Variables

All values should be lowercase.

[Required Fields for Adding a Customer to the Vault]

---

customer_vault_action=add_customer

cc_number

cc_exp (format: mmyy)

[Required Fields for a Credit Card Transaction]

---

type (sale, auth, capture, credit, void, refund)

cc_number

cc_exp (format: mmyy)

amount (format 0.00)

[Available Fields]

---

authorization_code

address_1

address_2

city

company

country

email

fax

first_name

last_name

billing_method

phone

postal_code

state

cvv

cardholder_auth

cavv

cell_phone

cc_exp

cc_number

currency (for example USD, EUR, JPY)

customer_vault (add_customer, update_customer, delete_customer)

customer_vault_id

descriptor

descriptor_phone

dup_seconds (Duplicate transaction threshold: “20” would set duplicate checking for 20 seconds)

ignore

industry

ipaddress

merchant_defined_field_1

merchant_defined_field_2

merchant_defined_field_3

merchant_defined_field_4

merchant_defined_field_5

merchant_defined_field_6

merchant_defined_field_7

merchant_defined_field_8

merchant_defined_field_9

merchant_defined_field_10

merchant_defined_field_11

merchant_defined_field_12

merchant_defined_field_13

merchant_defined_field_14

merchant_defined_field_15

merchant_defined_field_16

merchant_defined_field_17

merchant_defined_field_18

merchant_defined_field_19

merchant_defined_field_20

order_description

order_id

po_number

payment

processor_id (the id name for the merchant account)

product_sku_1

tracking_number

shipping_amount

shipping_address_1

shipping_address_2

shipping_carrier (ups, usps, fedex, dhl)

shipping_city

shipping_company

shipping_country

shipping_date

shipping_email

shipping_first_name

shipping_last_name

shipping_postal_code

shipping_state

tax_amount

amount

transaction_id

type

website

xid

Adding a Customer via Batch Processing

Pass the following variables:

If adding credit card information:

- `customer_vault_action=add_customer`
- `customer_vault_id`- pass if you want to specify the ID. If not passed, the system will generate a vault ID for you.
- `cc_number`
- `cc_exp`

Additional optional fields:

Any variables can be stored in the vault (see list of all variables).

Some common fields passed are:

- `first_name`
- `last_name`
- `address_1`
- `city`
- `state`
- `zip`
- `email`

Default Response Variables

There are 62 unique variables that will always be included in the same order in the response file. A field will be blank if no data was passed for that field in the request.
[Response Variables]

---

transactionid

response – 1, 2, or 3

responsetext

avs_results

csc_results

auth_code

amount

BLANK – This field is intentionally left blank as a separator between the response and the data.

[Request Variables]

---

order_id

order_description

po_number

shipping_amount

tax_amount

firstname

lastname

company

email

phone

fax

website

address_1

address_2

city

state

postal_code

country

shipping_first_name

shipping_last_name

shipping_company

shipping_email

shipping_address1

shipping_address2

shipping_city

shipping_state

shipping_postal_code

shipping_country – Format: CC (ISO-3166)

type – Transaction Type

responsecode

processor_id

payment – Payment type

merchant_defined_field_1

merchant_defined_field_2

merchant_defined_field_3

merchant_defined_field_4

merchant_defined_field_5

merchant_defined_field_6

merchant_defined_field_7

merchant_defined_field_8

merchant_defined_field_9

merchant_defined_field_10

merchant_defined_field_11

merchant_defined_field_12

merchant_defined_field_13

merchant_defined_field_14

merchant_defined_field_15

merchant_defined_field_16

merchant_defined_field_17

merchant_defined_field_18

merchant_defined_field_19

merchant_defined_field_20

customer_vault

customer_vault_id

Response Explanation

Variable Name	Description
response	"1" = Transaction Approved "2" = Transaction Declined "3" = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: "AUTH DECLINED" (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) "LOST/STOLEN CARD" (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and "CALL VOICE OPER" (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree's transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer vault id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions available (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

Braintree Managed Recurring Billing

Managing recurring billing is extremely simple with Braintree. For the most part, it can be completed via the API. The one exception is the initial set up of your specific plans which is done via the Virtual Terminal. There is not a lot of logic written into the Recurring Billing API because we feel it provides greater flexibility, which you'll see when it comes time to add a customer to a plan. For more complicated plans/models, we recommend storing customers in our Vault and writing the logic your side to initiate the transactions.

Setting Up A Plan

The first step is to add a plan under Recurring -> Add Plan.

You have the ability to set up the following parameters that accommodate your billing needs:

- Amount
- Frequency – e.g. every 30 days, 1st day of the month, etc.
- Duration – e.g. only four times, perpetually, etc.
- Plan descriptor – SKU. We recommend you choose a simple name for the HTTP variable. If you choose a longer SKU name remember to escape the value, like 'A%20Monthly%20Plan'.

For this example, we will use the following parameters:

- Amount – $10
- Frequency – 1st of each month
- Duration – until canceled
- Plan Descriptor – monthly

Note- Recurring billing runs at approx 3 Central time daily. If a recurring plan falls on the 29th, 30th, or 31st of each month and the month does not have those dates, it will automatically be processed on the last day of the month.

After creating this plan, you can view it in the Virtual Terminal under Recurring -> List Plans. That's basically all you need to do to setup recurring billing plans. Now, on to application integration.

Managing Customers

We've already setup our plan in the Virtual Terminal and now we can start to administer customers to it via the API. The key parameter for recurring billing is `product_sku_1`. A customer can only be added to a recurring billing plan along with a transaction (a "type=sale", "type=validate", "type=auth" or a "type=add_recurring").

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

Using the Direct Post method, we will start by setting up our regular credentials:

```
username=testapi&password=password1
```

Adding Customers

We will now add a customer to the 'monthly' plan. This means that we're going to add `product_sku_1=monthly` on to our query parameter string.

Using Direct Post:

```
username=testapi&password=password1&product_sku_1=monthly
```

Adding a customer to this SKU means that they will be charged according to the plan we've called "monthly", and set up to charge the customer $10.00 on the first day of each month. Recurring plans will not start charging the customer until the interval you selected, so if you want them to be charged immediately, you will have to run a transaction as explained below.

Note: Make sure that you save the transaction ID from the gateway response, as you'll need it to delete a customer from the plan and track what customer is a part of what plan. For the time being, you have to manually track who's a member of what plan in your system. This isn't difficult if the transaction ID is kept from the previous step.

You should run a transaction when you add the customer to the billing plan, so the card is validated. You can charge the customer the full or pro rated amount of the plan. When the credit card number is entered into our system, the only validation check we do is Luhn-10, which validates the number of digits is correct. We can't do any further validation until we hit the issuing bank with a transaction.

If you are not giving any sort of 'grace' period, meaning that you're going to charge them the full or pro-rated amount now, instead of waiting until the next billing cycle, you will run a sale for the desired charge. For a prorated amount, we recommend that the sale amount be calculated as the monthly value divided by how many days are left in the current billing cycle. In our example, we bill $10.00 on the 1st. If a customer signs up on the 20th we would charge $3.33- ($10.00 / 30 days in a month) X 10 days = $3.33. This will leave us with this:

Direct Post

```
username=testapi&password=password1&product_sku_1=monthly& \
ccnumber=4111111111111111&ccexp=1010&type=sale&amount=3.33
```

Transparent Redirect

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php>
```

```
  <input id="product_sku_1" type="hidden" value="monthly" name="product_sku_1"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="776320" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="3.33" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="069831532de261651e0396bcea5c31b9"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value "1010" name="ccexp"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Adding Customers without Issuing a Transaction

We can add a customer to a plan without issuing a transaction. The variable `type=add_recurring` is required for this type of request.

Using this approach, you will also be able to specify the first charge date of the recurring billing plan using the parameter `start_date=YYYYMMDD`. This variable will specify the first actual charge of the recurring billing plan. Following this date, the plan will run according to the specified plan settings. This feature only works via Direct Post.

Direct Post

```
username=testapi&password=password1&product_sku_1=monthly&
ccnumber=4111111111111111&ccexp=1010&type=add_recurring&
start_date=20090630
```

Deleting Customers

Deleting a customer from a recurring billing plan is simple. We just need to add a parameter called `delete_recurring` with the value to be the transaction ID of the transaction that added them to the plan. If you need to look up this id, log into the control panel, select "List Subscriptions" and download the customers into .xls or .csv. The transaction id is listed in column AA, labeled "Reference Id".

Deleting a customer from a recurring billing plan is done using Direct Post and looks like this:

```
username=testapi&password=password1&delete_recurring=1234567
```

There is no way to simply stop a customer's recurring billing. If you want to stop charging the customer, you can delete them from the plan as above or:

1. Move them to a new plan that will only charge them once. After that charge, they will still be stored in Braintree's recurring billing system and their recurring plan will be completed. The process for is described below.

2. Use our Vault to store the customers billing information, so that when you delete them from the recurring plan, you can always add them again without asking the customer for their billing information again or storing the data on your system. Please contact us if you'd like to use the Vault and do not already have it enabled.

Switching Customers to A Different Plan

When someone upgrades to a different plan you’ll need to simply delete them from a current plan and add them to a different plan in two separate calls. In these examples, I remove a customer from the ‘monthly’ and plan add them to the ‘personal’ plan which costs $16.00 a month and charge them a pro-rated amount of $6.00 for the difference:

Direct Post:

```
username=testapi&password=password1&delete_recurring=1234567

username=testapi&password=password1&product_sku_1=personal&ccnumber=4111111111111111
&ccexp=1010&
type=sale&amount=6.00
```

Transparent Redirect

Use the Direct post to delete the customer from the first plan

```
username=testapi&password=password1&delete_recurring=1234567
```

and add them to the new plan using TR

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php>
  <input id="product_sku_1" type="hidden" value="personal" name="product_sku_1"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="776320" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="6.00" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="e0b02888e46071274aea62c011b1f73d"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value "1010" name="ccexp"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

If they are downgrading their account, you can choose to do a refund transaction (pass `type=refund` and `transactionid` for the transaction that added the customer to the first plan).

Using Vault Customers in Recurring Billing Plans

If you are adding a customer that is stored in the vault to a recurring billing plan, simply pass `customer_vault_id` in place of the credit card information variables `ccnumber` and `&ccexp`.

Direct Post

```
username=testapi&password=password1&product_sku_1=monthly& \
customer_vault_id=123456&type=sale&amount=3.33
```

Transparent Redirect

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php>
  <input id="product_sku_1" type="hidden" value="monthly" name="product_sku_1"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="776320" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="3.33" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="069831532de261651e0396bcea5c31b9"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="customer_vault_id" type="hidden" value="123456"
name="customer_vault_id"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

To add a customer to the vault and to a recurring billing plan in one call, include the variable `customer_vault=add_customer` along with the other required fields above.

Direct Post

```
username=testapi&password=password1&product_sku_1=monthly&ccnumber=4111111111111111
&ccexp=1010&customer_vault=add_customer&type=sale&amount=3.33
```

Transparent Redirect

```
<form method="post"
action="https://secure.braintreepaymentgateway.com/api/transact.php>
  <input id="product_sku_1" type="hidden" value="monthly" name="product_sku_1"/>
  <input id="type" type="hidden" value="sale" name="type"/>
  <input id="key_id" type="hidden" value="776320" name="key_id"/>
  <input id="orderid" type="hidden" value="123456-xfj" name="orderid"/>
  <input id="amount" type="hidden" value="3.33" name="amount"/>
  <input id="time" type="hidden" value="20080516190549" name="time"/>
  <input id="hash" type="hidden" value="069831532de261651e0396bcea5c31b9"
name="hash"/>
  <input id="redirect" type="hidden" value="http://example.tld/gateway_response"
name="redirect"/>
  <input id="ccnumber" type="text" value="4111111111111111" name="ccnumber"/>
  <input id="ccexp" type="text" value "1010" name="ccexp"/>
  <input id="customer_vault" type="hidden" value "add_customer"
name="customer_vault"/>
  <p><input type="submit" value="Submit" name="commit"/></p>
</form>
```

Billing

The customer will be billed according to the plan specified. You can check the next scheduled charge data and the billing history in the control panel.

If a transaction fails, you will receive the failure message in the transaction response. The system will not automatically re-attempt the transaction.

Receiving a Response

It is important to note that the response returned only references the result of the “type” request in your submission. You may pass a “type=sale” + “product_sku_1=(incorrect_plan_sku)” and see a successful response. This would only indicate that the “type=sale” was successful, and that a successful charge was assessed on the customer’s card. To see subscriptions, log into the gateway and go to Recurring -> List Subscriptions.

If you have added the customer using the variable “type=add-recrring”, you will receive a response of 1 with a response text “Recurring Added”.

Customizing Your Response Variables

You can choose to customize the variables you receive in the response to an API transaction. This functionality is particularly useful in pre-populating fields when a transaction fails, so a customer does not need to re-enter all of the information.

To do this, you must first log in to the control panel, then click “Options” in the left sidebar. Next, click on “API Configuration.” Here, you will be able to add variables to your response. You cannot remove any of the default variables from the response to a transaction. By default, the variables returned include:

- `response`
- `responsetext`
- `response_code`
- `authcode`
- `transactionid`
- `avsresponse`
- `cvvresponse`
- `orderid`
- `type`
- `customer_vault_id` (only returned by default when adding a customer to the vault)

Variables that cannot be added to the response include:

- `ccnumber`
- `ccexp`
- `cvv`

Default Response Variables

Variable Name	Description
response	“1” = Transaction Approved “2” = Transaction Declined “3” = Error in transaction data or system error
response_code	Numeric mapping of process responses (See Transaction Response Codes below)
responsetext	Textual response generated by the issuer. Common failed responses include: “AUTH DECLINED” (The issuer is rejecting the transaction without supplying a reason. The merchant needs to get another form of payment from their customer.) “LOST/STOLEN CARD” (The issuer has marked the card as fraudulent and is thus rejecting the transaction. The merchant needs to get another form of payment from their customer.) and “CALL VOICE OPER” (Issuer is refusing to accept the transaction, but requesting that the merchant conduct a voice authorization instead).
authcode	6 digit transaction authorization code
transactionid	Braintree’s transaction id
avsresponse	AVS Response Code (See below)
cvvresponse	CVV Response Code (See below)
orderid	The original order id passed in the transaction request
customer_vault_id	The customer vault id – only returned by default when a customer vault id is created

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

Merchant Managed Recurring Billing

Merchant managed recurring billing provides merchants complete control over how, when and how much a customer is billed and does not require the merchant to hold credit card data in their environment. The merchant creates the logic and passes us the transaction information at the time of the billing.

Making a Request

There are two methods that merchants can choose from to perform the transactions. In each case, the merchant does all the calculations in their own system and sends the transaction to Braintree when the customer is to be billed.

API

To regularly bill a customer via the API, you will want to add them to the vault in the initial request.

The credit card data is remotely stored (can be done with or without doing a transaction) in the Vault and merchants then use the unique vault ID that's returned for all subsequent transactions. Used in conjunction with the Transparent Redirect Payment API merchants can eliminate the handling, processing and storage of credit card information which greatly reduces PCI scope and increases credit card data security.

To make a request, pass the transaction type (i.e. `type=sale`) and pass the credit card number, expiration date and amount, using POST HTTP in the format `variable1=value1&variable2=value2&variable3=value3`.

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

For example, we are doing a $10.00 sale and adding a customer to the vault for subsequent billing. 123456 is the vault ID we are assigning:

```
username=testapi&password=password1&ccnumber=4111111111111111&ccexp=1010
&type=sale&amount=10.00&customer_vault=add_customer&customer_vault_id=123456
```

To initiate additional transactions using this vault id, pass:

```
username=testapi&password=password1&type=sale&amount=10.00&
customer_vault_id=123456
```

Additional variables can be added depending on the transaction type on the following pages.

Note: Only ASCII characters are accepted for the values.

Batch Upload

Batch processing is done via FTP over SSL and merchants can choose to pass the full credit card number or the Vault ID. The specified transaction type (`type=sale`, `type=auth`) is then performed and a response file returned.

Sale

Sales can be done via Transparent Redirect or Direct Post.

Transaction sales are submitted and immediately flagged for settlement.

[Required Variables]

`type=sale`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be charged. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`zip`

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

[Optional Variables]

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

See additional optional variables here.

Authorization

Authorizations can be done via Transparent Redirect or Direct Post Methods.

Authorizations are authorized immediately, but are not flagged for settlement. After a successful authorization, funds will be held in the customer's account for 3-7 business days. In order to submit these transactions for settlement, the merchant must submit a "type=capture" request. The merchant also has the option to void these authorizations before a capture is submitted. If the merchant does not run a subsequent capture or void, the authorization will fall of the customer's account after 3-7 business days. Visa assesses a small processing fee for allowing this to happen.

When part of an authorization is captured, the rest of the original authorization is automatically voided.

[Required Variables]

`type=auth`

`ccnumber`

Valid credit card number.

`ccexp` Format: MMYY

Credit card expiration date for the credit card passed. (1010 = October, 2010)

`amount`

Total amount to be authorized. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder's billing address.

`city`

`state`

Card billing state Format: CC (2 Character abbrev.)

`zip`

`country` Format:

Card billing country CC (ISO-3166) (ie. US)

`phone`

`email`

[Optional Variables]

`orderdescription`

`orderid`

`tax`

Required for Level 2 Processing Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 processing. Original Purchase Order number.

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in format: ups / fedex / dhl / usps

See additional optional variables here.

Capture

Captures can be done via Transparent Redirect or Direct Post Methods.

Transaction captures flag existing authorizations for settlement. Only authorizations can be captured. Captures can be submitted for an amount equal to or less than the original authorization. When an authorization has been partially captured, the rest of the original authorization is automatically voided.
[Required Variables]

`type=capture`

`amount`

Total amount to be charged. Format: x.xx

`transactionid`

This value was passed in the response to the previous gateway authorization transaction.

[Optional Variables]

`tracking_number`

Shipping Tracking number

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

`orderid`

Validate

Validate can be done via Transparent Redirect or Direct Post methods.

A “type=validate” will check for card validity without ever holding funds on the cardholder’s account. A validate request cannot not determine whether the customer has enough funds to complete a transaction or not, nor will it return AVS or CVV data. Currently, Visa and MasterCard are the only two credit cards that accept validate requests. For other card types, an authorization must be used to check for validity.

[Required Variables]

`type=validate`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`processor_id`

Required if account has multiple processor ids. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`cvv`

Card security code. This value cannot be stored.

`address1`

Cardholder’s billing address.

`zip`

[Optional Variables]

`amount`

If passed, the amount must be 0.00.

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`city`

`state`

Card billing state (2 Character abbrev.) Format: CC

`country`

Card billing country (ie. US). Format: CC (ISO-3166)

`phone`

`email`

`orderid`

`orderdescription`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format. Format: ups / fedex / dhl / usps

Void

Voids can only be done using the Direct Post method, not Transparent Redirect.

Voiding a transaction will cancel an existing sale or captured authorization from actually charging the card. In addition, non-captured authorizations can be voided to prevent any future capture. Note however, that the amount is still reserved on the card and will take a few days to expire. You will have to call the issuing bank to request that the authorization be removed if the customer does not want to wait for it to expire on its own.

Voids can only occur if the transaction has not been settled; settled transactions should be refunded.
[Required Variables]

`type=void`

`transactionid`

This value was passed in the response of the previous gateway transaction that needs to be void.

Refund

Refunds can only be done using the Direct Post method, not Transparent Redirect.

Transaction refunds will reverse a previously settled transaction. If the transaction has not been settled, it must be voided instead of refunded.
[Required Variables]

`type=refund`

`amount`

Total amount to be refunded. Format: x.xx

`transactionid`

This value was passed in the response of the previous Gateway Transaction that needs to be refunded.

Credit

Credits can only be done using the Direct Post method, not Transparent Redirect.

Transaction credits apply a negative amount to the cardholder’s card. In most situations, credits are disabled as transaction refunds should be used instead.

[Required Variables]

`type=credit`

`ccnumber`

Valid credit card number.

`ccexp`

Credit card expiration date for the credit card passed. Format: MMYY (1010 = October, 2010)

`amount`

Total amount to be refunded. Format: x.xx

`processor_id`

Required if account has multiple processor ids. The `processor_id` variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

[Recommended Variables]

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`address1`

Cardholder’s billing address.

`city`

`state`

Card billing state. Format: CC (2 Character abbrev.)

`zip`

`country`

Card billing country Format: CC (ISO-3166)(ie. US)

`phone`

`email`

[Optional Variables]

`orderdescription`

`orderid`

`company`

Cardholder’s company

`address2`

Card billing address – line 2

`fax`

Billing fax number

`website`

`shipping_firstname`

`shipping_lastname`

`shipping_company`

`shipping_address1`

`shipping_address2`

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

`shipping_country`

Format: CC (ISO-3166)

`tracking_number`

`shipping_carrier`

Shipping carrier for order, value may only the ones listed in the format: ups / fedex / dhl / usps

See additional optional variables here.

Reporting and Reconciliation

Overview

Our dynamic reporting and reconciliation capabilities allow for the quick and easy retrieval of customer and transaction information.

In addition to reporting, merchants can use the Query API to create a self service customer management portal within their own application where cardholders can add, update and delete payment types in addition to viewing past transaction history details – all without the merchant ever processing, transmitting or storing any credit card data.

There are two ways merchants can access information:

Control Panel

The reporting tools in the Control Panel allows merchants to generate high level reports and drill down into the finest detail. Custom fields can be created in the gateway to capture business specific information which is all stored and searchable. Merchants can seamlessly manage multiple merchant accounts and sales channels and structure for easy access and reporting. Data from the Control Panel can be downloaded in XLS and CSV format.

Query API

The Query API provides customer and transaction data in XML format using a simple POST interface using our Direct Post method. It does not work using the Transparent Redirect method. Merchants can automate reporting and reconciliation processes in their own applications or by building something entirely custom for the organization.

Security

Using the Query API requires that you provide your username and password, therefore requiring that this information be stored on your server. As a security precaution, you could create a second username and password and give that username access to only view information and no access to make transactions.

You set this up in the control panel by clicking “Options” from the sidebar. Select “User Accounts” and “Add a New User”. Give the new user permission to “Access Transaction Reports”, “Access Other Users’ Transactions”, and “Access the Customer Vault”.

Request

The Query URL: `https://secure.braintreepaymentgateway.com/api/query.php`

Use POST to pass your request to the Query API and you must pass your username and password.

The following fields are searchable via the API:

Variable Name	Description
`email`	Cardholder's email address
`order_id`	Order ID passed with transaction
`last_name`	Last name of cardholder
`cc_number`	Credit card number of card holder, can pass last 4 or all 16.
`start_date`	Only transactions that have been modified on or after this date will be retrieved.
`end_date`	Only transactions that have been modified on or before this date will be retrieved.
`condition`	Retrieves only transactions with the specified transaction condition. Values: `pending` ('Auth Only' transactions that are awaiting capture) `pendingsettlement` (transaction is awaiting settlement) `failed` (transaction has failed) `canceled` (transaction has been voided) `complete` (transaction has settled) `unknown` (An unknown error was encountered while processing this transaction). For example, to retrieve all transactions pending settlement or complete, the following could be used: `condition=pendingsettlement,complete`
`action_type`	Retrieves only transactions with the specified action types- `sale`, `refund`, `credit`, `auth`, `capture`, `void`, `settle`- A combination of the values above can be used, and should be separated by commas.
`transaction_id`	Original payment gateway transaction ID, which was passed in the response of the transaction.
`customer_vault_id`	Retrieves all transactions for the specified vault id
`report_type`	Value '`customer_vault`' pulls back vault record information. Passing the variable by itself will pull back all vault records, pass both `report_type=customer_vault` and `customer_vault_id` to pull back the vault record information for a specific vault id. The credit card number will be masked.
`merchant_defined_field_x` (where x is the field number)	Retrieves only transactions with a particular value for the custom field.

Retrieving Vault record information

Simply decide on what information you need to have returned and setup the variable parameters to pass through the query string.

Pull Vault Record Information for a Customer Vault ID

It is also possible to query to retrieve Vault record information for a given ID. Simply pass `report_type=customer_vault` and the vault ID.

Pull All Transactions for a Customer Vault ID

Simply pass the variable `customer_vault_id`. This will pull back a list of all transactions done using that vault record.

Showing Customers Their Recent Transactions

First query our records server side using the variable `customer_vault_id` and then render the returned transaction information to the customer.

Query to See if a Customer is Already in the Vault

Pass the variable `report_type=customer_vault` along with the information you know about the customer, such as the full credit card number. This will pull back a list of the vault records associated with that information.

Response

Responses will be received in an XML document format. Transactions can be queried immediately using the “transaction_id” variable. Querying via any other variable may fail for up to one minute after the transaction has been submitted. The root node will be `<nm_response>` with each node underneath being a node named for the variable parameter.

After you’ve submitted your request using POST, our system will respond with a set of transactions in XML format. The examples below shows what a response might look like.

Transaction Example

```
<nm_response>
  <transaction>
    <transaction_id>560117067</transaction_id>
    <platform_id />
    <transaction_type>cc</transaction_type>
    <condition>pendingsettlement</condition>
    <order_id>1234</order_id>
    <authorization_code>123456</authorization_code>
    <ponumber />
    <order_description>Test Order</order_description>
    <first_name>John</first_name>
    <last_name>Smith</last_name>
    <address_1>1234 Main St.</address_1>
    <address_2>Suite 2</address_2>
    <company>Acme, Inc.</company>
    <city>Chicago</city>
    <state>IL</state>
    <postal_code>60120</postal_code>
    <country>US</country>
    <email>test@test.com</email>
    <phone />
    <fax />
    <cell_phone />
    <customertaxid />
    <customerid />
    <website />
    <shipping_first_name>Mary</shipping_first_name>
    <shipping_last_name>Smith</shipping_last_name>
    <shipping_address_1>5551W Arbol Ln.</shipping_address_1>
    <shipping_address_2 />
    <shipping_company />
    <shipping_city>Madison</shipping_city>
    <shipping_state>WI</shipping_state>
    <shipping_postal_code>60133</shipping_postal_code>
    <shipping_country>US</shipping_country>
    <shipping_email>mary@test.com</shipping_email>
    <shipping_carrier />
    <tracking_number />
    <shipping_date />
    <shipping />
    <cc_number>4xxxxxxxxxxx1111</cc_number>
    <cc_hash>f6c609e195d9d4c185dcc8ca662f0180</cc_hash>
    <cc_exp>1010</cc_exp>
    <cavv />
    <cavv_result />
    <xid />
    <avs_response>N</avs_response>
    <csc_response>N</csc_response>
    <cardholder_auth />
    <check_account />
    <check_hash />
    <check_aba />
    <check_name />
    <account_holder_type />
    <account_type />
    <sec_code />
    <processor_id>default</processor_id>
    <tax />
    <merchant_defined_field id="1">Flight 201, Seat 2,
4</merchant_defined_field>
    <merchant_defined_field id="3">12345539</merchant_defined_field>
    <cc_bin>411111</cc_bin>
    <product>
        <sku>monthly</sku>
        <quantity></quantity>
        <description></description>
        <amount></amount>
     </product>
    □ <action>
      <amount>125.00</amount>
      <action_type>sale</action_type>
      <date>20080201165812</date>
      <success>1</success>
      <ip_address>76.193.193.213</ip_address>
      <source>virtual_terminal</source>
      <username>demoapi</username>
      <response_text>SUCCESS</response_text>
    </action>
  </transaction>
</nm_response>
```

Vault Record Example

```
<nm_response>
<customer_vault>
<customer id="123456">
<first_name>Jack</first_name>
<last_name>Bauer</last_name>
<address_1></address_1>
<address_2></address_2>
<company></company>
<city></city>
<state></state>
<postal_code></postal_code>
<country></country>
<email></email>
<phone></phone>
<fax></fax>
<cell_phone></cell_phone>
<customertaxid></customertaxid>
<website></website>
<shipping_first_name></shipping_first_name>
<shipping_last_name></shipping_last_name>
<shipping_address_1></shipping_address_1>
<shipping_address_2></shipping_address_2>
<shipping_company></shipping_company>
<shipping_city></shipping_city>
<shipping_state></shipping_state>
<shipping_postal_code></shipping_postal_code>
<shipping_country></shipping_country>
<shipping_email></shipping_email>
<shipping_carrier></shipping_carrier>
<tracking_number></tracking_number>
<shipping_date></shipping_date>
<shipping></shipping>
<cc_number>4xxxxxxxxxxx1111</cc_number>
<cc_hash>5a3ce15f56656efa406ec981602fd2e2</cc_hash>
<cc_exp>1010</cc_exp>
<check_account></check_account>
<check_hash></check_hash>
<check_aba></check_aba>
<check_name></check_name>
<account_holder_type></account_holder_type>
<account_type></account_type>
<sec_code></sec_code>
<processor_id></processor_id>
<cc_bin>411111</cc_bin>
<customer_vault_id>123456</customer_vault_id>
<merchant_defined_field id="2">HighPriority</merchant_defined_field>
</customer_vault>
</nm_response>
```

Handling Errors

If the Query API encounters any problems during its operation, an error response will be returned. The example response below was returned when an invalid merchant username was supplied in the request:

```
<?xml version="1.0"?>
<nm_response>
    <error_response>Invalid Username/Password</error_response>
</nm_response>
```

No Matches

If there are no transactions found matching your criteria, it is not handled as an error. An empty nm_response element will be returned.

```
<?xml version="1.0"?>
<nm_response>
</nm_response>
```

How Tos

Set AVS and CVV rules

AVS & CVV Overview

Address Verification Service (AVS) and Card Verification Value (CVV) are basic fraud prevention and verification tools for credit card processing. Address verification service checks the address provided by the customer with the issuing bank. Requiring the address to match helps prevent fraud, as it is unlikely a criminal would know the address and zip code of the cardholder. Including the customers 5 digit zip also helps qualify for the best interchange rates. CVV value has no effect on credit card rates, but it does assist in preventing fraud.

The cardholders billing address can be stored along with the credit card primary account number (PAN) but the CVV can never be stored after the initial transaction per PCI Compliance guidelines.

By configuring AVS and CVV rules in the gateway, merchants can accept or decline transactions based upon the match or mismatch of the information submitted. If a merchant wants to verify (but not charge) a certain card, they can perform a $1.00 authorization (`type=auth`) to obtain the AVS and CVV response variables from the issuing financial institution. The $1.00 auth may or may not show up in the cardholder’s online banking view but if it does appear, it will most often fall off in roughly 3 to 7 days.

When a transaction is processed using Vault ID, the address information is submitted for AVS validation if it was collected.

By default, no AVS or CVV rules are configured in the gateway when a live account is set up. These are configured via Options in the Control Panel.

AVS & CVV settings recommendations

In the Control Panel, via Options, we recommend options (A) and (N), which will ensure that the five digit zip code entered by the customer matches what the issuing financial institution has on file.

For CVV, if you concerned about fraud or want an additional validation tool, we recommend choosing option (N) which checks for match or mismatch.

Your account can be set up in one of two ways:

1. Void on Decline (default)

If a transaction is rejected based upon configured AVS or CVV rules in the gateway, an authorization for the full transaction amount remains on the card until the authorization expires. The issuing financial institution determines the expiration window, but it is typically 3-7 days. Issuing banks may cancel the authorization if contacted directly by the merchant.

2. Pre-authorization

Braintree can set up your account to conduct a pre-authorization of $1.00 to check the submitted values against the CVV and AVS rules set up in the gateway. If the transaction passes the rules, the cardholder will see the full transaction amount on their account. If the transaction fails the CVV/AVS rules, the cardholder will only see an authorization for $1.00 on their card. The full transaction amount will not be attempted.

Merchants typically choose the second option if they are requiring CVV and/or AVS validation and if their tickets are higher. This option avoids tying up money on the customer's card if their transaction is rejected due to these rules.

AVS & CVV settings for international transactions

While Address Verification is extremely effective for cards issued in the United States, very few international card issuers participate. We do not recommend setting up AVS rules for international cards. The settings will default to U.S. Domestic settings when submitting a transaction, so you will need to specify the billing country in order for international AVS rules to apply.

We recommend leaving all boxes unchecked for non-US international transactions. **Applying AVS checking to international transactions can cause a large percentage of your transactions to be declined.**

We also recommend that you select option "N", to reject international transactions if the CVV does not match. Requiring this information will help you contest possible chargebacks, but will not eliminate your ability to process these cards.

AVS and CVV test information

To simulate a CVV Match, pass 999 in the cvv field. Anything else will simulate a mismatch.

- To simulate an AVS Match:
    - Pass 77777 in the zip field for a 'Z – 5 Character Zip match only'.
    - Pass 888 in the address1 field to generate an 'A – Address match only'.
    - Pass them both for a 'Y – Exact match, 5-character numeric ZIP' match.
    - Anything else will simulate a AVS mismatch.

Note Transactions will not fail based on AVS or CVV responses in test mode. The above information will cause the response to be a match or mismatch, but AVS or CVV rules will not cause the transaction to fail. An AVS or CVV mismatch will not prevent a customer from being added to the vault in test mode.

Set IP restrictions

IP Restrictions are a recommended security practice for the Direct Post method, providing increased defense-in-depth measures to prevent unauthorized access. Access is restricted by IP address/host server and can be configured in the Control Panel under Options => IP Restrictions. IP Restrictions work on a whitelist-style filtering. IP restrictions for the Direct Post API and the Control Panel operate independent of one another. For example, the merchant can restrict Direct Post API access to a single IP address, but leave access to the Control Panel open to all locations by leaving that field blank.

It is important to note that IP restrictions do not apply to transactions submitted via the Transparent Redirect API. This is because all communications between the merchant's server and the Braintree Gateway go through the customer's browser, so every transaction will be associated with a different IP address.

Adjust duplicate transaction checking

Our duplicate checking, if enabled, will reject transactions done with the same credit card number, expiration date and amount within a specified time frame. By default, we enable duplicate checking on all live gateway accounts with a 20 minutes interval.

Merchants can request that the time frame be adjusted or duplicate checking to be disabled entirely. Merchants can also adjust the settings on a per transaction basis. Contact Braintree to allow for merchant override and then pass the parameter `dup_seconds`. To disable it for a single transaction, merchants can pass `dup_seconds=0`. Passing parameter `dup_seconds=` will set the time frame. If not specified on individual transactions, the system will default to the account settings.

Duplicate checking can also not reject if the second transaction has a different Order ID. Please email support@getbraintree.com to make an adjustment to this setting.

Send transaction receipts

Transaction receipts can be sent for `type=sale` or `type=capture` transactions. The receipt will only be sent if the transaction is successful.

For all transactions

We can enable email receipts to be sent for all API transactions. If the feature is enabled, but needs to disabled for a specific transaction, include `customer_receipt=false`.

For individual transactions

If you would like to send customer receipts on a transaction by transaction basis, pass `customer_receipt=true`. If using a customer vault ID, ensure there is an email address on the vault record. If passing the credit information, pass the variable `email`.

Customization

Receipts can be customized in either in text or html format. Email Support@getbraintree.com with the desired content.

Set up custom fields

Merchants can capture specific information unique to their business when processing a transaction. This can later be used for internal reporting or customer support purposes. All information captured is available in the Braintree gateway or via the Query API.

Merchants can create as many custom fields as desired. Example fields include customer ID, account number, product part, product description, quantity, Federal Tax ID, SSN, drivers license number, etc.

Creating Fields

To configure a new field, merchants can log into the gateway and go to Options > Merchant Defined Fields. Click the '+' next to the number of the field. Type in a name, then select the type of field that is to be created. 'Text' is the default and permits any text entry. 'Checkbox' is either on or off corresponding to a Yes or No value. 'Radio' and 'Select' provide discrete options from a list of values you choose to provide.

Gateway

After these fields are created, they will appear just like the standard fields in the gateway when you run a transaction.

API

To use these Merchant Defined fields in the API, you will need to use the variable `merchant_defined_field_X`, where X is the number of the Merchant Defined field as assigned in the gateway. As an example, let's assume that a merchant created a custom field called "status" for Merchant Defined Field 1. In order to populate that field through the API you would use, `merchant_defined_field_1=pending`. To use the 2nd Merchant Defined field you would use `merchant_defined_field_2`.

Capture swiped credit card data

For card present retail merchants, we can capture and process the magnetic stripe data collected from a card reader. Our solutions can be integrated with existing Point of Sale systems or merchants can download a small executable file on a computer in a retail environment that will auto populate the transaction form with the swiped credit card information. Additional information can then be entered and the transaction completed.

Variable Name	Format	Description
`track_1`	raw	Raw Magnetic Stripe Data
`track_2`	raw	Raw Magnetic Stripe Data
`track_3`	raw	Raw Magnetic Stripe Data

Capture swiped credit card data from encrypted swiper

Native integration with our HTTPS POST API adding five extra variables:

<u>New API Variables:</u>

magnesafe_track_1

magnesafe_track_2

mangesafe_magneprint

mangesafe_ksn

magnesafe_magneprint_status

There are two separate methods that can be used with the encrypted swipe: Keyboard Emulation and HID.

1. For keyboard emulation, you can swipe the card, and the computer will view the information as a single string of data that has been entered from a keyboard. This information will be encrypted, but you will need to build a javascript or similar application to break out the information into separate variables in order to send the data to us.

2. For HID, you'll need to create a client application (in any language) that can read the swiped track data from the Magensa device and transmit it to our gateway. HID requires more development initially, but you will find more freedom in the long run because the variables are properly recognized separately straight from the swipe.

Either way, the POS device will have to access the fairly raw text coming through in order to craft their request to the gateway. They would not, however have access to the card number, that will already be encrypted by the swiper. The Magensa device can be purchased from us.

Process Level 3 cards

Level 3 processing is a unique class of credit card processing. The Level 3 classification only applies to specially issued Purchase Cards (p-cards) that are able to accept an enhanced set of data about the goods and services that are being purchased.

Your account must have Level 3 processing enabled. Please contact us.

Level III Processing Variables:

shipping	Freight or shipping amount included in the transaction amount *Defaults to 0.00*
tax	The sales tax included in the transaction amount associated with the purchase. Setting tax equal to -1 indicates an order that is exempt from sales tax *Defaults to 0.00*
ponumber	Purchase order number supplied by cardholder
orderid	Identifier assigned by the merchant
shipping_country	Shipping country (ie. US)
shipping_postal	Postal/ZIP code of the address where purchased goods will be delivered This field can be identical to the ship_from_postal if the customer is present and takes immediate possession of the goods.
ship_from_postal	Postal/ZIP code of the address from where purchased goods are being shipped *Defaults to Merchant Profile Postal Code*
summary_commodity_code	4 Character International description code of the overall goods or services being supplied *The Acquirer or processor will provide a list of current codes*
duty_amount	Amount included in the transaction amount associated with the import of the purchased goods *Defaults to 0.00*
discount_amount	Amount included in the transaction amount of any discount applied to the complete order by the merchant *Defaults to 0.00*
national_tax_amount	The national tax amount included in the transaction amount *Defaults to 0.00*
alternate_tax_amount	Second tax amount included in the transaction amount in countries where more than one type of tax can be applied to the purchases *Defaults to 0.00*
alternate_tax_id	Tax identification number of the merchant that reported the alternate tax amount
vat_tax_amount	Contains the amount of any value added taxes which can be associated with the purchased item *Defaults to 0.00*
vat_tax_rate	Contains the tax rate used to calculate the sales tax amount appearing Can contain up to 2 decimal places, ie 1% = 1.00 *Defaults to 0.00*
vat_invoice_reference_number	Invoice number that is associated with the VAT invoice

customer_vat_registration	Value Added Tax registration number supplied by the cardholder
merchant_vat_registration	Government assigned tax identification number of the merchant for whom the goods or services were purchased from.
order_date	Purchase order date supplied in the format YYMMDD *Defaults to the date of the transaction*
item_product_code_#	Merchant defined description code of the item being purchased
item_description_#	Description of the item(s) being supplied
item_commodity_code_#	International description code of the individual good or service being supplied *The Acquirer or processor will provide a list of current codes*
item_unit_of_measure_#	Code for units of measurement as used in international trade *Defaults to EACH*
item_unit_cost_#	Unit cost of item purchased May contain up to 4 decimal places
item_quantity_#	Quantity of the item(s) being purchased *Defaults to 1*
item_total_amount_#	Purchase amount associated with the item *Defaults to: item_unit_cost_# x item_quantity_# rounded to the nearest penny*
item_tax_amount_#	Amount of tax on specific item Amount should not be included in total_amount_# *Defaults to 0.00*
item_tax_rate_#	Percentage representing the value-added tax applied Can contain up to 2 decimal places, ie 1% = 1.00 *Defaults to 0.00*
item_discount_amount_#	Discount amount which can have been applied by the merchant on the sale of the specific item Amount should not be included in total_amount_#
item_discount_rate_#	Discount rate for the line item Can contain up to 2 decimal places, ie 1% = 1.00 *Defaults to 0.00*
item_tax_type_#	Type of value-added taxes that are being used
item_alternate_tax_id_#	Tax identification number of the merchant that reported the alternate tax amount

View full credit card numbers

Our system is configured so that no merchant or user can access unmasked credit card information without the proper permissions. Merchants that use both our Transparent Redirect and Vault will completely eliminate credit card data from ever entering their environment without changing the user experience and maintaining full control over the data. Our approach was designed to try and eliminate the threat of a breach by either rouge internal employees or outside criminals. There are two ways that merchants can access unmasked credit card data:

1) Via the Control Panel, users with the permission enabled can click to “Unlock Payment Data” and will then be prompted to enter their password. Uses can only access a single unmasked credit card at a time. To enable access to unmasked credit cards numbers, you must provide proof of PCI Compliance from a qualified security assessor to Braintree. An account administrator will then need to enable users with this permission by logging into the Control Panel and going to Options > User Accounts > Unlock Payment Data.

2) Via the API, we will enable this feature on your account, but you must first provide proof of PCI Compliance.

This request must be submitted in a POST method.

URL: `https://secure.braintreepaymentgateway.com/api/fetch_full.php`

Variable Name	Description
`username`	Gateway username
`password`	Gateway password
`key`	Unlock key (provided by Braintree)
`transaction_id`	If pulling full credit card number used on a transaction
`customer_vault_id`	If pulling full credit card number stored on a vault record

Note: You can only pull full credit card number for a single transaction id or a single vault record per request.

If successful, the response will read:

response=1&responsetext=&cc_number=4111111111111111

Process transactions with multiple merchant accounts

Each merchant account is set up as a processor ID, which is visible under Options/Load Balancing. Pass variable `processor_id` to route the transaction to the proper merchant account.

Load balance

Merchant's with multiple Merchant Account or Processor IDs can load balance transactions between them using a simple rule-based system.

To configure Load Balancing for a Merchant Account, navigate in the Control Panel to Options => Load Balancing. For each Processor ID listed, make sure that a default value is correctly checked. If any transactions are passed to the API without a specific `processor_id` query pair, the processor listed as default will be used.

Issue a Voice Authorized Transaction

Log in to the Braintree Gateway and select "capture" on the sidebar. Click the fine print "click here" link in blue for voice authorization, and input the given code, credit card number, expiration date and amount. The transaction should then be successful.

For voice authorizations via the API, you will want to include the following in the post string along with the other payment variables: type=offline&authorizationcode=<code supplied by issuer>

Store custom data in the Vault

Merchants can capture specific information unique to their business when processing a transaction. This can later be used for internal reporting or customer support purposes. All information captured is available in the Braintree gateway or via the Query API.

Merchants can create as many custom fields as desired. Example fields include customer ID, account number, product part, product description, quantity, Federal Tax ID, SSN, drivers license number, etc.

Creating Fields

To configure a new field, merchants can log into the gateway and go to Options > Merchant Defined Fields. Click the '+' next to the number of the field. Type in a name, then select the type of field that is to be created. 'Text' is the default and permits any text entry. 'Checkbox' is either on or off corresponding to a Yes or No value. 'Radio' and 'Select' provide discrete options from a list of values you choose to provide.

Gateway

After these fields are created, they will appear just like the standard fields in the gateway when you run a transaction.

API

To use these Merchant Defined fields in the API, you will need to use the variable merchant_defined_field_X, where X is the number of the Merchant Defined field as assigned in the gateway.

As an example, a custom field called "status" is created for Merchant Defined Field 1. In order to populate that field through the API you would use: `merchant_defined_field_1=pending`. To use a second Merchant Defined Field already set up in the control panel, you would use: `merchant_defined_field_2`.

Storing data without payment information

The Vault is primarily set up to store payment information and therefore these fields are required when creating a new Vault record. If you need to store custom information and do not have the customer's credit card or bank account information, you can use the test information (credit card number 4111111111111111 and any future expiration date). Be sure that you only add a customer to the vault and not run an authorization.

AVS and CVV for Vault Transactions

AVS

If address information is stored on the vault record, it will be automatically submitted when a transaction is run for that vault ID. AVS rules apply to vault transactions in the same way they apply to non-vault transactions.

You can pass the address information while running the initial sale and adding the customer to vault in one call. Simply post the fields (Direct Post Method) or include them as hidden fields on your form (Transparent Redirect Method).

Required fields for a credit card transaction:

- `ccnumber`
- `ccexp`
- `type=sale`
- `amount`
- `zip`
- `customer_vault=add_customer`

If you would like to specify the ID for the new vault transaction, pass `customer_vault_id` as well.

We recommend requiring the zip code on your web form and selecting AVS options A and N in the control panel. These settings will only accept transactions where the correct five digit zip code is provided. Passing AVS information assists in lowering the risk of fraud, chargebacks and can help you qualify for lower rates.

CVV

We recommend setting option N in the control panel as well as requiring the field on your web form. This will require that the CVV code is provided and matches what is actually on the card.

CVV is prohibited from being stored at any time according to PCI DSS regulations, therefore it cannot be stored on the vault record to be used for for subsequent vault transactions. On the initial transaction, you can submit the CVV number, which allows you to verify the cardholder. Subsequent transactions using the vault ID do not verify this CVV code again. CVV rules only apply if CVV information is passed with the transaction so vault records will not be rejected. CVV does not have any effect on rates, but does assist in reducing fraud and chargebacks.

Monitor expiration dates for vault transactions

Expiration dates can be monitored using our Query API.

Pass variable `report_type=customer_vault` using the Direct Post method to the query URL: https://secure.braintreepaymentgateway.com/api/query.php. The Query API does not work with the Transparent Redirect method. This will pull all the customer vault records (not transactions). You can then parse the results to review the expiration dates.

Example: `username=testapi&password=password1&report_type=customer_vault`

Updating

To update the expiration date, pass the variables:

`customer_vault=update_customer`

`customer_vault_id=`

`ccexp=`

Direct Post example:

```
username=testapi&password=password1&ccnumber=4111111111111111&ccexp=1010
&customer_vault=add_customer&customer_vault_id=123456
```

Retrieve all transactions for a vault ID

Simply pass the variables below using the Direct Post method to the query URL: https://secure.braintreepaymentgateway.com/api/query.php. The Query API does not work with the Transparent Redirect method.

- `username`
- `password`
- `customer_vault_id`

In the response, you will have a list of all the transactions that were run for this customer ID.

Direct Post Example:

```
username=testapi&password=password1&customer_vault_id=123456
```

Retrieve all Vault records

Pass variable `report_type=customer_vault` using the Direct Post method to the query URL: https://secure.braintreepaymentgateway.com/api/query.php. The Query API does not work with the Transparent Redirect method. This will pull all the customer vault records (not transactions). Transaction information will not be returned.

Direct Post Example:

```
username=testapi&password=password1&report_type=customer_vault
```

Retrieve masked billing information

Pass the variables using the Direct Post method to the query URL: https://secure.braintreepaymentgateway.com/api/query.php. The Query API does not work with the Transparent Redirect method.

- `Username`
- `Password`
- `customer_vault_id`
- `report_type=customer_vault`

In these examples, we are passing customer vault id 123456.

Direct Post Example:

```
username=demo&password=password&customer_vault_id=123456&report_type=customer_vault
```

Add a vault customer to a recurring billing plan

Include the variables:

- `Username`
- `Password`
- `Customer_vault_id`
- `Product_sku_X`
- `Type`
- `Amount`

In this Direct Post example, we are running a $10.00 sale on a credit card:

```
username=testapi&password=password1&customer_vault_id=123456&product_sku_1=monthly&
type=sale&amount=10.00
```

Accept payment in different currencies

First, your account needs to be set up with the ability to accept multiple currencies. If you are unsure about your account, please contact us.

On transactions, you can simply specify the currency along with your other variables by passing `currency=USD` or `currency=EUR`, etc..

The following currencies are supported in the Braintee APIs. The values listed below would be for param `currency`.

Country	Currency	ISO-4217
Australian	**dollar**	`AUD`
Brazilian	**real**	`BRL`
British	**pound**	`GBP`
Canadian	**dollar**	`CAD`
Chinese	**renminbi**	`CNY`
Danish	**krone**	`DKK`
European Union	**euro**	`EUR`
Hong Kong	**dollar**	`HKD`
Indian	**rupee**	`INR`
Japanese	**yen**	`JPY`
Mexico	**peso**	`MXN`
New Zealand	**dollar**	`NZD`
Norwegian	**krone**	`NOK`
Singaporean	**dollar**	`SGD`
South African	**rand**	`ZAR`
South Korean	**won**	`KRW`
Swedish	**krona**	`SEK`
Swiss	**franc**	`CHF`
Taiwanese	**dollar**	`TWD`
U.S.	**dollar**	`USD`

See All Customers Enrolled in a Recurring Billing Plan

1. Log into the control panel and select “List Subscriptions”
2. Download the list into .xls or .csv format
3. Sort the list by plan name (Column A)

Column AA includes the “Reference ID”, which is the transaction id for the transaction that added them to the recurring plan. If you need to remove them from the plan, you can do so by referencing this ID.

Monitor failed recurring transactions

Failed recurring billing transactions can be monitored in a couple ways.

Emailed Reports

You can set up to be emailed settlement reports in the Braintree gateway. Click on Options/User Accounts and enable option “S”. The email will include information regarding both the failed and successful transactions.

API Query

You can also pass variable `condition=failed` using Direct Post to the query URL: https://secure.braintreepaymentgateway.com/api/query.php. The query does not work using Transparent Redirect. This will pull all failed transactions, not just failed recurring billing transactions.

Example:

```
username=testapi&password=password1&condition=failed
```

Set up recurring billing with a free trial

[These instructions are for recurring billing managed by Braintree. If you are managing on your own systems, you will have to write the logic internally.]

When you assign a customer to a recurring billing plan, the customer is not charged the plan amount until the next billing cycle. For example, if the plan bills the customers every 30 days, the customer will not be billed until 30 days after they are added to the plan. At the time of adding them to the plan, you can charge their card a prorated amount, a set up fee, or the full plan amount.

If you are offering a free trial and thus don’t want the customer charged anything before the billing plan takes effect, you can add them to the plan without charging them, but the card will not be validated until the first transaction.

Add a vault customer to a recurring billing plan

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

Include the variables:

- `Username`
- `Password`
- `Customer_vault_id`
- `Product_sku_X`
- `Type`
- `Amount`

In this Direct Post example, we are running a $10.00 sale on a credit card:

```
username=testapi&password=password1&customer_vault_id=123456&product_sku_1=monthly&
type=sale&amount=10.00
```

Set up non regular billing

If you have customers you regularly charge, but not on a regular schedule, you can easily bill them without the security risk or requiring them to enter their billing information each time.

Ask us to enable the use of the Vault on your account and then see our payment API with vault documentation.

The Vault allows you to capture customer payment information once and use a token that represents it for future transactions. You can store these tokens on your system and issue API calls to our gateway when you'd like to bill customers.

Change a customer’s billing plan

Using our API, you will be deleting the customer from the existing plan and adding them to the new plan.

Post URL: `https://secure.braintreepaymentgateway.com/api/transact.php`

Using the full credit card number

Deleting from existing

Pass:

```
username=testapi&password=password1&delete_recurring=123456
```

“123456” is the transaction ID that first added them to the plan.

Adding to the new plan

Pass:

```
username=testapi&password=password1&product_sku_1=monthly& \
ccnumber=4111111111111111&ccexp=1010
```

“product_sku_1=monthly” is the new plan sku. In this example, the customer was added to the new plan and was not charged.

If you wanted to charge the customer a prorated or full amount, you would run a sale by passing:

```
username=testapi&password=password1&product_sku_1=monthly& \
ccnumber=4111111111111111&ccexp=1010&type=sale&amount=10.00
```

If they are downgrading their account, you can choose to do a refund transaction (pass `type=refund` and `transactionid` for the transaction that added the customer to the first plan).

Using a Vault ID

Run two separate calls, one to delete the customer from the existing plan and another to add them to the new plan.

Deleting:

```
username=testapi&password=password1&delete_recurring=123456
```

“123456” is the transaction ID that first added them to the plan.

Adding:

```
username=testapi&password=password1&product_sku_1=monthly& \
customer_vault_id=demo&ccexp=1010&type=auth&amount=1.00
```

If they are downgrading their account, you can choose to do a refund transaction (pass `type=refund` and `transactionid` for the transaction that added the customer to the first plan).

Failure Reasons

If a transaction fails, we provide response text to explain why. Transactions can fail at the processor level (`response=2`) or gateway level (`response=3`). Here's a list of common failure response texts and what they mean.

Processor Decline

`response=2`

The card issuer is rejecting the charge. Common response texts are below, though a more complete list is listed in the Response section of the API you are using.

Auth Decline	Most common decline message. Issuer does not specify the reason for refusing the transaction.
Do Not Honor	Issuer does not specify the reason for refusing the transaction.
Credit Floor	The cardholder has reached their credit limit.
Fraudulent Card	The card has been reported stolen.
Call Voice Oper	The issuer is requiring that you obtain an authorization on the phone before they will approve the transaction.

Gateway Rejection

`response=3`

The gateway is making the transaction fail due to a formatting issue with the data or a permission or option setting. Here are some common response texts

Authentication Failed	IP Restrictions or user permissions are causing the failure.
Expired Hash	The hash must be created within 15 minutes of the current UTC time.
Invalid card number	The card number is not formatted correctly, such as the wrong number of digits or includes characters other than numbers.
Invalid Transaction Id	The referenced Transaction ID does not exist.
Invalid Customer Vault Id	The referenced Customer Vault ID does not exist.
Duplicate transaction	You have the option to reject duplicate transactions, matching card number, expiration date, and amount. Please contact us to determine what your duplicate transaction checking is currently set at or to adjust this timeframe.
You do not have permission to use the customer vault	You need to sign up for the vault if you haven't done so, or we need to enable it for you.
Credits are not enabled	We need to enable your account to issue the credit transaction type.

Transaction Response Codes

The following are the possible response codes along with their meaning. A “Soft Decline” means that the transaction may go through if you try it again later. “Soft Decline (Fixable)” indicates that the transaction could be approved if tried again with different information. A “Hard Decline” will not go through if attempted again.

100	Transaction was approved	Approved
200	Transaction was declined by Processor	Soft Decline
201	Do Not Honor	Soft Decline
202	Insufficient Funds	Soft Decline
203	Over Limit	Soft Decline
204	Transaction not allowed	Hard Decline
220	Incorrect Payment Data	Soft Decline (Fixable)
221	No such card issuer	Hard Decline
222	No card number on file with Issuer	Hard Decline
223	Expired card	Hard Decline
224	Invalid expiration date	Soft Decline (Fixable)
225	Invalid card security code	Soft Decline (Fixable)
240	Call Issuer for further information	Soft Decline (Fixable)
250	Pick up card	Hard Decline
251	Lost card	Hard Decline
252	Stolen card	Hard Decline
253	Fraudulent card	Hard Decline
260	Declined with further instructions available (see response text)	Soft Decline (Fixable)
261	Declined – Stop all recurring payments	Hard Decline
262	Declined – Stop this recurring program	Hard Decline
263	Declined – Updated cardholder data available	Hard Decline
264	Declined – Retry in a few days	Soft Decline
300	Transaction was rejected by gateway	Soft Decline (Fixable)
400	Transaction error returned by processor	Soft Decline (Fixable)
410	Invalid merchant configuration	Soft Decline (Fixable)
411	Merchant account is inactive	Soft Decline (Fixable)
420	Communication error	Soft Decline (Fixable)
421	Communication error with issuer	Soft Decline (Fixable)
430	Duplicate transaction at processor	Soft Decline (Fixable)
440	Processor format error	Soft Decline (Fixable)
441	Invalid transaction information	Soft Decline (Fixable)
460	Processor feature not available	Soft Decline (Fixable)
461	Unsupported card type	Soft Decline (Fixable)

AVS Response Codes

X	Exact match, 9-character ZIP
Y	Exact match, 5-character numeric ZIP
D	Exact match, 5-character numeric ZIP
M	Exact match, 5-character numeric ZIP
A	Address match only
W	9-character numeric ZIP match only
Z	5-character ZIP match only
P	5-character ZIP match only
L	5-character ZIP match only
N	No address or ZIP match
C	No address or ZIP match
U	Address unavailable
G	Non-U.S. Issuer, does not participate
I	Non-U.S. Issuer, does not participate
R	Issuer system unavailable
E	Not a mail/phone order
S	Service not supported
0	AVS Not Available
O	AVS Not Available
B	AVS Not Available

CVV Response Codes

M	CVV2/CVC2 Match
N	CVV2/CVC2 No Match
P	Not Processed
S	Merchant has indicated that CVV2/CVC2 is not present on card
U	Issue is not certified and/or has not provided Visa encryption keys

List of Variables

Payment Variables

https://secure.braintreepaymentgateway.com/api/transact.php

`type` “sale”, “void”, “refund”, “auth”, “capture”, “validate”, “credit”, “update”, or “offline”

`customer_vault` “add_customer”, “update_customer”, or “delete_customer”

Should only be used when making additions or changes to a Vault record.

`orderid`

Required for Level 3 processing.

`amount`

Required for sale, auth, refund and credit transactions.

`ccnumber`

Valid credit card number. Pass if adding/running a credit card.

`ccexp`

Credit card expiration date if adding/running a credit card. (Format: 1010 = October, 2010)

`customer_vault_id`

Specifies the name of the vault ID. If you do not pass a customer_vault_id when adding a customer to the Vault, the Vault will randomly generate one and return it in the response.

`transactionid`

Use to capture, void, refund, or update an existing transaction.

`processor_id`

Required if account has multiple processor ids. The processor_id variable allows merchants to seamlessly manage multiple merchant accounts, sales channels or company divisions. The processor ID is obtained under the Options => Load Balancing section in the Braintree Virtual Terminal Control Panel. If the processor id is not passed, it will default to the first one set up on your account.

`cvv`

Card security code. This value cannot be stored.

`dup_seconds`

Manually set duplicate transaction checking time.

`currency`

Defaults to USD. If your account is set up to accept multiple currencies, include this variable. Format **`currency=USD`**

`ipaddress`

IP address of the cardholder. Format: xxx.xxx.xxx.xxx

`firstname`

`lastname`

`company`

`address1`

`address2`

`city`

`state`

Card billing state. Format: NY (2 Character abbrev.)

`zip`

`country`

Card billing country. Format: CC (ISO-3166). Example country=US for the United States

`phone`

`email`

`orderdescription`

`fax`

`product_sku_1` “nameofplan”

Required for recurring billing plans only. For example, to add a user to a plan called “monthly”, you would pass “product_sku_1=monthly”.

`tax`

Required for Level 2 and Level 3 processing. Total tax amount. Format: x.xx

`taxexempt`

Required for Level 2 and Level 3 processing. Value should be either ‘T’ (true) or ‘F’ (false).

`ponumber`

Required for Level 2 and Level 3 processing. Original Purchase Order/Customer Code number.

`website`

`merchant_receipt_email` “email@merchant.com”

Allows the merchant to specify when to receive transaction receipts.

`merchant_defined_field_#`

For use with custom fields.

`customer_receipt`

Value ‘true’ or ‘false’ specifies when the customer receives a receipt on sale, credit, capture, or offline requests. If multiple customer receipt templates are set-up, you will need to specify the textual value for the correct receipt template.

`authorizationcode`

6 digit code supplied by issuer.

`shipping`

Required for Level 3 processing. Total shipping amount. Format: x.xx

`shipping_firstname`

`shipping_lastname`

`shipping_company`

Company name where the product is being shipped

`shipping_address1`

Shipping address – line 1

`shipping_address2`

Shipping address – line 2

`shipping_city`

`shipping_state`

Shipping state, 2 character abbreviation.

`shipping_zip`

Shipping zip code.

`shipping_country`

Required for Level 3 processing. Format: CC (ISO-3166). Example country=US for the United States

`tracking_number`

Shipping Tracking number

`shipping_carrier` “ups”, “fedex”, “dhl”, or “usps”

Shipping carrier for order, value may only equal the ones listed here.

`descriptor`

Set dynamic descriptor (on supported processors). Company name/DBA section must be either 3, 7 or 12 characters and the product descriptor can be 18, 14, or 9 characters (with an * in between for a total descriptor name of 22 characters).

`descriptor_phone`

Set payment descriptor phone (on supported processors). Can be up to 14 alphanumeric characters.

`billing_method`

Value ‘recurring’. This will override any value specified in the industry variable.

`industry`

Values are ‘ecommerce’ ‘retail’ or ‘moto’ to specify the transaction source.

cc_start_date

Required for Maestro cards. Format is MMYY.

cc_issue_number

Required for Maestro cards. Format is an incremental counter of either one or two characters defined by the issuer.

shipping_postal

Required for Level 3 processing. Postal/ZIP code of the address where purchased goods will be delivered. This field can be identical to the ship_from_postal if the customer is present and takes immediate possession of the goods.

ship_from_postal

Required for Level 3 processing. Postal/ZIP code of the address from where purchased goods are being shipped. Defaults to Merchant Profile Postal Code.

summary_commodity_code

Required for Level 3 processing. 4 Character International description code of the overall goods or services being supplied. The Acquirer or processor will provide a list of current codes.

duty_amount

Required for Level 3 processing. Amount included in the transaction amount associated with the import of the purchased goods. Defaults to 0.00.

discount_amount

Required for Level 3 processing. Amount included in the transaction amount of any discount applied to the complete order by the merchant. Defaults to 0.00.

national_tax_amount

Required for Level 3 processing. The national tax amount included in the transaction amount. Defaults to 0.00.

alternate_tax_amount

Required for Level 3 processing. Second tax amount included in the transaction amount in countries where more than one type of tax can be applied to the purchases. Defaults to 0.00.

alternate_tax_id

Required for Level 3 processing. Tax identification number of the merchant that reported the alternate tax amount.

`vat_tax_amount`

Required for Level 3 processing. Contains the amount of any value added taxes which can be associated with the purchased item. Defaults to 0.00.

`vat_tax_rate`

Required for Level 3 processing. Contains the tax rate used to calculate the sales tax amount appearing. Can contain up to 2 decimal places, ie 1% = 1.00. Defaults to 0.00.

`vat_invoice_reference_number`

Required for Level 3 processing. Invoice number that is associated with the VAT invoice.

`customer_vat_registration`

Required for Level 3 processing. Value Added Tax registration number supplied by the cardholder.

`merchant_vat_registration`

Required for Level 3 processing. Government assigned tax identification number of the merchant for whom the goods or services were purchased from.

`order_date`

Required for Level 3 processing. Purchase order date supplied in the format YYMMDD Defaults to the date of the transaction.

`item_product_code_#`

Required for Level 3 processing. Merchant defined description code of the item being purchased.

`item_description_#`

Required for Level 3 processing. Description of the item(s) being supplied.

`item_commodity_code_#`

Required for Level 3 processing. International description code of the individual good or service being supplied. The Acquirer or processor will provide a list of current codes.

`item_unit_of_measure_#`

Required for Level 3 processing. Code for units of measurement as used in international trade. Defaults to EACH.

`item_unit_cost_#`

Required for Level 3 processing. Unit cost of item purchased. May contain up to 4 decimal places.

`item_quantity_#`

Required for Level 3 processing. Quantity of the item(s) being purchased. Defaults to 1.

`item_total_amount_#`

Required for Level 3 processing. Purchase amount associated with the items. Defaults to: item_unit_cost_# x item_quantity_# rounded to the nearest penny.

`item_tax_amount_#`

Required for Level 3 processing. Amount of tax on specific item. Amount should not be included in total_amount_#. Defaults to 0.00.

`item_tax_rate_#`

Required for Level 3 processing. Percentage representing the value-added tax applied. Can contain up to 2 decimal places, ie 1% = 1.00. Defaults to 0.00.

`item_discount_amount_#`

Required for Level 3 processing. Discount amount which can have been applied by the merchant on the sale of the specific item. Amount should not be included in total_amount_#.

`item_discount_rate_#`

Required for Level 3 processing. Discount rate for the line item. Can contain up to 2 decimal places, ie 1% = 1.00. Defaults to 0.00.

`item_tax_type_#`

Required for Level 3 processing. Type of value-added taxes that are being used.

`item_alternate_tax_id_#`

Required for Level 3 processing. Tax identification number of the merchant that reported the alternate tax amount.

Query Variables

https://secure.braintreepaymentgateway.com/api/query.php

`transaction_id`

`customer_vault_id`

`order_id`

orderid passed with transaction

`last_name`

`email`

`cc_number`

Query using the whole number or the last four digits.

`start_date`

Only transactions that have been modified on or after this date will be retrieved. Note that any actions performed on a transaction (ie. a VOID) will cause the modified date to be updated.

`end_date`

Only transactions that have been modified on or before this date will be retrieved. Note that any actions performed on a transaction (ie. a VOID) will cause the modified date to be updated.

`condition` pending, pendingsettlement, failed, canceled, complete, or unknown

Retrieves only transactions with the specified transaction condition. A combination of the values above can be used, and should be separated by commas.

`transaction_type` “ck” or “cc”

`action_type` “sale”, “refund”, “credit”, “auth”, “capture”, or “void”

A combination of the values above can be used, and should be separated by commas. For example, to retrieve all transactions with credit or refund actions, use the following: refund,credit

`report_type` “customer_vault”

Used to pull back a vault record instead of a transaction.

`merchant_defined_field_#`