Moneris Checkout Integration Guide

Version: 1.0.13

Table of Contents

Getting Help

Moneris has help for you at every stage of the integration process.

Getting Started	During Development	Production
Contact our Client Integration Specialists: clientintegrations@moneris.com	If you are already working with an integration specialist and need technical development assistance, contact our eProducts Technical Consultants: 1-866-319-7450 eproducts@moneris.com	If your application is already live and you need production support, contact Moneris Customer Service: onlinepayments@moneris.com 1-866-319-7450 Available 24/7

For additional support resources, you can also make use of our community forums at

http://community.moneris.com/product-forums/

System and Skills Requirements

In order to integrate with Moneris Checkout as a merchant, you must have:

- An e-commerce website with a back-end server

For development, you should have some understanding of the following:

- JavaScript
- JSON
- Server-side programming

Additionally, for Google Pay™ integration, all your front-end web pages must use the HTTPS protocol.

Changes in v1.0.13

- Adjusted format of **startdate** for the optional Recurring object within preload requests. Field supports YYYY/MM/DD or YYYY-MM-DD format.
- Added Escalate response code E to the Kount Result Code field **code** found in "Definition of Response Fields – Response to Receipt Request Receipt Object Fields" on page 60
- Removed Cardholder Authentication Value **cavv** from the response fields found in "Definition of Response Fields – Response to Receipt Request Receipt Object Fields" on page 60
- Added 3DS Transaction Status Reason field **transStatusReason** to the 3DS object in the Fraud section of in "Definition of Response Fields – Response to Receipt Request Receipt Object Fields" on page 60
- Added Data Key Format field **data_key_format** to the optional Preload request fields in "Preload Request " on page 16
- Updated "Kount as a Fraud Tool in Moneris Checkout" on page 89 with information on Kount User Defined Fields for consuming 3DS data. Merchants using Kount Enterprise may need to consult these UDFs as part of their Kount management portal configuration.

Changes in v1.0.12

- Added **shipping amount** field to the pre-load request as a general optional field
- Added **advice code** field to the "Definition of Response Fields – Response to Receipt Request Receipt Object Fields" on page 60
- Added Callback Response 2004 to "Callback Response Codes – Moneris Checkout" on page 97

Changes in v1.0.11

- Added fields for 3DS 2.2 to the Response to Receipt Request: **ThreeDSVersion, AuthenticationType, ThreeDSACSTransID, ThreeDSAuthTimeStamp, DSTransID**

Changes in v1.0.10

- Clarified limits for request field **request type**
- Amended JSON sample code for Preload request and Response to Receipt Request

Changes in v1.0.8

- Added new array object **token** to the Preload request, which contains one to three pairs of **data key** and **issuer ID** request fields; each pair within token represents an instance of a payment card stored in the Moneris Vault
- Added new topic for the optional Token object
- Added new request field **prompt for CVV** in Preload request
- Added new topic about Pay by Token: Tokens and Pay By Token
- Added new response field **pay by token** in Response to Receipt Request
- Amended data structure diagrams in Preload Request
- Corrected value for INTERAC® in response field **card type**
- Added topic about Installments

Changes in v1.0.7

- Added convenience fee response field information to Definition of Response Fields - Response to Receipt Request and the sample JSON response
- Added information to Tokenization of Credentials in Moneris Checkout to indicate that Moneris Checkout now supports updating of tokens

- Added explication about adding and updating tokens to the description of the tokenization message response field in the Definition of Response Fields - Response to Receipt Request
- Added topic about test cards
- Changed name of convenience fee response fields to reflect service fee name change
- Corrected human name and description for the details field in the CVD sub-object in the Fraud object in Response to Receipt Request
- New data object in the Response to Receipt Request, **vault data** object, with two new response fields **data key** and **is data key valid**
- Updated sample code for Example Preload Request and Example Response to Receipt Request
- Amended data structure diagrams in Response to Receipt Request topics
- Added note about billing-related fields and 3-D Secure in the Billing Fields Object topic
- Added new request URLs for testing and production in Implementing Preload Server-to-Server Logic, Implementing Receipt Request Server-to-Server Logic, Testing Your Moneris Checkout Solution and Moving to Production with Moneris Checkouttopics
- Added new Checkout JS path URLs for testing and production in Preparing Your Client-Side Checkout Page, Testing Your Moneris Checkout Integration and Moving to Production with Moneris Checkout topics
- Added API token for testing Convenience Fee in Test Cards for Moneris Checkout

Changes in v1.0.6

- Changed limits for request fields **shipping province** and **billing province** to 2 characters

Changes in v1.0.5

- Corrected limit for the request field **start date**

Changes in v1.0.4

- Added new callback types Page Closed and Payment Submitted
- Changed references to the **monerisCheckout** object to **myCheckout**
- Added information about restricted special character " in some request fields
- Removed reference to restricted special characters in **items**. **description** in the Shopping Cart object, as they are now supported for that field

Changes in v1.0.3

- Added new response field **isDebit** to Response to Receipt Request
- Added support for Apple Pay and Google Pay™ wallet transactions

Changes in v1.0.2

- Corrected the limit of **order number** response field to 45 characters

Changes in v1.0.1

- Added information about 3-D Secure 2.0, including a new response field, **transaction status**
- Added information about Multi-Currency Pricing
- Updated diagrams and sample code to reflect 3-D Secure 2.0 and Multi-Currency Pricing
- Corrected limits of request fields in the Preload request: order number, customer ID, dynamic descriptor
- Corrected limits of request fields in the Recurring Billing object: number of recurs
- Corrected limits of request fields in the Shopping Cart object: item description, item product code, tax description
- Corrected limits of request fields in the Shipping Details object: shipping address line 1 and 2, shipping city, shipping province, shipping country, shipping postal code

- Corrected limits of request fields in the Billing Details object: billing address line and 2, billing city, billing province, billing postal code

- Added additional information about the behaviour of callbacks in the topics Payment Receipt and Payment Complete

- In the Definition of Response Fields - Response to Receipt Request, corrected description of the response field **3-D Secure code**

- In Callback Response Codes topic, corrected the description of code 2001 and added new codes 2002 and 2003

1 About Moneris Checkout

Moneris Checkout gives e-commerce merchants a simple and secure way to process payments by integrating a Moneris-hosted payment module into the merchant checkout page.

2 Building Your Moneris Checkout Integration

2.1 Configuring Moneris Checkout in Merchant Resource Center

The first step is to configure your Moneris Checkout page in the Moneris Merchant Resource Center (MRC).

In the initial stage of development, you create a test configuration in the testing MRC. Once the solution is ready to be deployed to production, you must create a new, separate configuration for the production environment in the production MRC.

The **checkout ID** is the key value that is generated after the configuration is completed and used within the Preload Request in order to identify the specific Moneris Checkout configuration.

To get the checkout ID and start configuring your page, do the following:

1. Log into the Merchant Resource Center at one of the following URLs (according to your stage of development)
   Testing: https://esqa.moneris.com/mpg

   Production: https://www3.moneris.com/mpg

2. In the Admin menu, select **Moneris Checkout Config**

3. Click the **Create Profile** button
4. Follow the on-screen steps to complete the configuration

For more information, see the Merchant Resource Center documentation available for download on the Moneris developer portal at:

developer.moneris.com

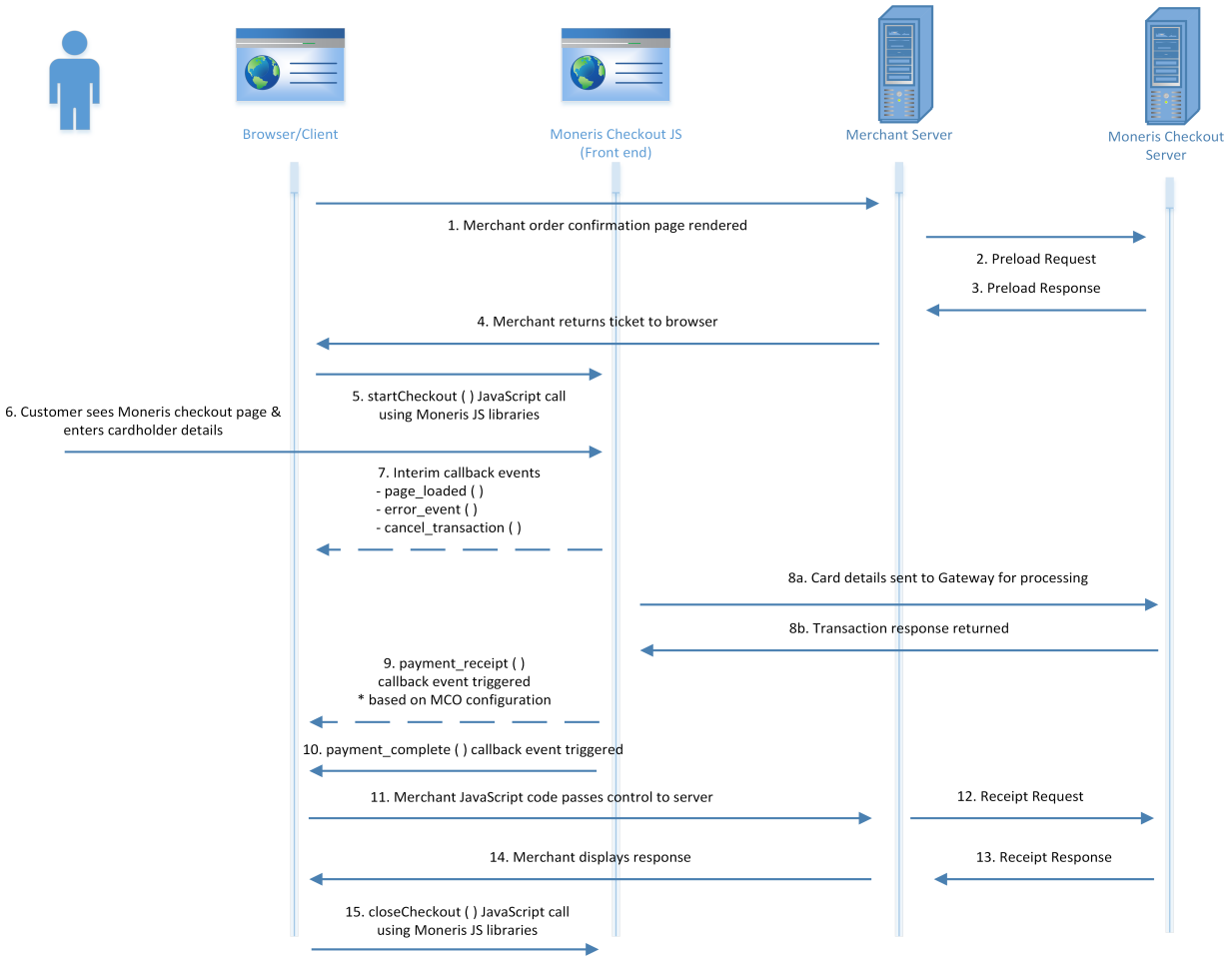
2.1.1 Additional Features to Configure in the MRC

There are other features of the Moneris Checkout page that you can enable using the configurator in the Merchant Resource Center. They include:

- Tokenization of credentials
- Fraud tool behaviour
- Window sizing
- Multi-Currency Pricing

For more on configuring these features, see 3 Additional Features in Moneris Checkout.

2.2 Moneris Checkout Transaction Process Flow



2.3 Preparing Your Client-Side Checkout Page

In order to prepare your client-side checkout page for interacting with Moneris Checkout, you need to do a few tasks first:

1. Add a call to the Moneris Checkout JavaScript library in a `<script>` tag:

   Testing:

   ```
   <script src="https://gatewayt.moneris.com/chktv2/js/chkt_
   v2.00.js"></script>
   ```

   Production:

   ```
   <script src="https://gateway.moneris.com/chktv2/js/chkt_
   v2.00.js"></script>
   ```

2. Create a `<div>` in the HTML:

```
<div id="monerisCheckout"></div>
```

   a. (optional): If you are not using the "Full screen" window sizing option, you will need to define the size of your window by creating another `<div>` around this one, for example:

```
<div id="outerDiv" style="width:400px"; height"300px">

<div id="monerisCheckout"></div>

</div>
```

3. Instantiate the **monerisCheckout** object and set it up:

```
var myCheckout = new monerisCheckout();

myCheckout.setMode("qa");

myCheckout.setCheckoutDiv("monerisCheckout");
```

4. Set callbacks in JavaScript:

```
myCheckout.setCallback("page_loaded",myPageLoad);

myCheckout.setCallback("cancel_transaction",myCancelTransaction);

myCheckout.setCallback("payment_receipt",myPaymentReceipt);

myCheckout.setCallback("payment_complete",myPaymentComplete);
```

For more information about callbacks in Moneris Checkout, see 2.6 Handling Callbacks.

2.4 Implementing Preload Server-to-Server Logic

The Preload request is the means by which a Moneris Checkout instance is securely generated at transaction time. It involves a server-to-server post using the JSON format documented in 2.4.1 Preload Request .

The response to the Preload request returns a ticket number which uniquely identifies the instance and must be passed in the JavaScript `myCheckout.startCheckout (ticket #)` request in order to display the Moneris Checkout page in the browser.

> **NOTE:** The ticket number expiration time is set to 30 minutes.

In your server implementation, use the following Moneris Checkout URLs to post to, depending on the development stage:

Testing:

https://gatewayt.moneris.com/chktv2/request/request.php

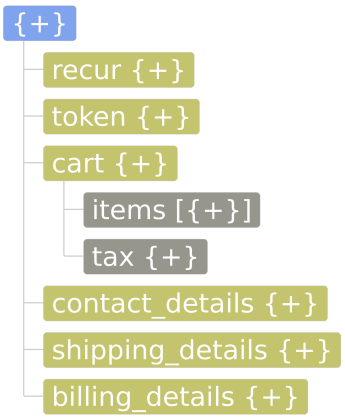
Production:

https://gateway.moneris.com/chktv2/request/request.php

2.4.1 Preload Request

Transaction requests are sent to the Moneris Checkout server using JSON.

JSON structure overview for Preload request



Request fields for Preload request – Required

Variable Name	Type and Limits	Description
store ID `store_id`	*String* N/A	Unique identifier provided by Moneris upon merchant account setup
API token `api_token`	*String* N/A	Unique alphanumeric string assigned upon merchant account activation
checkout ID `checkout_id`	*String* 30-character alphanumeric	Identifies your Moneris Checkout configuration; this is given to you when you configure your page in the Merchant Resource Center

Variable Name	Type and Limits	Description
transaction total amount `txn_total`	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	The total dollar amount of the transaction
developmental mode `environment`	*String* alphabetic	Indicates the stage of development you are sending the request for: testing = qa production = prod
request type `action`	*String* alphabetic case-sensitive, lowercase only	Type of request being made to Moneris Checkout server Allowable values: `preload` or `receipt`

Request fields for Preload request – Optional

Variable Name	Type and Limits	Description
order number `order_no`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	The order number is a unique identifier appended to every financial transaction
customer ID `cust_id`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center

Variable Name	Type and Limits	Description
Data Key Format `data_key_format`	*String* 2-character alphanumeric	Specifies the data key format being returned. If left blank, data key format will default to 25-character alpha-numeric. **NOTE:** If the request uses a format with uniqueness, indicated by suffix [U], any attempt to tokenize a card that is already stored under this merchant account will return the existing token instead of a new token. Possible values: 0 = 25 character alphanumeric data key 0U = 0 = unique 25 character alpha-numeric data key 1 = first 6, last 4 from card number with random characters between. Preserves card length. 1U = first 6, last 4 from card number with random characters between. Preserves card length. Unique. 2 = first 6, last 4 from card number with random characters between. Does not preserve card length. 2U = first 6, last 4 from card number with random characters between. Does not preserve card length. Unique. 4U = first 6, last 4 from card number. 7th character is random alpha, others between are random. Forces 16 character length. Unique.
dynamic descriptor `dynamic_descriptor`	*String* 20-character alphanumeric total of 22 characters including your merchant name and separator	Merchant-defined description sent on a per-transaction basis that will appear on the credit card statement appended to the merchant's business name

Variable Name	Type and Limits	Description
	NOTE: Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Dependent on the card issuer, the statement will typically show the dynamic descriptor appended to the merchant's existing business name separated by the "/" character; additional characters will be truncated **NOTE:** The 22-character maximum limit must take the "/" into account as one of the characters
language `language`	*String* 2-character alphabetic	Determines which language Moneris Checkout will display information in Allowable values: en – English fr – French
shipping amount `shipping_amount`	*String* 10-character decimal	Shipping cost of the items to be shipped
data key `data_key`	*String* 25-character alphanumeric	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile Data key is generated by Moneris and returned to you in the Receipt object when the profile is first registered **NOTE:** Only send this field with Vault Card Update, or else the Preload request will be rejected; this feature is only available when the Preload transaction type sent is a Card Verification
prompt for CVV `ask_cvv`	*String* 1-character alphabetic Y or N	When set to Y, Moneris Checkout will prompt the cardholder to enter their CVV when they select a payment card that has been stored as a token **NOTE:** This field is only applicable when sending the Token object

Additional request objects in Preload request – Optional

Variable Name	Type and Limits	Description
Recurring Billing recur	*Object* N/A	Contains fields related to Recurring Billing
Shopping Cart `cart`	*Object* N/A	The virtual shopping cart and its contents
Contact Details `contact_details`	*Object* N/A	Customer contact information This object is returned in the Response to Receipt Request as the Customer Information response object (`cust_info`)
Shipping Details `shipping_details`	*Object* N/A	Customer shipping information
Billing Details `billing_details`	*Object* N/A	Customer billing information
token `token`	*Array Object* N/A	Array object containing between 1 to 3 pairs of a unique data key and an associated issuer ID, each pair representing an instance of a payment card stored in the Moneris Vault; used for Pay By Token transactions.

2.4.1.1 Optional Preload Request Objects

Moneris Checkout also allows you to send optional objects in the Preload request that reflect additional information entered by the customer at checkout, enable additional features, or meet transaction processing requirements.

Optional objects you can use include:

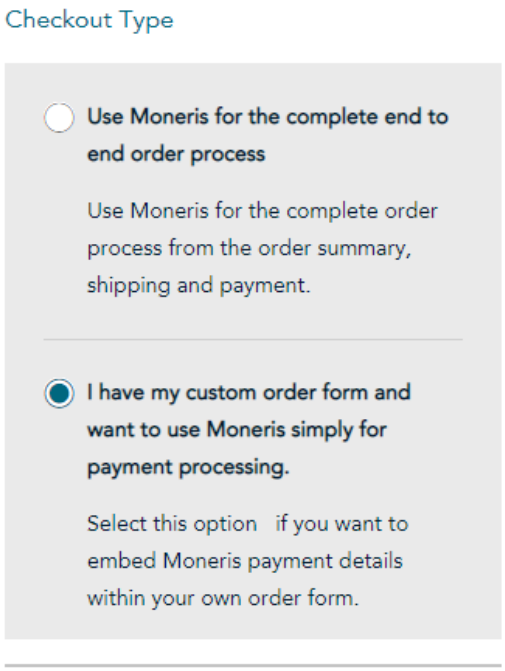
- Recurring Billing Object
- Shopping Cart Object
- Contact Details Object
- Shipping Details Object
- Billing Details Object

If you have configured Moneris Checkout to handle these additional items, you do not need to send the corresponding object in the Preload request.

- For Recurring Billing, Shopping Cart, and Token objects, only send these optional objects if you are using your own e-commerce page to collect them separately from Moneris Checkout.

- For Customer, Shipping, and Billing Details objects, you may configure Moneris Checkout to handle collecting information AND send it in the Preload request as well.

The following screenshot shows what you select in the Merchant Resource Center if you are collecting additional items on your own e-commerce page:



Recurring Billing Object

Optional object

Include this object in Preload request to indicate the start of a series of Recurring Billing transactions that will be managed by Moneris.

> **NOTE:** Recurring Billing is not allowed when using Multi-Currency Pricing or Gift Cards.

Top level object field

`recur`

Request fields for Recurring Billing object

Variable Name	Type and Limits	Description
number of recurs number_of_recurs	*String* numeric 1-999	The number of times that the transaction must recur
period recur_period	*String* numeric 1-999	Number of recur unit intervals that must pass between recurring billings
recurring amount recur_amount	*String* 10-character decimal, minimum three digits Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Dollar amount of the recurring transaction This amount will be billed on the start date, and then billed repeatedly based on the interval defined by period and recur unit
recur unit recur_unit	*String* day, week, month or eom	Unit to be used as a basis for the interval Works in conjunction with the period

Variable Name	Type and Limits	Description
		variable to define the billing frequency
start date start_date	*String* YYYY/MM/DD or YYYY-MM-DD format	Date of the first future recurring billing transaction; this must be a date in the future If an additional charge will be made immediately, the start now variable must be set to true
bill now bill_now	*String* `true` or `false`	Set to true if a charge will be made against the card immediately; otherwise set to false

Shopping Cart Object

Optional object

The shopping cart object can contain multiple items (each item is represented as its own array within the Shopping Cart object).

Top level object field

`cart`

Request fields for Shopping Cart object

Variable Name	Type and Limits	Description
shopping cart items `items`	*Object* sub-object containing arrays, nested within `cart` contains following items in blue	Encapsulates the entire array of items in the shopping cart
item URL **`items.url`**	*String* alphanumeric	URL that corresponds to the image of the Moneris Checkout shopping cart item
item description **`items.description`**	*String* 200-character alpha-	Describes the item in the shopping cart

Variable Name	Type and Limits	Description
	numeric	
item product code `items.product_code`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	The SKU for the item
item unit cost `items.unit_cost`	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Per-unit cost of the item
item quantity `items.quantity`	*String* numeric 6 characters maximum	Number of individual instances of the given item in the shopping cart
subtotal `subtotal`	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Total dollar amount of the shopping cart, before taxes
tax `tax`	*Object* sub-object nested within `cart` contains following items in blue	Contains information related to taxes charged on the items in the shopping cart
tax amount	*String*	Dollar amount of taxes

Variable Name	Type and Limits	Description
`tax.amount`	10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	
tax description `tax.description`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Describes type of tax being applied
tax rate `tax.rate`	*String* Must be a number with up to 3 decimal places **EXAMPLE:** xx or xx.x or xx.xx or xx.xxx	Percentage tax rate charged

Contact Details Object

Optional object

Top level object field

`contact_details`

Request fields for Contact Details object

Variable Name	Type and Limits	Description
first name `first_name`	*String* 30-character alphanumeric	Customer first name
last name	*String*	Customer last name

Variable Name	Type and Limits	Description
`last_name`	30-character alphanumeric	
email `email`	*String* 255-character alpha-numeric	Customer email
phone number `phone`	*String* 30-character alphanumeric	Customer phone number

Shipping Details Object

Optional object

Top level object field

`shipping_details`

Request fields for Shipping Details object

Variable Name	Type and Limits	Description
shipping address line 1 `address_1`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address
shipping address line 2 `address_2`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address
shipping city `city`	*String* 50-character alphanumeric **NOTE:**	Customer shipping address city

Variable Name	Type and Limits	Description
	Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	
shipping province `province`	*String* 2-character alphanumeric	Customer shipping address province Country subdivision ISO 3166-2
shipping country `country`	*String* 2-character alphanumeric	Customer shipping address country ISO 3166-1 alpha-2
shipping postal code `postal code`	*String* 20-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address postal code

Billing Details Object

Optional object

> **NOTE:** Billing-related fields are required when sending 3-D Secure authentication transactions, or else the authentication process may fail.

Top level object field

`billing_details`

Request fields for Billing Details object

Variable Name	Type and Limits	Description
billing address line 1 `address_1`	*String* 50 character alphanumeric	Customer billing address

Variable Name	Type and Limits	Description
	NOTE: Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	
billing address line 2 `address_2`	*String* 50 character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer billing address
billing city `city`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer billing address city
billing province `province`	*String* 2-character alphanumeric	Customer billing address province Country subdivision ISO 3166-2
billing country `country`	*String* 2-character alphabetic	Customer billing address country ISO 3166-1 alpha-2
billing postal code `postal code`	*String* 20-character alphanumeric	Customer billing address postal code

Token Object

Optional array object

Array object containing between 1 to 3 pairs of a unique data key and an associated issuer ID, each pair representing an instance of a payment card stored in the Moneris Vault. Used for Pay By Token transactions.

Moneris Checkout will accept a maximum three token pairs of data key and issuer ID for each customer

Top level object field

`token`

Request fields for Token object

Variable Name	Type and Limits	Description
data key `data_key`	*String* 25-character alphanumeric	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile Data key is generated by Moneris and returned to you in the Receipt object when the profile is first registered
issuer ID issuer_id	*String* 15-character alphanumeric variable length	Unique identifier for the cardholder's stored credentials Sent back in the response from the card brand when processing a Credential on File transaction If the cardholder's credentials are being stored for the first time, and the issuer ID was returned in the response, you must save the issuer ID on your system to use in subsequent Credential on File transactions (applies to merchant-initiated transactions only) The issuer ID must be saved to your systems when returned from Moneris Gateway in the response data, regardless if the value was received or not As a best practice, if the issuer ID is not returned and you received a value of NULL instead, store that value and send it in the subsequent transaction

2.4.1.2 Example Preload Request JSON

> **NOTE:** This example reflects a Preload request with all optional objects; the code is for illustrative purposes only and is not executable.

```
{
   "store_id":"moneris",
   "api_token":"hurgle",
   "checkout_id":"chkt5BF66neris",
   "txn_total":"452.00",
   "environment":"qa",
   "action":"preload",
   "token": [
    {
      "data_key": "abc123datakey1",
      "issuer_id": "645sddfvdrt4tefd"
    },
    {
      "data_key": "abc123datakey2",
      "issuer_id": "645sddfvdrt4tefd"
    },
    {
      "data_key": "abc123datakey3",
      "issuer_id": "645sddfvdrt4tefd"
    }
  ],
  "ask_cvv":"Y"
   "order_no":"",
   "cust_id":"chkt - cust - 0303",
   "data_key_format":"2",
   "dynamic_descriptor":"dyndesc",
   "language":"en",
   "shipping_amount":"200.00",
   "recur":{
      "bill_now":"true",
      "recur_amount":"1.00",
      "start_date":"2021-11-21",
      "recur_unit":"month",
      "recur_period":"1",
      "number_of_recurs":"10"
```

```
},
"cart":{
   "items":[
      {
         "url":"https:\/\/example.com\/examples\/item1.jpg",
         "description":"One item",
         "product_code":"one_item",
         "unit_cost":"100.00",
         "quantity":"1"
      },
      {
         "url":"https:\/\/example.com\/examples\/item2.jpg",
         "description":"Two item",
         "product_code":"two_item",
         "unit_cost":"200.00",
         "quantity":"1"
      },
      {
         "url":"https:\/\/example.com\/examples\/item3.jpg",
         "description":"Three item",
         "product_code":"three_item",
         "unit_cost":"100.00",
         "quantity":"1"
      }
   ],
   "subtotal":"400.00",
   "tax":{
      "amount":"52.00",
      "description":"Taxes",
      "rate":"13.00"
   }
},
"contact_details":{
   "first_name":"bill",
   "last_name":"smith",
   "email":"test@moneris.com",
   "phone":"4165551234",

},
"shipping_details":{
   "address_1":"1 main st",
   "address_2":"Unit 2012",
   "city":"Toronto",
```

```
      "province":"ON",
      "country":"CA",
      "postal_code":"M1M1M1",
   },
   "billing_details":{
      "address_1":"1 main st",
      "address_2":"Unit 2000",
      "city":"Toronto",
      "province":"ON",
      "country":"CA",
      "postal_code":"M1M1M1",
   }
}
```

2.4.2 Response to Preload Request

Response Fields – Response to Preload Request

Variable Name	Description
response `"response":{`	Top level response object
success `"success":`	Denotes whether the Preload request was successful
ticket `"ticket":`	Identifies the specific Moneris Checkout instance Only returned if success = true
error `"error":{`	Sub-object that encapsulates all errors that occurred as a result of the Preload request Only returned if success = false
data `"data":`	Describes the specific type of error that occurred as a result of some aspect of the Preload request

2.4.2.1 Example Preload Response – Successful Preload

```
{
   "response":{
      "success":"true",
      "ticket":"1585G9G9GIKKGGGIGIOG09G9OGKGJFKFJFNjuit8g9"
   }
}
```

2.4.2.2 Example Preload Response – Failed Preload

```
{
   "response":{
      "success":"false",
      "error":{
         "billing_details":{
```

```
            "data":"billing address must be set when AVS is enabled"
         }
      }
   }
}
```

2.5 Displaying the Moneris Checkout Page in the Browser

When a customer goes to check out their items for purchase, the Moneris Checkout page is displayed in the tag you created on your web site

To insert the Moneris Checkout instance into the , you call the JavaScript function:

```
myCheckout.startCheckout([ticket #])
```

2.6 Handling Callbacks

Callbacks are the means by which Moneris Checkout communicates with your merchant checkout page. All callbacks include a single parameter defined as a JSON-formatted string.

In order to handle callbacks, you need to create JavaScript functions that receive the callbacks being sent by Moneris Checkout when the events occur. These are the functions being referred to as part of the callback set methods, as described in 2.3 Preparing Your Client-Side Checkout Page.

2.6.1 Callback Types

These callbacks are required to be included in the JavaScript of your page:

- Page Loaded
- Cancel Transaction
- Payment Receipt
- Payment Complete
- Page Closed
- Payment Submitted

2.6.1.1 Callback Response Fields

Variable Name	Type and Limits	Description
handler `handler`	*String* alphanumeric	Describes the type of callback being used Possible values: cancel_transaction page_loaded payment_complete payment_receipt
ticket `ticket`	*String* alphanumeric	Identifies the specific Moneris Checkout instance This is also returned in the response to the original Preload
response code `response_code`	*String* alphanumeric	Identifies the result of the callback For information on response codes, see Callback Response Codes – Moneris Checkout

2.6.1.2 Page Loaded

Callback Use

To get the page loaded status of the Moneris Checkout page.

This callback is called once the Moneris Checkout is loaded.

JavaScript Set Method for Callback

```
myCheckout.setCallback("page_loaded",myPageLoad);
```

JSON Response Message Format

```
{
   "handler":"page_loaded",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
```

```
   "response_code":"001"
}
```

2.6.1.3 Cancel Transaction

Callback Use

This callback is called in the event the cardholder presses the cancel button in Moneris Checkout.

Standard is to call the `closeCheckout()` method to close the Moneris Checkout `<div>`.

The `closeCheckout( )` method will need to be called and a new Preload request will be required in order to initiate a new Moneris Checkout instance.

JavaScript Set Method for Callback

```
myCheckout.setCallback("cancel_transaction",myCancelTransaction);
```

JSON Response Message Format

```
{
   "handler":"cancel_transaction",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code":"001"
}
```

2.6.1.4 Payment Receipt

Callback Use

Transaction is complete and receipt is ready to be collected.

If you have chosen to have Moneris Checkout generate the receipt, this callback is called once the Moneris Checkout displays the transaction receipt.

If you have chosen Moneris Checkout not to generate a receipt, this callback will not be called. For information on when to obtain the receipt response for the transaction, refer to the Payment Complete callback.

JavaScript Set Method for Callback

```
myCheckout.setCallback("payment_receipt",myPaymentReceipt);
```

JSON Response Message Format

```
{
   "handler": "payment_receipt",
   "ticket": "1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code": "001"
}
```

2.6.1.5 Payment Complete

Callback Use

This callback is called once Moneris Checkout has completed payment.

If you have chosen Moneris Checkout to generate a receipt, the cardholder has to return to your Checkout page in order for the callback to be called. For information on obtaining the receipt response for the transaction, refer to the Payment Receipt callback .

Moneris Checkout should be closed by calling the `closeCheckout()` method

JavaScript Set Method for Callback

```
myCheckout.setCallback("payment_complete",myPaymentComplete);
```

JSON Response Message Format

```
{
   "handler":"payment_complete",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code":"001"
}
```

2.6.1.6 Page Closed

Callback Use

Called when the user is on the payment page and has submitted payment, but tries to close the window, clicks the back button in the browser or reloads the page before the payment has been confirmed, causing a JavaScript error to occur.

Moneris Checkout should be closed by calling the **closeCheckout()** method. The payment proceeds, with no changes to the payment flow.

JavaScript Set Method for Callback

```
myCheckout.setCallback("page_closed",myPageClosed);
```

JSON Response Message Format

When the user closes the window, clicks back or reload in the browser:

```
{"handler":"page_closed", "response_code":"001"}
```

When a JavaScript error occurs:

```
{
   "handler":"page_closed",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code":"001"
}
```

2.6.1.7 Payment Submitted

Callback Use

This callback is called will be triggered when cardholder clicks Checkout button and payment processing is started.

JavaScript Set Method for Callback

```
myCheckout.setCallback("payment_submitted",myPaymentSubmitted);
```

JSON Response Message Format

```
{
   "handler":"payment_submitted",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code":"001"
}
```

2.7 Implementing Receipt Request Server-to-Server Logic

Once the Payment Complete callback has been called, your merchant website can make the server-to-server Receipt Request call in order to obtain the details of the transaction for the receipt and to determine whether the transaction was approved or declined.

In your server implementation, use the following Moneris Checkout URLs to post to, depending on the development stage:

Testing:

https://gatewayt.moneris.com/chktv2/request/request.php

Production:

https://gateway.moneris.com/chktv2/request/request.php

2.7.1 Receipt Request

Once the transaction is finished, you can request the receipt details from the Moneris Checkout server.

Request fields for Receipt Request – Required

Variable Name	Type and Limits	Description
store ID `store_id`	*String* N/A	Unique identifier provided by Moneris upon merchant account set up
API token `api_token`	*String* N/A	Unique alphanumeric string assigned upon merchant account activation
checkout ID `checkout_id`	*String* 30-character alphanumeric (maximum)	Identifies your Moneris Checkout configuration; this is given to you when you configure your page in the Merchant Resource Center
ticket number `ticket`	*String* maximum 40-character alphanumeric	The unique ticket number that identifies a particular transaction; this returned in the response to the Preload request
developmental mode `environment`	*String* alphabetic	Indicates the stage of development you are sending the request for: testing = qa production = prod
request type `action`	*String* alphabetic case-sensitive, lowercase	Type of request being made to Moneris Checkout server Allowable values:

Variable Name	Type and Limits	Description
	only	`preload` or `receipt`

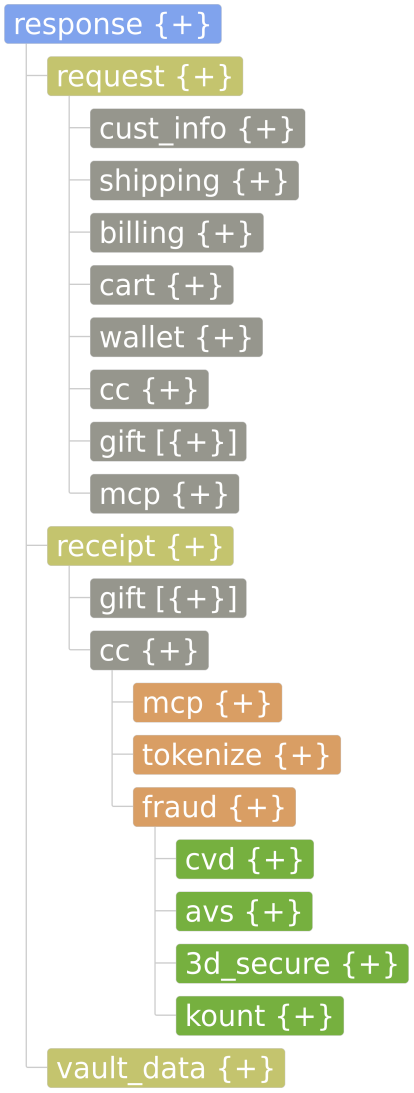
2.7.1.1 Example Receipt Request JSON

```
{
   "store_id":"example_storeId",
   "api_token":"example_apiToken",
   "checkout_id":"example_checkoutId",
   "ticket":"1539966660vfTyEASfnwNrsQqFE8VkMAOcN169zt",
   "environment":"qa",
   "action":"receipt"
}
```

2.7.2 Response to Receipt Request

Responses to Receipt Requests can contain multiple, nested response objects.

JSON structure for Response to Receipt Request



WARNING: Do not code your solution to strictly consume a mapping of the fields contained within the response. These responses are subject to frequent updates with new additional fields added.

2.7.2.1 Definition of Response Fields - Response to Receipt Request Top Level Fields

Response Field Name and Key	Type and Limits	Description
response `{"response":{`	*Object* N/A	Top level response object
success `"success":`	*String* true/false	Contains information relating to the Preload request and other information that Moneris Checkout sends to the Moneris Gateway when processing the financial transaction. Possible values: true or false

2.7.2.2 Definition of Response Fields - Response to Receipt Request - Request Within Response Object Fields

The following are fields that may be returned in the Response to Receipt Request Request object within the response, shown with nesting

Response Field Name and Key	Type and Limits	Description
request `"request":{`	*Object* N/A	Contains information relating to the Preload request and other information that Moneris Checkout sends to the Moneris Gateway when processing the financial transaction
transaction total amount `"txn_total":`	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	The total dollar amount of the transaction

Response Field Name and Key	Type and Limits	Description
Customer Information `"cust_info":{`	*Object* N/A	Customer contact information The information presented in this response object will reflect one of three scenarios: • If sent in the Preload request, this object will echo the Contact Details object • if Moneris Checkout is set to handle the customer contact information, it will reflect what the customer entered in the web form • If Moneris Checkout was set to not ask for this information, the response object will be empty
first name `"first_name":`	*String* 30-character alphanumeric	Customer first name
last name `"last_name":`	*String* 30-character alphanumeric	Customer last name
phone number `"phone":`	*String*	Customer phone number

Response Field Name and Key	Type and Limits	Description
	30-character alpha-numeric	
email `"email":`	*String* 255-character alpha-numeric	Customer email
Shipping `"shipping":{`	*Object* N/A	Contains customer shipping information The information presented in this response object will reflect one of three scenarios: • If sent in the Preload request, this object will echo the Shipping Details object • if Moneris Checkout is set to handle the customer shipping information, it will reflect what the customer entered in the web form • If Moneris Checkout was set to not ask for this information, the response object will be empty
shipping address line 1 `"address_1":`	*String* 50-character alpha-numeric	Customer shipping address

Response Field Name and Key	Type and Limits	Description
	NOTE: Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	
shipping address line 2 `"address_2":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address
shipping city `"city":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address city
shipping country `"country":`	*String* 2-character alpha-numeric	Customer shipping address country ISO 3166-1 alpha-2
shipping province `"province":`	*String* 2-character alpha-numeric	Customer shipping address province Country subdivision ISO 3166-2
shipping postal code `"postal_code":`	*String* 20-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address postal code
Billing	*Object*	Contains customer

Response Field Name and Key	Type and Limits	Description
`"billing":{`	N/A	billing information The information presented in this response object will reflect one of three scenarios: • If sent in the Preload request, this object will echo the Billing Details object • if Moneris Checkout is set to handle the customer billing information, it will reflect what the customer entered in the web form • If Moneris Checkout was set to not ask for this information, the response object will be empty
billing address line 1 `"address_1":`	*String* 50 character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer billing address
billing address line 2 `"address_2":`	*String* 50 character alpha-	Customer billing address

Response Field Name and Key	Type and Limits	Description
	numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	
billing city `"city":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer billing address city
billing country `"country":`	*String* 2-character alpha-betic	Customer billing address country ISO 3166-1 alpha-2
billing province `"province":`	*String* 2-character alpha-numeric	Customer billing address province Country subdivision ISO 3166-2
billing postal code `"postal_code":`	*String* 20-character alpha-numeric	Customer billing address postal code
same as shipping `"same_as_shipping":`	*String* true/false	Indicates whether the shipping address is the same as the billing address Possible values: `true` or `false`
Recurring Billing `"recur":{`	*Object* N/A	Contains fields related to Recurring Billing
number of recurs `"number_of_recurs":`	*String* numeric	The number of times that the transaction must recur

Response Field Name and Key	Type and Limits	Description
	1-999	
period `"recur_period":`	*String* numeric 1-999	Number of recur unit intervals that must pass between recurring billings
recurring amount `"recur_amount":`	*String* 10-character decimal, minimum three digits Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Dollar amount of the recurring transaction This amount will be billed on the start date, and then billed repeatedly based on the interval defined by period and recur unit
recur unit `"recur_unit":`	*String* day, week, month or eom	Unit to be used as a basis for the interval Works in conjunction with the period variable to define the billing frequency
start date `"start_date":`	*String* YYYY/MM/DD or YYYY-MM-DD format	Date of the first future recurring billing transaction; this must be a date in the future If an additional charge will be made immediately, the start now variable must be set to true
bill now `"bill_now":`	*String* `true` or `false`	Set to true if a charge will be made against the card immediately; otherwise set to false
Shopping Cart	*Object*	The virtual shopping cart and its contents

Response Field Name and Key	Type and Limits	Description
`"cart":{`	N/A	This echos the information contained in the Shopping Cart request object
shopping cart items `"items":[{`	*Object* N/A	Encapsulates the entire array of items in the shopping cart
item URL `"url":`	*String* alphanumeric	URL that corresponds to the image of the Moneris Checkout shopping cart item
item description `"description":`	*String* 200-character alphanumeric	Describes the item in the shopping cart
item product code `"product_code":`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	The SKU for the item
item unit cost `"unit_cost":`	*String* 10-character decimal	Per-unit cost of the item
item quantity `"quantity":`	*String* numeric 6 characters maximum	Number of individual instances of the given item in the shopping cart
subtotal `"subtotal":`	*String* 10-character decimal	Total dollar amount of the shopping cart, before taxes
tax `"tax":`	*Object* N/A	Contains information related to taxes charged on the items in the shopping cart

Response Field Name and Key	Type and Limits	Description
tax amount `"amount":`	*String* 10-character decimal	Dollar amount of taxes
tax description `"description":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Describes type of tax being applied
tax rate `"rate":`	*String* Must be a number with up to 3 decimal places **EXAMPLE:** xx or xx.x or xx.xx or xx.xxx	Percentage tax rate charged
credit card total `"cc_total":`	*String* 10-character decimal	Total amount being charged to the credit card
token `"token":`	*Object* N/A	Top level object containing details about token/data key selected by customer for payment.
data key `"data key":`	*String* 25-character alpha-numeric	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile
issuer ID `"issuer_id":`	*String* 15-character alpha-numeric variable length	Unique identifier for the cardholder's stored credentials Sent back in the response from the card brand when processing

Response Field Name and Key	Type and Limits	Description
		a Credential on File transaction If the cardholder's credentials are being stored for the first time, and the issuer ID was returned in the response, you must save the issuer ID on your system to use in subsequent Credential on File transactions (applies to merchant-initiated transactions only) The issuer ID must be saved to your systems when returned from Moneris Gateway in the response data, regardless if the value was received or not As a best practice, if the issuer ID is not returned and you received a value of NULL instead, store that value and send it in the subsequent transaction
vault `"vault":`	*Object* N/A	Object containing info on if the token is valid or not
success (tokenize) `"success":`	*String* true/false	Indicates whether the card was successfully tokenized Possible values: `true` or `false`
first 4 last 4 `"first4last4":`	*String* 11-character numeric	The first 4 and last 4 digits of the card
data key	*String*	Unique identifier for a

Response Field Name and Key	Type and Limits	Description
`"datakey":`	25-character alpha-numeric	Vault profile, and used in future Vault financial transactions to associate a transaction with that profile Data key is generated by Moneris and returned to you in the Receipt object when the profile is first registered
expiry date `"exp_date":`	*String* 4-character numeric	Card expiry date
tokenization status `"status":`	*String* 3-character numeric	Specifies what type of failure, if any, occurred during the tokenization request Possible values: 001 = Successful creation of a temporary token 940 = Invalid profile id (on tokenization request) 941 = Error generating token 942 = Invalid Profile ID, or source URL 943 = Card data is invalid (not numeric, fails mod10, we will remove spaces) 944 = Invalid expiration date (mmyy, must be current month or in the future) 945 = Invalid CVD data (not 3-4 digits)
tokenization message `"message":`	*String* alphabetic	Provides additional details about the success or failure of the tokenization Message will reflect whether details have been added or updated

Response Field Name and Key	Type and Limits	Description
customer ID `"cust_id":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center
phone number `"phone":`	*String* 30-character alpha-numeric	Customer's phone number Can be sent in when creating or updating a Vault profile
email address `"email":`	*String* 30-character alpha-numeric	Customer's email address Can be sent in when creating or updating a Vault profile
pan `"pan":`	*String* null	Credit card number - always null
expiry date `"exp_date":`	*String* 4-character numeric	Card expiry date
electronic commerce indicator `"eci":`	*String* 1-character numeric	The e-commerce indicator or crypt type that was used to process the transaction Possible values are: 5 - Authenticated e-commerce transaction (3-D Secure) 6 - Non-authenticated e-commerce transaction (3-D Secure) 7 - SSL-enabled merchant
Credit Card (request)	*Object*	Contains cardholder

Response Field Name and Key	Type and Limits	Description
`"cc":{`	N/A	information
first 6 last 4 `"first6last4":`	*String* 10-character numeric	First 6 and last 4 digits of card number
expiry date `"expiry":`	*String* 4-character numeric	Card expiry date
cardholder `"cardholder":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Cardholder name
Multi-Currency Pricing in the Preload `"mcp":{`	*Object* N/A	Contains fields related to Multi-Currency Pricing sent in the transaction
merchant settlement amount `"merchant_settlement_amount":`	*String* 12-character decimal	The amount the merchant will receive in the transaction, in Canadian dollars
cardholder currency code `"cardholder_currency_code":`	*String* 3-character numeric	ISO code representing the foreign currency of the cardholder
Gift (request) `"gift":[{`	*Object* N/A	Object containing information about a gift card
balance remaining `"balance_remaining":`	*String* 10-character decimal	The remaining balance on the gift card
gift card description `"description":`	*String*	Description of the gift card used for the transaction
first 4 last 4	*String*	The first 4 and last 4

Response Field Name and Key	Type and Limits	Description
`"first4last4":`		digits of the card
gift card number `"pan":`	*String*	The account number of the gift card
gift card CVD `"cvd":`	*String*	Card validation digits on the gift card
balance used `"balance_used":`	*String* 10-character decimal	The amount that was removed from the card's balance as part of the transaction
Wallet `"wallet":`	*Object* N/A	Contains information from the digital wallet that was used in the transaction
wallet type `"type":`	*String* `applepay` or `googlepay`	Contains information from the digital wallet that was used in the transaction
payment data `"paymentData":{`	*Object* N/A	Object containing various information related to the payment sent from the digital wallet
API version (minor) `"apiVersionMinor":`	*String*	Minor version of the API
API version `"apiVersion":`	*String*	Version of the digital wallet's payment API
payment method data `"paymentMethodData":{`	*Object* N/A	Object containing information about the payment method used in the transaction
payment method description `"description":`	*String*	User-facing message to describe the payment method that funds this transaction

Response Field Name and Key	Type and Limits	Description
tokenization data `"tokenizationData":{`	*Object* N/A	Object containing information related to tokenization and the digital wallet
tokenization type `"type":`	*String*	The type of tokenization to be applied to the selected payment method Possible values: `PAYMENT_GATEWAY` or `DIRECT`
token `"token":`	*String*	The generated payment method token
payment method type `"type":`	*String*	A short identifier for the supported payment method Possible values: CARD PAYPAL
info `"info":{`	*Object* N/A	Object that echoes information about the cardholder, the card and the card network from the digital wallet
card network `"cardNetwork":`	*String*	The payment card network
card details `"cardDetails":`	*String*	The details about the card; this value is commonly the last four digits of the selected payment account number
digital wallet billing address `"billingAddress":{`	*Object* N/A	Object that echoes the cardholder's billing information from the digital wallet

Response Field Name and Key	Type and Limits	Description
address 3 `"address3":`	*String*	Third line of the address
sorting code `"sortingCode":`	*String*	The sorting code
address 2 `"address2":`	*String*	Second line of the address
country code `"countryCode":`	*String*	ISO 3166-1 alpha-2 country code
address 1 `"address1":`	*String*	First line of the address
postal code `"postalCode":`	*String*	Address postal code or ZIP
name `"name":`	*String*	Name of the addressee
locality `"locality":`	*String*	City, town, neighbourhood, or suburb
administrative area `"administrativeArea":`	*String*	A country subdivision, such as a state or province
digital wallet shipping address `"shippingAddress":{`	*Object* N/A	Object containing the cardholder's default shipping address information stored in the digital wallet
address 3 `"address3":`	*String*	Third line of the address
sorting code `"sortingCode":`	*String*	The sorting code

Response Field Name and Key	Type and Limits	Description
address 2 `"address2":`	*String*	Second line of the address
country code `"countryCode":`	*String*	ISO 3166-1 alpha-2 country code
address 1 `"address1":`	*String*	First line of the address
postal code `"postalCode":`	*String*	Address postal code or ZIP
name `"name":`	*String*	Name of the addressee
locality `"locality":`	*String*	City, town, neighbourhood, or suburb
administrative area `"administrativeArea":`	*String*	A country subdivision, such as a state or province
pay by token `"pay_by_token":`	*String*	For a pay by token transaction, indicates whether the customer used an existing tokenized payment card or added a new card in order to pay for the transaction Possible values: 1 – indicates that the cardholder used an existing tokenized card to pay 0 – indicates that cardholder added a new card in order to pay, and the merchant sent a new corresponding data key

Response Field Name and Key	Type and Limits	Description
ticket number `"ticket":`	*String* maximum 40-character alphanumeric	The unique ticket number that identifies a particular transaction; this returned in the response to the Preload request
customer ID `"cust_id":`	*String* 50-character alphanumeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center
dynamic descriptor `"dynamic descriptor":`	*String* 20-character alphanumeric total of 22 characters including your merchant name and separator **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Merchant-defined description sent on a per-transaction basis that will appear on the credit card statement appended to the merchant's business name Dependent on the card issuer, the statement will typically show the dynamic descriptor appended to the merchant's existing business name separated by the "/" character; additional characters will be truncated **NOTE:** The 22-character maximum limit must take the "/" into account as one of the characters
order number `"order_no":`	*String* 50-character alphanumeric **NOTE:** Some special characters	The order number is a unique identifier appended to every financial transaction

Response Field Name and Key	Type and Limits	Description
	are not allowed: < > $ % = ? ^ " { } [ ] \	
electronic commerce indicator `"eci":`	*String* 1-character numeric	The e-commerce indicator or crypt type that was used to process the transaction Possible values are: 5 - Authenticated e-commerce transaction (3-D Secure) 6 - Non-authenticated e-commerce transaction (3-D Secure) 7 - SSL-enabled merchant

2.7.2.3 Definition of Response Fields – Response to Receipt Request Receipt Object Fields

The following are fields that may be returned in the Response to Receipt Request Receipt object, shown with nesting

Response Field Name and Key	Type and Limits	Description
Receipt `"receipt":{`	*Object* N/A	Object containing the receipt information
result (financial transaction) `"result":`	*String* 1-character alphabetic `a` or `d`	Indicates the result of the financial transaction Possible values are: a = Accepted d = Declined
Gift (receipt) `"gift":[{`	*Object* N/A	Contains information related to gift card
order number `"order_no":`	*String* 50-character alphanumeric	The order number is a unique identifier appended to every financial trans-

Response Field Name and Key	Type and Limits	Description
	NOTE: Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	action
transaction number `"transaction_no":`	*String* 20-character alpha-numeric	Moneris Gateway-specific transaction identifier This field is required for any future follow-on transaction requests, such as Refund, Purchase Correction and Pre-Authorization Completion transactions
reference number `"reference_no":`	*String* 18-character alpha-numeric	Terminal used to process the transaction, followed by the shift, batch and sequence number This data is typically used to reference transactions on the host systems, and must be displayed on any receipt presented to the customer This information should be stored by the merchant **EXAMPLE** 660123450010690030 66012345: Terminal ID 001: Shift number 069: Batch number 003: Transaction number within the batch.
response code `"response_code":`	*String* 3-character numeric or null	Transaction response code Possible values are: <50 – transaction approved >=50 –transaction declined

Response Field Name and Key	Type and Limits	Description
		NULL – transaction was not sent for authorization For more details on specific response, please see the Response Codes reference topic
benefit amount `"benefit_amount":`	*String* 10-character decimal	This is the benefit that was generated for the transaction; the amount that was removed from the card as part of the transaction
benefit remaining `"benefit_remaining":`	*String* 10-character decimal	The remaining balance on the gift card
first 6 last 4 `"first6last4":`	*String* 10-character numeric	First 6 and last 4 digits of card number
Credit Card (receipt) `"cc":{`	*Object* N/A	Contains fields describing the response to the credit card transaction
order number `"order_no":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ { } [ ] \ MCP orders have a "_mcp" suffix Card verification orders have a "_veri"suffix	The order number is a unique identifier appended to every financial transaction
customer ID `"cust_id":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ { } [ ] \	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center This is the echo of the same customer ID sent in the Pre-

Response Field Name and Key	Type and Limits	Description
		load request
transaction number `"transaction_no":`	*String* 20-character alpha-numeric	Used to reference the original transaction when performing a follow-on transaction (i.e., Pre-Authorization Completion, Purchase Correction or Refund) This value is returned in the response of the original transaction Pre-Authorization Completion: references a Pre-Authorization Refund/Purchase Correction: references a Purchase or Pre-Authorization Completion
reference number `"reference_no":`	*String* 18-character alpha-numeric	Terminal used to process the transaction, followed by the shift, batch and sequence number This data is typically used to reference transactions on the host systems, and must be displayed on any receipt presented to the customer This information should be stored by the merchant **EXAMPLE** 660123450010690030 66012345: Terminal ID 001: Shift number 069: Batch number 003: Transaction number within the batch.

Response Field Name and Key	Type and Limits	Description
transaction code `"transaction_code":`	*String* 2-character alpha-numeric	Type of financial transaction that was performed Possible values: 00 – Purchase 01 – Pre-Authorization
transaction type `"transaction_type":`	*String* 2-character numeric	ISO transaction code for financial transaction
transaction date and time `"transaction_date_time":`	*String* YYYY-MM-DD HH:MM:SS	Processing host date and time stamp Format: YYYY-MM-DD HH:MM:SS
corporate card `"corporateCard":`	*String* true/false	Indicates whether the payment card is a corporate card
credit card amount `"amount":`	*String* 10-character decimal	The total dollar amount that was charged to the credit card
response code `"response_code":`	*String* 3-character numeric or null	Transaction response code Possible values are: <50 – transaction approved >=50 –transaction declined NULL – transaction was not sent for authorization For more details on specific response, please see the Response Codes reference topic
ISO response code `"iso_response_code":`	*String* 2-character numeric	ISO response code returned from issuing institution For more details on specific ISO codes returned, see the Response Codes reference topic

Response Field Name and Key	Type and Limits	Description
approval code `"approval_code":`	*String* 8-character alpha-numeric	Authorization code returned from the issuing institution
card type `"card_type":`	*String* 2-character alpha-numeric	Type of payment card used to process the transaction Allowable values: V = Visa M = Mastercard AX = American Express DC = Diner's Card NO = Novus/Discover SE = Sears P = INTERAC® Debit C1 = JCB 00 = Unknown Card Type
wallet type `"wallet_type":`	*String* `applepay` or `googlepay`	Type of digital wallet used in this transaction Possible values: `applepay` or `googlepay`
dynamic descriptor `"dynamic_descriptor":`	*String* 20-character alpha-numeric total of 22 characters including your merchant name and separator **NOTE:** Some special characters are not allowed: < > $ % = ? ^ { } [ ] \	Merchant-defined description sent on a per-transaction basis that will appear on the credit card statement appended to the merchant's business name Dependent on the card issuer, the statement will typically show the dynamic descriptor appended to the merchant's existing business name separated by the "/" character; additional characters will be truncated

Response Field Name and Key	Type and Limits	Description
		NOTE: The 22-character maximum limit must take the "/" into account as one of the characters
invoice number `"invoice_number":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ { } [ ] \	Identifies an invoice number associated with the transaction
customer code `"customer_code":`	*String* 50-character alpha-numeric **NOTE:** Some special characters are not allowed: < > $ % = ? ^ { } [ ] \	User-defined identifier
electronic commerce indicator `"eci":`	*String* 1-character numeric	The e-commerce indicator or crypt type that was used to process the transaction Possible values: 5 - Authenticated e-commerce transaction (3D-Secure) 6 - Non-authenticated e-commerce transaction (3D-Secure) 7 - SSL-enabled merchant
CVD result code `"cvd_result_code":`	*String* 2-character alpha-numeric	Indicates the CVD validation result The first byte is the numeric CVD indicator sent in the request; the second byte is the response code Possible response codes are shown in the CVD Response

Response Field Name and Key	Type and Limits	Description
		Codes reference
AVS result code `"avs_result_code":`	*String* 1-character alpha-numeric	Indicates the address verification result For a full list of possible response codes refer to the AVS Response Codes reference
CAVV result code `"cavv_result_code":`	*String* 1-character alpha-numeric	Indicates the 3-D Secure CAVV result Possible response codes are shown in the tables in 6.4 CAVV Result Codes
first 6 last 4 `"first6last4":`	*String* 10-character numeric	First 6 and last 4 digits of card number
expiry date `"expiry_date":`	*String* 4-character alpha-numeric	Expiry date of the card MMYY format
recur success `"recur_success":`	*String* true/false	Indicates whether the recurring billing transaction has been successfully set up for future billing Possible values: `true` or `false`
issuer ID `"issuer_id":`	*String* 15-character alpha-numeric variable length	Unique identifier for the cardholder's stored credentials Sent back in the response from the card brand when processing a Credential on File transaction If the cardholder's credentials are being stored for the first time, and the issuer ID was returned in the response, you must

Response Field Name and Key	Type and Limits	Description
		save the issuer ID on your system to use in sub-sequent Credential on File transactions (applies to merchant-initiated trans-actions only) The issuer ID must be saved to your systems when returned from Moneris Gateway in the response data, regardless if the value was received or not As a best practice, if the issuer ID is not returned and you received a value of NULL instead, store that value and send it in the sub-sequent transaction
is debit `"is_debit":`	*String* true/false	Indicates whether a debit card was used in the trans-action
advice code `"advice_code":`	*String* 3-character numeric	Merchant advice code returned by Mastercard for financial transactions
ECR (electronic cash register) number `"ecr_no":`	*String* 8-character numeric	Terminal ID/ECR Number from the request
batch number `"batch_no":`	*String* 3-character numeric	Batch number; also presen-ted as a component of the reference number
sequence number `"sequence_no":`	*String* 3-character numeric	Transaction number within the batch; also presented as a component of ref-erence number
result (financial transaction) `"result":`	*String* 1-character alpha-numeric	Indicates the result of the financial transaction Possible values are: a = Accepted

Response Field Name and Key	Type and Limits	Description
		d = Declined
convenience fee/service fee success `"cf_success":`	*String* true/false	Indicates whether the Convenience Fee transaction processed successfully **NOTE:** Convenience fee is not supported for MCP, recurring billing, tokenization or digital wallets
convenience fee/service fee amount `"cf_fee_amt":`	*String* 9-character decimal	The Convenience Fee amount **NOTE:** Convenience fee is not supported for MCP, recurring billing, tokenization or digital wallets
convenience fee/service fee rate `"cf_fee_rate":`	*String* 9-character decimal	The convenience fee rate that has been defined on the merchant's profile **EXAMPLE** 1.00 – a fixed amount or 10.0 - a percentage amount **NOTE:** Convenience fee is not supported for MCP, recurring billing, tokenization or digital wallets
convenience fee/service fee type `"cf_fee_type":`	*String* 3-character alphabetic Possible values: AMT/PCT	The type of convenience fee that has been defined on the merchant's profile Possible values: AMT – fixed amount PCT – percentage **NOTE:** Convenience fee is not

Response Field Name and Key	Type and Limits	Description
		supported for MCP, recurring billing, tokenization or digital wallets
convenience fee/service fee status `"cf_status":`	*String* 2-character alpha-numeric	Indicates the status of the merchant and convenience fee transactions The convenience fee status field provides details about the transaction behaviour and should be referenced when contacting Moneris Customer Support Possible values: 1 or 1F – Completed 1st purchase transaction 2 or 2F – Completed 2nd purchase transaction 3 – Completed void transaction 9 or 9F – Completed 1st void transaction 10 or 10F – Completed 2nd void transaction **NOTE:** Convenience fee is not supported for MCP, recurring billing, tokenization or digital wallets
Tokenize `"tokenize":{`	*Object* N/A	Contains information related to the tokenization of cardholder credentials
success (tokenize) `"success":`	*String* true/false	Indicates whether the card was successfully tokenized Possible values: `true` or `false`
first 4 last 4 `"first4last4":`	*String* 11-character numeric	The first 4 and last 4 digits of the card

Response Field Name and Key	Type and Limits	Description
data key `"data_key":`	*String* 25-character alpha-numeric	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile
tokenization status `"status":`	*String* 3-character numeric	Specifies what type of failure, if any, occurred during the tokenization request Possible values: 001 = Successful creation of a temporary token 940 = Invalid profile id (on tokenization request) 941 = Error generating token 942 = Invalid Profile ID, or source URL 943 = Card data is invalid (not numeric, fails mod10, we will remove spaces) 944 = Invalid expiration date (mmyy, must be current month or in the future) 945 = Invalid CVD data (not 3-4 digits)
tokenization message `"message":`	*String* alphabetic	Provides additional details about the success or failure of the tokenization Message will reflect whether details have been added or updated
Multi-Currency Pricing in the Response `"mcp":{`	*Object* N/A	Contains fields related to Multi-Currency Pricing received in the response
merchant settlement amount `"merchant_settlement_amount":`	*String* 12-character decimal	The amount the merchant will receive in the transaction, in Canadian dollars
cardholder currency code	*String*	ISO code representing the

Response Field Name and Key	Type and Limits	Description
`"cardholder_currency_code":`	3-character numeric	foreign currency of the cardholder
multi-currency pricing rate `"mcp_rate":`	*String*	The foreign exchange rate (foreign currency to CAD) that was used for the transaction
decimal precision `"decimal_precision":`	*String* 1-character numeric Possible values: 0/1/2/3	Decimal precision of the amount
cardholder amount `"cardholder_amount":`	*String* 12-character numeric	Amount, in units of foreign currency, the cardholder will be charged on the transaction
cardholder currency description `"cardholder_currency_desc":`	*String* 3-character alphanumeric	Describes the foreign currency being used in the transaction
Fraud `"fraud":{`	*Object* N/A	Contains sub-objects that describe information related to fraud tool inquiries
CVD `"cvd":{`	*Object* N/A	Contains information related to the CVD fraud tool
decision origin `"decision_origin":`	*String* alphanumeric, see description for possible values	Possible values: `Moneris` or `Merchant`
CVD result `"result":`	*String* 1-character numeric	Possible values: 1 = Success 2 = Failed 3 = Not performed

Response Field Name and Key	Type and Limits	Description
		4 = Card not eligible
condition `"condition":`	*String* 1-character numeric	Indicates whether this fraud tool was set as a factor for Moneris to use when making an automatic decision on a transaction Possible values are as follows: 0 = Optional 1 = Mandatory
status `"status":`	*String* alphabetic, see description for possible values	Indicates whether the fraud tool inquiry was performed, and if it was used for auto-decisioning purposes Possible values: success = Fraud tool successful failed = Fraud tool failed (non-auto decision) disabled = Fraud tool not performed ineligible = Fraud tool was selected but card is not a credit card or card not eligible failed_optional = Fraud tool failed and auto decision is optional failed_mandatory = Fraud tool failed auto decision is mandatory
CVD code `"code":`	*String* 2-character alpha-numeric	CVD result code; for a list of possible codes see the CVD Response Codes reference
details `"details":`	*String* N/A - details is not used for this fraud tool	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure

Response Field Name and Key	Type and Limits	Description
AVS `"avs":{`	*Object* N/A	Contains information related to the AVS fraud tool
decision origin `"decision_origin":`	*String* alphanumeric, see description for possible values	Possible values: `Moneris` or `Merchant`
AVS result `"result":`	*String* 1-character numeric	Possible values: 1 = Success 2 = Failed 3 = Not performed 4 = Card not eligible
condition `"condition":`	*String* 1-character numeric	Indicates whether this fraud tool was set as a factor for Moneris to use when making an automatic decision on a transaction Possible values are as follows: 0 = Optional 1 = Mandatory
status `"status":`	*String* alphabetic, see description for possible values	Indicates whether the fraud tool inquiry was performed, and if it was used for auto-decisioning purposes Possible values: success = Fraud tool successful failed = Fraud tool failed (non-auto decision) disabled = Fraud tool not performed ineligible = Fraud tool was selected but card is not a credit card or card not eligible failed_optional = Fraud tool failed and auto decision is optional

Response Field Name and Key	Type and Limits	Description
		failed_mandatory = Fraud tool failed auto decision is mandatory
AVS code `"code":`	*String* 1-character alpha-numeric	AVS result code; for a list of potential codes, see the AVS Response Codes reference
details `"details":`	*String* N/A - details is not used for this fraud tool	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
3-D Secure `"3d_secure":{`	*Object* N/A	Contains information related to the 3-D Secure fraud tool
decision origin `"decision_origin":`	*String* alphanumeric, see description for possible values	Possible values: `Moneris` or `Merchant`
3-D Secure result `"result":`	*String* 1-character numeric	Possible values: 1 = Success 2 = Failed 3 = Not performed 4 = Card not eligible
condition `"condition":`	*String* 1-character numeric	Indicates whether this fraud tool was set as a factor for Moneris to use when making an automatic decision on a transaction Possible values are as follows: 0 = Optional 1 = Mandatory
status `"status":`	*String* alphabetic, see	Indicates whether the fraud tool inquiry was performed, and if it was used for auto-

Response Field Name and Key	Type and Limits	Description
	description for possible values	decisioning purposes Possible values: success = Fraud tool successful failed = Fraud tool failed (non-auto decision) disabled = Fraud tool not performed ineligible = Fraud tool was selected but card is not a credit card or card not eligible failed_optional = Fraud tool failed and auto decision is optional failed_mandatory = Fraud tool failed auto decision is mandatory
3-D Secure code `"code":`	*String* 1-character numeric	The crypt type that was used to process the transaction Possible values: 5 = Authenticated e-commerce transaction (3-D Secure) 6 = Non-authenticated e-commerce transaction (3-D Secure) 7 = SSL-enabled merchant
3DS Transaction Status Reason `"transStatusReason":`	*String* 2-character numeric	The crypt type that was used to process the transaction Possible values: 01 = Card authentication failed 02 = Unknown device 03 = Unsupported device 04 = Exceeds authentication frequency limit 05 = Expired card 06 = Invalid card number 07 = Invalid transaction

Response Field Name and Key	Type and Limits	Description
		08 = No card record 09 = Security failure 10 = Stolen card
details `"details":{`	*Object* N/A - details is not used for this fraud tool	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
3-D Secure message `"message":`	*String* alphabetic	Describes the reasoning for the outcome of the 3-D Secure inquiry Possible values: "Authentication Not Available" "Unable to Verify Enrollment" "Successful Payer Authentication" "Cardholder Not Participating" "failed 3-D Secure authentication" "Successful Merchant Attempt"
VERes `"VERes":`	*String* 1-character alphabetic	Verification response code Possible values: N = The card/issuer is not enrolled U = The card type is not participating Y = The card is enrolled **NOTE:** Only returned for 3-D Secure1.0 transactions
PARes `"PARes":`	*String* true/false	Payer authentication response code Possible values: true = Fully authenticated or attempted to verify PIN

Response Field Name and Key	Type and Limits	Description
		false = Failed to authenticate **NOTE:** Only returned for 3-D Secure 1.0 transactions
transaction status `"transStatus":`	*String* 1-character alphabetic	Indicates whether a transaction qualifies as an authenticated transaction or account verification Possible values: Y = Cardholder has been fully authenticated A = A proof of authentication attempt was generated U = Authentication could not be performed dude to technical or other issues N = Not authenticated R = Not authenticated because the Issuer is rejecting authentication and requesting that authorization not be attempted **NOTE:** Only returned for 3-D Secure2.0
load 3-D Secure `"loadvbv":`	*String* true	Only present with value "true" if page was successfully redirected from the 3-D Secure site.
3-D Message Version `"threeDSVersion":`	*String* 3-character numeric; middle character is a decimal point	Contains the message protocol for the 3-D Secure authentication.
3DS Authentication Type `"authenticationType":`	*String* This is a snippet	3-D Secure Authentication method the issuer will use to challenge the cardholder. 01 = Static

Response Field Name and Key	Type and Limits	Description
		02 = Dynamic 03 = OOB 04 = Decoupled
3DS ACS Transaction ID `"ThreeDSACSTransID":`	*String* 36-character alpha-numeric	Universally Unique transaction identifier assigned by the issuer Access Control Server (ACS) to identify a single transaction.
3DS Authentication Time Stamp `"ThreeDSAuthTimeStamp":`	*String* 12-character numeric, format = YYYYMMDDHHMM	Date and time in UTC of the cardholder 3DS authentication.
3DS Directory Server Transaction ID `"DSTransID":`	*String* 36-character alpha-numeric	Universally unique transaction identifier assigned by the 3DS Directory Server (DS) to identify a single transaction.
Kount `"kount":{`	*Object* N/A	Contains information related to the Kount fraud tool
decision origin `"decision_origin":`	*String* alphanumeric, see description for possible values	Possible values: `Moneris` or `Merchant`
Kount result `"result":`	*String* 1-character numeric	Possible values are as follows: 1 = Success 2 = Failed 3 = Not performed 4 = Card not eligible
condition `"condition":`	*String* 1-character numeric	Indicates whether this fraud tool was set as a factor for Moneris to use when making an automatic decision on a transaction

Response Field Name and Key	Type and Limits	Description
		Possible values are as follows: 0 = Optional 1 = Mandatory
status `"status":`	*String* alphabetic, see description for possible values	Indicates whether the fraud tool inquiry was performed, and if it was used for auto-decisioning purposes Possible values: success = Fraud tool successful failed = Fraud tool failed (non-auto decision) disabled = Fraud tool not performed ineligible = Fraud tool was selected but card is not a credit card or card not eligible failed_optional = Fraud tool failed and auto decision is optional failed_mandatory = Fraud tool failed auto decision is mandatory
Kount Result Code `"code":`	*String* 1-character alphabetic	Possible values: A = Approve D = Decline R = Review E = Escalate
details `"details":{`	*Object*	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
Kount response code `"responseCode":`	*String* 3-character numeric	Final risk score returned from Kount system Possible values: 001 = Success

Response Field Name and Key	Type and Limits	Description
		973 = Unable to locate merchant Kount details 984 = Data error 987 = Invalid transaction
message (Kount) `"message":`	*String* 255-character alpha-numeric	Brief description message about the Kount inquiry
receipt ID `"receiptID":`	*String* 64-character alpha-numeric	The order ID echoed from the original financial transaction
Kount Result Code `"code":`	*String* 1-character alphabetic	Possible values: A = Approve D = Decline R = Review E = Escalate
Kount score `"score":`	*String* 3-character numeric	Final risk score returned from Kount system
Kount error `"error":`	*String* alphabetic	List of errors the Kount request generated
vault data `"vault_data":`	*Object* N/A	Object containing information related to Moneris Vault
data key `"data_key":`	*String* 25-character alpha-numeric	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile Data key is generated by Moneris and returned to you in the Receipt object when the profile is first registered

Response Field Name and Key	Type and Limits	Description
is data key valid `"is_valid":`	*String* true/false	Indicates whether the data key is valid Possible values: `true` or `false`

2.7.2.4 Example JSON Response to Receipt Request

NOTE: Not all features in Moneris Checkout are supported simultaneously, and therefore some objects would not actually appear in the response at the same time in a real-world scenario; response code below is provided for illustrative purposes only.


```
{
   "response":{
      "success":"true",
      "request":{
         "txn_total":"6.00",
         "cust_info":{
            "first_name":"bill",
            "last_name":"smith",
            "phone":"4165551234",
            "email":"test@moneris.com"
         },
         "shipping":{
            "address_1":"1 main st",
            "address_2":"Unit 2012",
            "city":"Toronto",
            "country":"Ca",
            "province":"On",
            "postal_code":"M1M1M1"
         },
         "billing":{
            "address_1":"1 main st",
            "address_2":"Unit 2000",
            "city":"Toronto",
            "province":"ON",
            "country":"CA",
            "postal_code":"M1M1M1"
```

```
},
"recur":{
   "number_of_recurs":"3",
   "recur_period":"1",
   "recur_amount":"15.00",
   "recur_unit":"month",
   "start_date":"20220902",
   "bill_now":"false"
},
"cart":{
   "items":[
      {
         "url":"https:\\/\\/esqa.moneris.com\\/cr\\/checkout\\/item3.jpg",
         "description":"Three item",
         "product_code":"two_item",
         "unit_cost":"11.00",
         "quantity":"1"
      }
   ],
   "subtotal":"10.24",
   "tax":{
      "amount":"0.00",
      "description":"Tax",
      "rate":"0"
   }
},
"cc_total":"6.00",
"cc":{
   "first6last4":"4242424242",
   "expiry":"1221",
   "cardholder":"test"
},
"mcp":{
   "merchant_settlement_amount":"452.00",
   "cardholder_currency_code":"978"
},
"gift":[
   {
      "balance_remaining":"0.00",
      "Description":"Gift Fixed Reload",
      "first4last4":"************0214",
      "pan":"0211020000001000214",
      "cvd":"123",
```

```
            "balance_used":"200.00"
         }
      ],
      "wallet":{
         "type":"applepay",
         "paymentData":{
            "token":{
               "paymentData":{
                  "data":"0YJMuivQ6+xILNQyDfwO+kzKWZ//TkNa5nxBzHIf7fw==",
                  "signature":"MIAGCSqGSIb3DJ7mrwhISSB+Ic6kAAAAAAAA=",
                  "header":{
                     "publicKeyHash":"18hkrHSrxIdbZs5qKY4US8bFqEk6bBGXde14yQrwrd8=",
                     "ephemeralPublicKey":"MFkwEwYHKoZIzj0h6ilzF+Z4dseqHDjsdYA==",
                     "transactionId":"4c0d6ae158aa0322b1f5baf6467195e0238ca48f"
                  },
                  "version":"EC_v1"
               },
               "paymentMethod":{
                  "displayName":"Discover 2780",
                  "network":"Discover",
                  "type":"credit"
               },
               "transactionIdentifier":"4C0D6AE158AA03CC4BAF6467195E0238CA48F"
            }
         }
      },
      "pay_by_token":1,
      "ticket":"1635780027iwm4IczLlO2LqHQ6xHmvDJ1xFIS2vT",
      "cust_id":"chkt- cust -1101",
      "dynamic_descriptor":"dyndesc",
      "order_no":"20211101152026",
      "eci":"7"
   },
   "receipt":{
      "result":"a",
      "gift":[
         {
            "order_no":"1583250405Ad1BmCSsfHHDeu4_g1",
            "transaction_no":"6198-1583250435590-00157838_15",
            "reference_no":"3276071",
            "response_code":"000",
            "benefit_amount":"200.00",
            "benefit_remaining":"0.00",
```

```
            "first6last4":"0211020214"
         }
      ],
      "cc":{
         "order_no":"20211101152026",
         "cust_id":"chkt- cust -1101",
         "transaction_no":"8291-0_20",
         "reference_no":"660115340017373030",
         "transaction_code":"00",
         "transaction_type":"200",
         "transaction_date_time":"2021-11-01 11:20:53",
         "corporate_card":null,
         "amount":"6.00",
         "response_code":"027",
         "iso_response_code":"01",
         "approval_code":"489642",
         "card_type":"V",
         "dynamic_descriptor":"dyndesc",
         "invoice_number":null,
         "customer_code":null,
         "eci":"7",
         "cvd_result_code":"1M",
         "avs_result_code":"null",
         "cavv_result_code":null,
         "first6last4":"4242424242",
         "expiry_date":"1221",
         "recur_success":"null",
         "issuer_id":null,
         "is_debit":null,
         "advice_code": "41",
         "ecr_no":"66011534",
         "batch_no":"737",
         "sequence_no":"303",
         "result":"a",
         "cf_success":"true",
         "cf_fee_amt":"0.11",
         "cf_fee_rate":"1.75",
         "cf_fee_type":"PCT",
         "cf_status":"2",
         "tokenize":{
            "success":"true",
            "first4last4":"2222***0011",
            "datakey":"4sbe08wFMEePj4632EVIbWNL2",
```

```
    "status":"001",
    "message":"Successfully updated CC details."
},
"mcp":{
    "merchant_settlement_amount":"452.00",
    "cardholder_currency_code":"978",
    "mcp_rate":"1.508",
    "decimal_precision":"2",
    "cardholder_amount":"299.73",
    "cardholder_currency_desc":"EUR"
},
"fraud":{
  "3d_secure": {
     "decision_origin": "Moneris",
     "result": "3",
     "condition": "1",
     "status": "success",
     "code": "5",
     "transStatus": "Y",
     "details": ""
    },
    "kount":{
       "decision_origin":"Merchant",
       "result":"3",
       "condition":null,
       "status":"disabled",
       "code":"",
       "details":""
    },
    "avs":{
       "decision_origin":"Merchant",
       "result":"3",
       "condition":"0",
       "status":"disabled",
       "code":"",
       "details":""
    },
    "cvd":{
       "decision_origin":"Merchant",
       "result":"1",
       "condition":"0",
       "status":"success",
       "code":"1M",
```

```
                "details":""
            }
        },
        "vault_data":[
            {
                "data_key":"JNDSFQKD8iRsCQM5TcBObz9V7",
                "is_valid":"true"
            },
            {
                "data_key":"4jZb8X1gCpWfC4f4KKh6Fizf2",
                "is_valid":"true"
            },
            {
                "data_key":"jjq6xx5cUQsS1AhwaQhNYpKG8",
                "is_valid":"true"
            }
        ]
      }
    }
  }
}
```


2.8 Terminating the Moneris Checkout Instance

To terminate the Moneris Checkout instance, call **myCheckout.closeCheckout()**, for example:

```
myCheckout.closeCheckout([ticket #]);
```

3 Additional Features in Moneris Checkout

3.1 Tokenization of Credentials With Moneris Checkout

You can use Moneris Checkout to store a cardholder's credentials in the Moneris Vault and receive a token that represents those credentials for use in future transactions. You can also use Moneris Checkout to update the credentials associated with the token.

If you want to tokenize credentials in Moneris Checkout transactions, you select the **Tokenize Card** option in the Merchant Resource Center. For updating existing tokens, select **Vault Update Card**.

For more information, see the Merchant Resource Center documentation available for download on the Moneris developer portal at:

developer.moneris.com

3.1.1 Tokens and Pay by Token

With tokenization enabled, Moneris Checkout also enables you to allow returning customers to select stored payment cards on the payment page. The customer's payment cards need to be already tokenized and stored in the Moneris Vault to be referenced in the Preload request and displayed to the customer when they get to the payment page.

In the Preload request, Moneris Checkout will accept a token composed of up to three pairs of **data key** and **issuer ID**, each one representing the cardholder's payment card stored in the Moneris Vault.

For more information about these fields in the Preload, see 2.4.1 Preload Request

3.2 Fraud Tools in Moneris Checkout

3.2.1 About Fraud Tools in Moneris Checkout

Several tools to mitigate the risk of fraud are available for transactions in Moneris Checkout, including:

- AVS
- CVD
- 3-D Secure
- Kount

To select which of these tools to use when performing transactions with Moneris Checkout, go to your Moneris Checkout configurator in the Moneris Merchant Resource Center under the Payment Security section.

For more information, see the Merchant Resource Center documentation available for download on the Moneris developer portal at:

developer.moneris.com

> **NOTE:** CVD is always enabled as a fraud tool and will be performed on each transaction request in Moneris Checkout, but you can choose whether Moneris will treat the CVD result as a mandatory or optional factor to approve or deny the transaction.

3.2.2 Kount as a Fraud Tool in Moneris Checkout

If you select Kount as a fraud tool in Moneris Checkout and your company has its own Enterprise service account from Kount, you will need to include your Kount Merchant ID, Kount API Key and Kount Website

ID when you configure your Moneris Checkoutstore in the Merchant Resource Center.

For Kount Enterprise users who have also enabled 3DS, Moneris sends the following as part of User Defined Fields “UDFs”. Merchants may require additional implementation of 3DS UDFs in their Kount profile or Kount risk inquiries may not utilize this data.

Kount UDFs for 3DS:

- ECI
- CAVV
- THREEDS_TRANS_STATUS
- THREEDS_TRAN_STATUS_REASON
- THREEDS_MESSAGE
- THREEDS_TRANSACTION_ID
- THREEDS_VERSION
- CHALLENGE_PERFORMED (“TRUE/FALSE”)

UDF data will only be sent to Kount if...

- 3DS `transStatus` = Y, A, U, or null*

*3DS errors such as invalid cardholder name can result in null 3DS `transStatus`

UDF data is not sent to Kount if...

- 3DS `transStatus` = N or R

For Kount Essentials merchants using the Moneris-defined fraud service package with 3DS enabled, you do not require any changes. Moneris and Kount manage the UDF data on your behalf.

3.2.3 Fraud Tools and Auto Decision-Making

Moneris Checkout can be configured to automatically proceed with or deny transactions as a result of a risk assessment it makes based on the responses it receives from the selected fraud tools.

When you check the box for auto decision-making, you also can choose whether each fraud tool's analysis will be treated by Moneris as an optional or mandatory factor in the decision to approve or deny the transaction.

This information applies to all fraud tools with the following exception:

- 3-D Secure, which is always mandatory *if* enabled.

> **NOTE:** For Google Pay transactions, 3DS authentication is only applied if the Google Pay wallet contained a funding card number (FPAN) and not applied when the wallet contains a device card number (DPAN)

3.3 Window Size in Moneris Checkout

You can customize the appearance of the Moneris Checkout window presented to the customer on their web browser, including how much of the browser window will be taken up by Moneris Checkout.

The default sizing behaviour of the Moneris Checkout window is full-screen, i.e., Moneris Checkout fills the entire web page. You can alter this behaviour to present the customer with a windowed view instead. If you do not use the full-screen option, you must define the size of the `<div>` for the windowed view. For more information, see 2.3 Preparing Your Client-Side Checkout Page.

You configure the sizing along with other aspects of the Moneris Checkout window in the Merchant Resource Center.

3.4 Multi-Currency Pricing in Moneris Checkout

You can configure Moneris Checkout to price goods and services in a variety of foreign currencies, while continuing to receive settlement and reporting in Canadian dollars.

If you want to use Multi-Currency Pricing (MCP) in Moneris Checkout transactions, you can enable the Multi-Currency Pricing option in the Merchant Resource Center. MCP is only available for Visa and Mastercard.

If Multi-Currency Pricing is enabled, the following features are not supported:

- Recurring Billing
- Gift Cards
- 3-D Secure 1.0
- Google Pay™

For more information, see the Merchant Resource Center documentation available for download on the Moneris developer portal at:

developer.moneris.com

3.5 Installments by Visa in Moneris Checkout

You can configure Moneris Checkout to display installment plan offerings to cardholders. These offers allow you to receive full funding for the transaction in a single payment, while the cardholder gains the convenience of paying their issuing bank in a series of installments.

For transactions with a minimum amount of $100, Moneris Checkout confirms eligibility on card data entry by the customer via a lookup to the Installments by Visa server. The checkout iframe displays up to three installment plan offerings for an eligible card with the monthly payment amount, the number of months, and the Annual Percentage Rate (APR) charged by the issuing bank. The cardholder can read terms and conditions for the plans and agree to them.

Installments by Visais only available for select issuers within the Visa, Mastercard, and Amex associations.

If Installments by Visa are utilized on a transaction by the customer, the following features are automatically disabled:

- Gift Cards
- Digital wallets such as Google Pay™ or Apple Pay

If you want to use Installments by Visa in Moneris Checkout transactions, you can enable theInstallments by Visa option in the Merchant Resource Center. For more information, see the Merchant Resource Center documentation available for download on the Moneris developer portal at:

developer.moneris.com

4 Testing Your Moneris Checkout Integration

In the testing stage of development:

1. Use the testing Merchant Resource Center at https://esqa.moneris.com/mpg to configure your Moneris Checkout page for testing purposes

2. Use the testing URL for server to server requests:
   `https://gatewayt.moneris.com/chktv2/request/request.php`

3. Reference the testing JavaScript library:
   ```
   <script src="https://gatewayt.moneris.com/chktv2/js/chkt_
   v2.00.js"></script>
   ```

4. Set your **myCheckout** object to the testing mode:
   `myCheckout.setMode("qa");`

5. In all Preload requests use the value "`qa`" for the **environment** variable

6. In all Preload requests, make sure that you are using the testing version of your credentials for **store ID**, **API token** and **checkout ID**

7. In all Receipt requests use the value "`qa`" for the **environment** variable

8. In all Receipt requests, make sure that you are using the testing version of your credentials for **store ID**, **API token** and **checkout ID**

4.1 Test Cards for Moneris Checkout

Test card numbers are available for testing your Moneris Checkout integration. For the most current test card information, see the Moneris developer portal at:

https://developer.moneris.com/en/More/Testing/Testing%20a%20Solution

Special information for testing convenience fee/service fee:

For testing transactions with convenience fee/service fee in Moneris Checkout, you must use the specific test credentials:

Store ID: monca00392

API token: qYdISUhHiOdfTr1CLNpN

Username: DemoUser

Password: password

5 Moving to Production with Moneris Checkout

Once you have finished testing your Moneris Checkout integration, do the following to move the integration into production:

1. Ensure that you have duplicated your final testing configuration in your Moneris Checkout production configuration in the production Merchant Resource Center at https://esqa.moneris.com/mpg to configure your Moneris Checkout page for testing purposes

2. Use the production URL for server to server requests:
   `https://gateway.moneris.com/chktv2/request/request.php`

3. Reference the production JavaScript library:
   ```
   <script src="https://gateway.moneris.com/chktv2/js/chkt_
   v2.00.js"></script>
   ```

4. Set your **myCheckout** object to the production mode:
   `myCheckout.setMode("prod");`

5. In all Preload requests use the value "`prod`" for the **environment** variable

6. In all Preload requests, make sure that you are using the production version of your credentials for **store ID**, **API token** and **checkout ID**

7. In all Receipt requests use the value "`prod`" for the **environment** variable

8. In all Receipt requests, make sure that you are using the production version of your credentials for **store ID**, **API token** and **checkout ID**

6 Reference

6.1 Callback Response Codes – Moneris Checkout

Response Code	Reason
001	Success
902	3-D Secure failed on response
2001	Invalid ticket
2002	Ticket re-use
2003	Ticket expired
2004	Network request on initial page load failed

6.2 AVS Response Codes – Moneris Checkout

Code	Visa	Mastercard	Discover	American Express/ JCB
A	Partial match. Street address matches, zip/postal code does not; acquirer rights not implied	Partial match. Address matches, zip/ postal code does not	Full match. Address matches, five-digit postal code matches	Partial match. Billing address matches, zip/-postal code does not
D	N/A	N/A	N/A	Partial match. Customer name incorrect; zip/postal code matches

Code	Visa	Mastercard	Discover	American Express/ JCB
E	N/A	N/A	N/A	Partial match. Customer name incorrect, billing address and zip/-postal code match
F	N/A	N/A	N/A	Partial match. Customer name incorrect; billing address matches
G	N/A	N/A	Unavailable. Address information not verified for international transaction	N/A
K	N/A	N/A	N/A	Partial match. Customer name matches
L	N/A	N/A	N/A	Partial match. Customer name and zip/postal code match
M	N/A	N/A	N/A	Full match. Customer name, billing address, and zip/postal code match
N	No match; acquirer sent: • postal/ZIP code only, or • street address only, or • both postal	No match. Neither address nor zip/postal code matches	No match. Neither address nor zip/postal code matches	No match. Billing address and zip/-postal code do not match

Code	Visa	Mastercard	Discover	American Express/ JCB
	code and street address Also used when acquirer requests AVS but sends no AVS data			
O	N/A	N/A	N/A	Customer name and billing address match
R	Unavailable. System unavailable or timed out Issuer ordinarily performs AVS, but was unavailable **NOTE:** Code R is used by Visa when issuers are unavailable; issuers should refrain from using this code.	Unavailable. System unable to process	N/A	Unavailable. System unavailable; retry
S	N/A	Unavailable. AVS currently not supported	Unavailable. AVS currently not supported	Unavailable. Service Establishment did not allow address verification function.
T	N/A	N/A	Partial match. Nine-digit zip code matches, address does not match.	N/A
U	Unavailable. Address not verified for domestic transaction, for any of the following reasons:	Unavailable. No data from issuer-/authorization system	Unavailable. Retry; system unable to process.	Unavailable.. Information is unavailable

Code	Visa	Mastercard	Discover	American Express/ JCB
	• Issuer is not an AVS participant, or • AVS data was present in the request but issuer did not return an AVS result, or • Visa performs AVS on behalf of the issuer and there was no address record on file for this account			
W	N/A	Partial match. For U.S. addresses, nine-digit postal code matches, address does not For addresses outside the U.S., postal code matches, address does not	Unavailable. Retry; system unable to process	No match. Customer name, billing address, and zip/postal code are all correct matches
X	N/A	Full match. For U.S. addresses, nine-digit postal code and address match For addresses outside the U.S., postal code and address match	Full match. Nine-digit postal code and address match	N/A
Y	Full match. Street address and zip/-postal code match	Full match. Billing address and zip/postal code both match	Partial match. Billing address	Full match. Billing address and zip/-

Code	Visa	Mastercard	Discover	American Express/ JCB
			matches but zip/-postal code does not.	postal code both match
Z	Partial match. Zip/-postal code matches; street address does not match, or street address not included in request	Partial match. For U.S. addresses, five-digit zip code matches, address does not match	Partial match. t zip/postal code matches but billing address does not.	Partial match. Zip/-postal code matches, billing address does not

6.3 CVD Response Codes – Moneris Checkout

CVD verification is available for Visa, Mastercard, Discover, American Express, JCB and UnionPay transactions.

Code	Description
M	Match
N	No match
P	Not processed
S	CVD should be on the card, but Merchant has indicated that CVD is not present
U	Issuer is not a CVD participant
Y	Match for American Express/JCB only
D	Invalid security code for American Express or JCB only
Other	Invalid response code

6.4 CAVV Result Codes

The Cardholder Authentication Verification Value (CAVV), the Accountholder Authentication Value (AAV), and the American Express Verification Value (AEVV), are the values that allows Visa, Mastercard

and American Express to validate the integrity of the Visa Secure, Mastercard Identity Check and American Express SafeKey transaction data. These values are passed back from the issuer to the merchant after the authentication has taken place.

The merchant then integrates the CAVV/AAV/AEVV value into the authorization request using the Purchase with 3-D Secure or Pre-Authorization with 3-D Secure transaction type, described below:

1. Merchant conducts 3D-Secure authentication request and receives CAVV/AAV/AEVV value in response.

2. Merchant sends the CAVV/AAV/AEVV value to Moneris using the Purchase or Pre-Authorization with 3-D Secure transaction type and receives the CAVV result code in the response.

Visa CAVV result codes

Result Code	Message	Significance to Merchants
Blank	CAVV not present or not verified	Not a Visa Secure transaction. No liability shift and merchant is not protected from chargebacks
0	CAVV authentication results invalid	Not a Visa Secure transaction. No liability shift and merchant is not protected from chargebacks
1	CAVV failed validation (authentication)	Provided that you have implemented the Visa Secureprocess correctly, the liability for this transaction should remain with the Issuer for chargeback reason codes covered by Visa Secure.
2	CAVV passed validation (authentication)	Fully authenticated transaction. There is a liability shift and the merchant is protected from chargebacks.
3, 8, A	CAVV passed validation (attempt)	Visa Secure has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks.
4, 7, 9	CAVV failed validation (attempt)	Visa Secure has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks.
6	CAVV not validated - Issuer not participating	Visa Secure has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks.

Result Code	Message	Significance to Merchants
B	CAVV passed validation; information only	Not a Visa Secure transaction. No liability shift and merchant is not protected from chargebacks
C	CAVV was not validated (attempt)	Visa Secure has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks.
D	CAVV was not validated (authentication)	Visa Secure has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks.

Mastercard CAVV result codes

Result Code	Message	Significance to Merchants
0	Authentication failed	Not a Mastercard Identity Check transaction. No liability shift and merchant is not protected from chargebacks
1	Authentication attempted	Mastercard Identity Check has been attempted. There is a liability shift and the merchant is protected from certain card fraud-related chargebacks (international commercial cards excluded).
2	Authentication successful	Fully authenticated transaction. There is a liability shift and the merchant is protected from chargebacks.

American Express CAVV result codes

NOTE: American Express SafeKey is only available to American Express direct acquired merchants (i.e., not OptBlue merchants). Any questions pertaining to chargebacks, liability and disputes should be addressed to your American Express representative given that American Express is the acquirer of record for these merchants.

Result Code	Description
1	AEVV Failed - Authentication, Issuer Key

Result Code	Description
2	AEVV Passed - Authentication, Issuer Key
3	AEVV Passed - Attempt, Issuer Key
4	AEVV Failed - Attempt, Issuer Key
7	AEVV Failed - Attempt, Issuer not participating, Network Key
8	AEVV Passed - Attempt, Issuer not participating, Network Key
9	AEVV Failed - Attempt, Participating, Access Control Server (ACS) not available, Network Key
A	AEVV Passed - Attempt, Participating, Access Control Server (ACS) not available, Network Key
U	AEVV Unchecked