BE PAYMENT READY

Moneris Checkout Integration Guide

Version: 1.0.6

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 v1.0.6

- 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

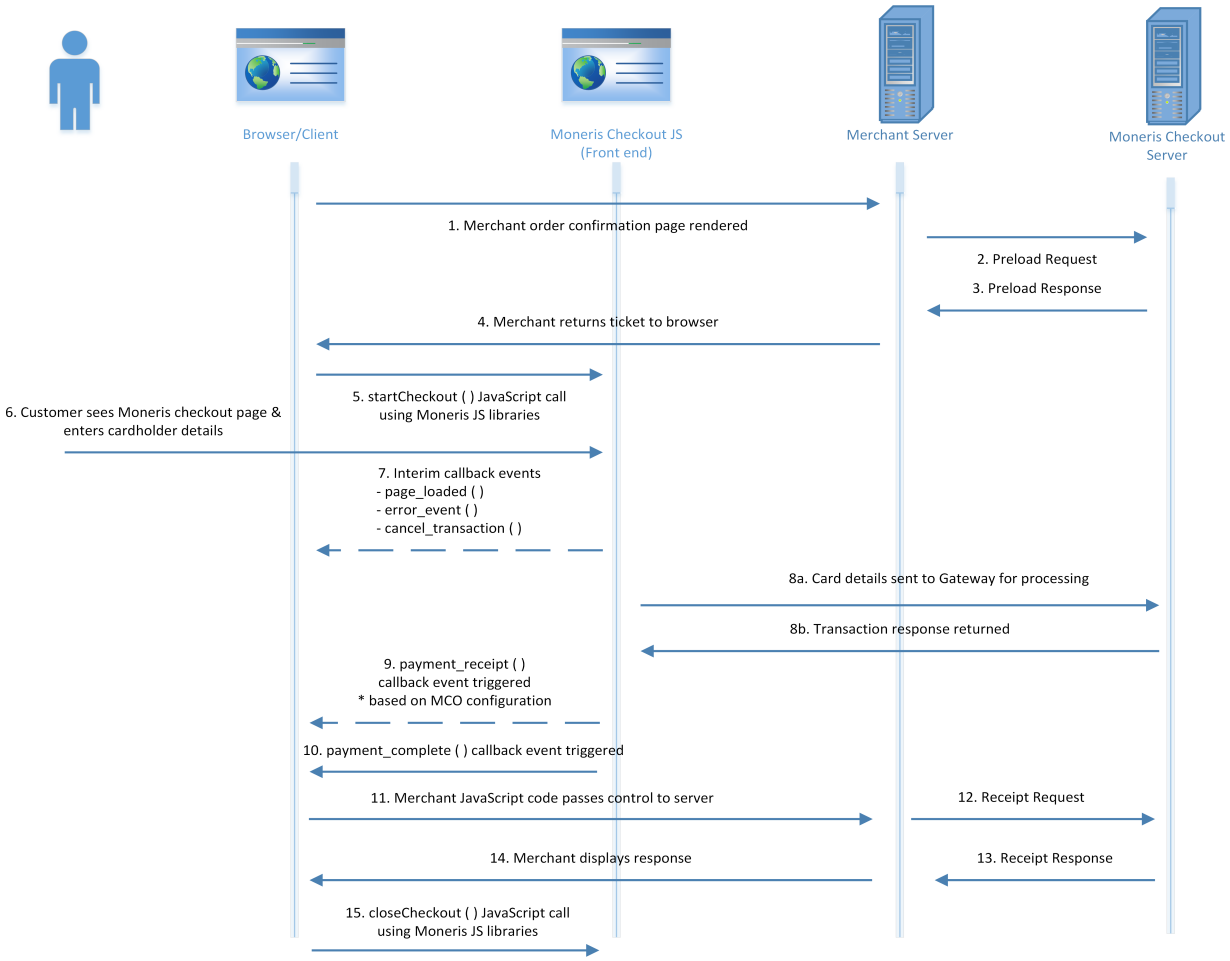
2.1.1 Additional Features to Configure in the MRC

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/chkt/js/chkt_
   v1.00.js"></script>
   ```

   Production:

   ```
   <script src="https://gateway.moneris.com/chkt/js/chkt_v1.00.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("error_event",myErrorEvent);

   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/chkt/request/request.php

Production:

https://gateway.moneris.com/chkt/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
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	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* 45-character alphanumeric	The order number is a unique identifier appended to every financial transaction

Variable Name	Type and Limits	Description
	NOTE: Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	
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
language `language`	*String* 2-character alphabetic	Determines which language Moneris Checkout will display information in Allowable values: en – English fr – French

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

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.

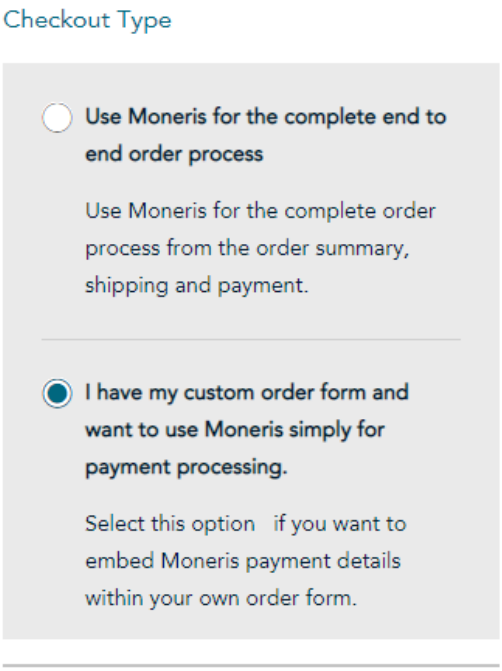
If you have configured Moneris Checkout to handle these additional items, you do not send the corresponding object in the Preload request. Only send these optional objects if you are using your own e-commerce page to collect them separately from Moneris Checkout.

Optional objects you can use include:

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

- Shipping Details Object
- Billing Details Object

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	*String*	The number of times that the trans-

Variable Name	Type and Limits	Description
number_of_recurs	numeric 1-999	action 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 to define the billing frequency
start date start_date	*String* YYYYMMDD 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-numeric	Describes the item in the shopping cart
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

Variable Name	Type and Limits	Description
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 **`tax.amount`**	*String* 10-character decimal Up to 7 digits (dollars) + decimal point (.) + 2 digits (cents) after the decimal point **EXAMPLE:** 1234567.89	Dollar amount of taxes
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 `last_name`	*String* 30-character alphanumeric	Customer last name
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

Variable Name	Type and Limits	Description
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:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer shipping address city
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

Top level 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 **NOTE:** Some special characters are not allowed: < > $ % = ? ^ " { } [ ] \	Customer billing address
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

2.4.1.2 Example Preload Request JSON

This example reflects a Preload request with all optional objects.

```
{
   "store_id":"moneris",
   "api_token":"hurgle",
   "checkout_id":"chkt5BF66neris",
   "txn_total":"452.00",
   "environment":"qa",
   "action":"preload",
   "order_no":"",
   "cust_id":"chkt - cust - 0303",
   "dynamic_descriptor":"dyndesc",
   "language":"en",
   "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
- Error Event
- 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 error_event 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 Error Event

Callback Use

When an error occurs during the checkout process. This requires the Moneris Checkout session to be closed using the `closeCheckout` function

To attempt the transaction again, a new Preload request must be sent to the Moneris Checkout server in order to get a new transaction ticket number.

JavaScript Set Method for Callback

```
myCheckout.setCallback("error_event",myErrorEvent);
```

JSON Response Message Format

```
{
   "handler":"error_event",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "response_code":"902"
}
```

2.6.1.5 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"
}
```

2.6.1.6 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"
}
```

2.6.1.7 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"
}
```

2.6.1.8 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/chkt/request/request.php

Production:

https://gateway.moneris.com/chkt/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	*String*	The unique ticket number that identifies a particular transaction; this

Variable Name	Type and Limits	Description
`ticket`	maximum 50-character alphanumeric	returned in the response to the Pre-load 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	Type of request being made to Moneris Checkout server Allowable values: `preload` or `receipt`

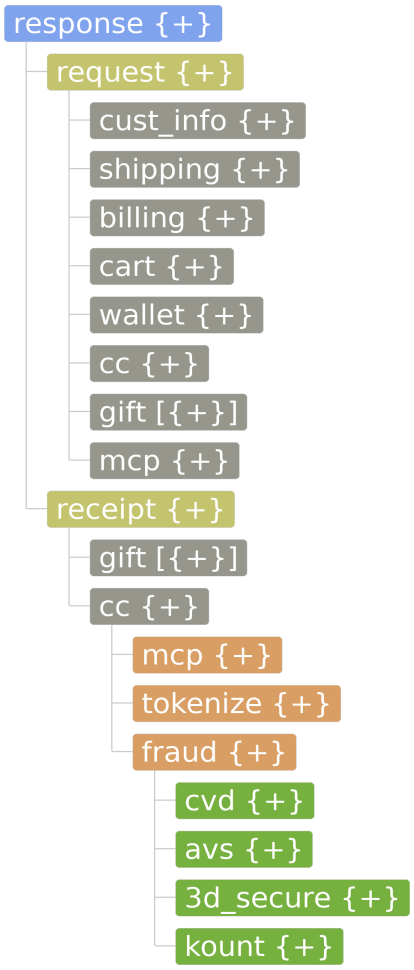
2.7.1.1 Example Receipt Request JSON

```
{
   "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



2.7.2.1 Definition of Response Fields – Response to Receipt Request

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

Response Field Name and Key	Description
response `{"response":{`	Top level response object
success `"success":`	Denotes whether request was successful (i.e., approved, declined) Unsuccessful means error or could not process Possible values: `true` or `false`
request	Contains information relating to the Preload

Response Field Name and Key	Description
`"request":{`	request and other information that Moneris Checkout sends to the Moneris Gateway when processing the financial transaction
transaction total amount `"txn_total":`	The total dollar amount of the transaction
Customer Information `"cust_info":{`	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":`	Customer first name
last name `"last_name":`	Customer last name
phone number `"phone":`	Customer phone number
email `"email":`	Customer email
Shipping `"shipping":{`	Contains customer shipping information The information presented in this response object will reflect one of three scenarios:

Response Field Name and Key	Description
	• 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":`	Customer shipping address
shipping address line 2 `"address_2":`	Customer shipping address
shipping city `"city":`	Customer shipping address city
shipping country `"country":`	Customer shipping address country ISO 3166-1 alpha-2
shipping province `"province":`	Customer shipping address province Country subdivision ISO 3166-2
shipping postal code `"postal_code":`	Customer shipping address postal code
Billing `"billing":{`	Contains customer 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

Response Field Name and Key	Description
	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":`	Customer billing address
billing address line 2 `"address_2":`	Customer billing address
billing city `"city":`	Customer billing address city
billing country `"country":`	Customer billing address country ISO 3166-1 alpha-2
billing province `"province":`	Customer billing address province Country subdivision ISO 3166-2
billing postal code `"postal_code":`	Customer billing address postal code
same as shipping `"same_as_shipping":`	Indicates whether the shipping address is the same as the billing address Possible values: `true` or `false`
Recurring Billing `"recur":{`	Contains fields related to Recurring Billing
number of recurs `"number_of_recurs":`	The number of times that the transaction must recur

Response Field Name and Key	Description
period `"recur_period":`	Number of recur unit intervals that must pass between recurring billings
recurring amount `"recur_amount":`	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":`	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":`	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":`	Set to true if a charge will be made against the card immediately; otherwise set to false
Shopping Cart `"cart":{`	The virtual shopping cart and its contents This echos the information contained in the Shopping Cart request object
shopping cart items `"items":[{`	Encapsulates the entire array of items in the shopping cart
item URL `"url":`	URL that corresponds to the image of the Moneris Checkout shopping cart item
item description `"description":`	Describes the item in the shopping cart
item product code `"product_code":`	The SKU for the item
item unit cost `"unit_cost":`	Per-unit cost of the item

Response Field Name and Key	Description
item quantity `"quantity":`	Number of individual instances of the given item in the shopping cart
subtotal `"subtotal":`	Total dollar amount of the shopping cart, before taxes
tax `"tax":`	Contains information related to taxes charged on the items in the shopping cart
tax amount `"amount":`	Dollar amount of taxes
tax description `"description":`	Describes type of tax being applied
tax rate `"rate":`	Percentage tax rate charged
credit card total `"cc_total":`	Total amount being charged to the credit card
Credit Card (request) `"cc":{`	Contains cardholder information
first 6 last 4 `"first6last4":`	First 6 and last 4 digits of card number
expiry date `"expiry":`	Card expiry date
cardholder `"cardholder":`	Cardholder name
Multi-Currency Pricing in the Preload `"mcp":{`	Contains fields related to Multi-Currency Pricing sent in the transaction
merchant settlement amount `"merchant_settlement_amount":`	The amount the merchant will receive in the transaction, in Canadian dollars
cardholder currency code	ISO code representing the foreign currency

Response Field Name and Key	Description
"cardholder_currency_code":	of the cardholder
Gift (request) "gift":[{	Object containing information about a gift card
balance remaining "balance_remaining":	The remaining balance on the gift card
gift card description "description":	Description of the gift card used for the transaction
first 4 last 4 "first4last4":	The first 4 and last 4 digits of the card
gift card number "pan":	The account number of the gift card
gift card CVD "cvd":	Card validation digits on the gift card
balance used "balance_used":	The amount that was removed from the card's balance as part of the transaction
Wallet "wallet":	Contains information from the digital wallet that was used in the transaction
wallet type "type":	Type of digital wallet used in this transaction Possible values: applepay or googlepay
payment data "paymentData":{	Object containing various information related to the payment sent from the digital wallet
API version (minor) "apiVersionMinor":	Minor version of the API
API version "apiVersion":	Version of the digital wallet's payment API
payment method data	Object containing information about the payment method used in the transaction

Response Field Name and Key	Description
`"paymentMethodData":{`	
payment method description `"description":`	User-facing message to describe the payment method that funds this transaction
tokenization data `"tokenizationData":{`	Object containing information related to tokenization and the digital wallet
tokenization type `"type":`	The type of tokenization to be applied to the selected payment method Possible values: `PAYMENT_GATEWAY` or `DIRECT`
token `"token":`	The generated payment method token
payment method type `"type":`	A short identifier for the supported payment method Possible values: CARD PAYPAL
info `"info":{`	Object that echoes information about the cardholder, the card and the card network from the digital wallet
card network `"cardNetwork":`	The payment card network
card details `"cardDetails":`	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 that echoes the cardholder's billing information from the digital wallet
address 3 `"address3":`	Third line of the address
sorting code `"sortingCode":`	The sorting code

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

Response Field Name and Key	Description
postal code `"postalCode":`	Address postal code or ZIP
name `"name":`	Name of the addressee
locality `"locality":`	City, town, neighbourhood, or suburb
administrative area `"administrativeArea":`	A country subdivision, such as a state or province
ticket number `"ticket":`	The unique ticket number that identifies a particular transaction; this returned in the response to the Preload request
customer ID `"cust_id":`	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center
dynamic descriptor `"dynamic descriptor":`	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":`	The order number is a unique identifier appended to every financial transaction
electronic commerce indicator `"eci":`	The e-commerce indicator or crypt type that was used to process the transaction Possible values are: 5 - Authenticated e-commerce transaction (3-D

Response Field Name and Key	Description
	Secure) 6 - Non-authenticated e-commerce transaction (3-D Secure) 7 - SSL-enabled merchant
Receipt `"receipt":{`	Object containing the receipt information
result (financial transaction) `"result":`	Indicates the result of the financial transaction Possible values are: a = Accepted d = Declined
Gift (receipt) `"gift":[{`	Contains information related to gift card
order number `"order_no":`	The order number is a unique identifier appended to every financial transaction
transaction number `"transaction_no":`	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":`	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** Example: 660123450010690030

Response Field Name and Key	Description
	66012345: Terminal ID 001: Shift number 069: Batch number 003: Transaction number within the batch.
response code `"response_code":`	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
benefit amount `"benefit_amount":`	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":`	The remaining balance on the gift card
first 6 last 4 `"first6last4":`	First 6 and last 4 digits of card number
Credit Card (receipt) `"cc":{`	Contains fields describing the response to the credit card transaction
order number `"order_no":`	The order number is a unique identifier appended to every financial transaction
customer ID `"cust_id":`	Merchant-defined field that can be used as an identifier Searchable from the Moneris Merchant Resource Center
transaction number `"transaction_no":`	Used to reference the original transaction when performing a follow-on transaction (i.e., Pre-Authorization Completion,

Response Field Name and Key	Description
	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":`	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** Example: 660123450010690030 66012345: Terminal ID 001: Shift number 069: Batch number 003: Transaction number within the batch.
transaction code `"transaction_code":`	Type of financial transaction that was performed Possible values: 00 – Purchase 01 – Pre-Authorization
transaction type `"transaction_type":`	ISO transaction code for financial transaction
transaction date and time `"transaction_date_time":`	Processing host date and time stamp Format: YYYY-MM-DD HH:MM:SS

Response Field Name and Key	Description
corporate card `"corporateCard":`	Indicates whether the payment card is a corporate card
credit card amount `"amount":`	The total dollar amount that was charged to the credit card
response code `"response_code":`	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":`	ISO response code returned from issuing institution For more details on specific ISO codes returned, see the Response Codes reference topic
approval code `"approval_code":`	Authorization code returned from the issuing institution
card type `"card_type":`	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 D = INTERAC® Debit C1 = JCB

Response Field Name and Key	Description
wallet type `"wallet_type":`	Type of digital wallet used in this transaction Possible values: `applepay` or `googlepay`
dynamic descriptor `"dynamic_descriptor":`	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
invoice number `"invoice_number":`	Identifies an invoice number associated with the transaction
customer code `"customer_code":`	User-defined identifier
electronic commerce indicator `"eci":`	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":`	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 Codes reference
AVS result code `"avs_result_code":`	Indicates the address verification result For a full list of possible response codes refer to the AVS Response Codes reference

Response Field Name and Key	Description
CAVV result code `"cavv_result_code":`	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":`	First 6 and last 4 digits of card number
expiry date `"expiry_date":`	Expiry date of the card MMYY format
recur success `"recur_success":`	Indicates whether the recurring billing transaction has been successfully set up for future billing Possible values: `true` or `false`
issuer ID `"issuer_id":`	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
is debit `"is_debit":`	Indicates whether a debit card was used in the transaction
ECR (electronic cash register) number `"ecr_no":`	Terminal ID/ECR Number from the request

Response Field Name and Key	Description
batch number `"batch_no":`	Batch number; also presented as a component of the reference number
sequence number `"sequence_no":`	Transaction number within the batch; also presented as a component of reference number
result (financial transaction) `"result":`	Indicates the result of the financial transaction Possible values are: a = Accepted d = Declined
Tokenize `"tokenize":{`	Contains information related to the tokenization of cardholder credentials
success (tokenize) `"success":`	Indicates whether the card was successfully tokenized Possible values: `true` or `false`
first 4 last 4 `"first4last4":`	The first 4 and last 4 digits of the card
data key `"data_key":`	Unique identifier for a Vault profile, and used in future Vault financial transactions to associate a transaction with that profile
tokenization status `"status":`	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)

Response Field Name and Key	Description
tokenization message `"message":`	Provides additional details about the success or failure of the tokenization
Multi-Currency Pricing in the Response `"mcp":{`	Contains fields related to Multi-Currency Pricing received in the response
merchant settlement amount `"merchant_settlement_amount":`	The amount the merchant will receive in the transaction, in Canadian dollars
cardholder currency code `"cardholder_currency_code":`	ISO code representing the foreign currency of the cardholder
multi-currency pricing rate `"mcp_rate":`	The foreign exchange rate (foreign currency to CAD) that was used for the transaction
decimal precision `"decimal_precision":`	Decimal precision of the amount
cardholder amount `"cardholder_amount":`	Amount, in units of foreign currency, the cardholder will be charged on the transaction
cardholder currency description `"cardholder_currency_desc":`	Describes the foreign currency being used in the transaction
Fraud `"fraud":{`	Contains sub-objects that describe information related to fraud tool inquiries
CVD `"cvd":{`	Contains information related to the CVD fraud tool
decision origin `"decision_origin":`	Possible values: `Moneris` or `Merchant`
CVD result `"result":`	Possible values: 1 = Success 2 = Failed 3 = Not performed

Response Field Name and Key	Description
	4 = Card not eligible
condition `"condition":`	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":`	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":`	CVD result code; for a list of possible codes see the CVD Response Codes reference
condition `"details":`	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
AVS `"avs":{`	Contains information related to the AVS fraud tool
decision origin `"decision_origin":`	Possible values: `Moneris` or `Merchant`

Response Field Name and Key	Description
AVS result `"result":`	Possible values: 1 = Success 2 = Failed 3 = Not performed 4 = Card not eligible
condition `"condition":`	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":`	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
AVS code `"code":`	AVS result code; for a list of potential codes, see the AVS Response Codes reference
details `"details":`	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
3-D Secure `"3d_secure":{`	Contains information related to the 3-D Secure fraud tool
decision origin	Possible values: `Moneris` or `Merchant`

Response Field Name and Key	Description
`"decision_origin":`	
3-D Secure result `"result":`	Possible values: 1 = Success 2 = Failed 3 = Not performed 4 = Card not eligible
condition `"condition":`	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":`	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
3-D Secure code `"code":`	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

Response Field Name and Key	Description
details `"details":{`	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
Cardholder Authentication Value (CAVV) `"cavv":`	Value provided by the Moneris MPI or by a third-party MPI Returned by Visa Secure, Mastercard Identity Check or American Express SafeKey transactions
3-D Secure message `"message":`	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":`	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 Secure 1.0 transactions
PARes `"PARes":`	Payer authentication response code Possible values: true = Fully authenticated or attempted to verify PIN false = Failed to authenticate **NOTE:** Only returned for 3-D Secure 1.0 transactions

Response Field Name and Key	Description
transaction status `"transStatus":`	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 Secure 2.0
load 3-D Secure `"loadvbv":`	Only present with value "true" if page was successfully redirected from the 3-D Secure site.
Kount `"kount":{`	Contains information related to the Kount fraud tool
decision origin `"decision_origin":`	Possible values: `Moneris` or `Merchant`
Kount result `"result":`	Possible values: 001 = Success 973 = Unable to locate merchant Kount details 984 = Data error 987 = Invalid transaction
condition `"condition":`	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

Response Field Name and Key	Description
status `"status":`	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 code `"code":`	Possible values: 001 = Success 973 = Unable to locate merchant Kount details 984 = Data error 987 = Invalid transaction
details `"details":{`	Provides detailed information about the fraud tool query Only populated for Kount and 3-D Secure
Kount response code `"responseCode":`	Final risk score returned from Kount system
message (Kount) `"message":`	Brief description message about the Kount inquiry
receipt ID `"receiptID":`	The order ID echoed from the original financial transaction
Kount result `"result":`	Possible values are as follows: 1 = Success 2 = Failed 3 = Not performed

Response Field Name and Key	Description
	4 = Card not eligible
Kount score "score":	Final risk score returned from Kount system
Kount error "error":	List of errors the Kount request generated

2.7.2.2 Example JSON Response to Receipt Request

```
"response":{
   "success":"true",
   "request":{
      "txn_total":"452.00",
      "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"
         }
      },
      "cust_info":{
         "first_name":"bill",
         "last_name":"smith",
         "phone":"4165551234",
         "email":"test@moneris.com"
      },
      "shipping":{
         "address_1":"1 main st ",
         "address_2":"Unit 2000",
         "city":"Toronto",
         "country":"CA",
         "province":"ON",
         "postal_code":"M1M1M1"
      },
      "billing":{
         "address_1":"1 main st ",
         "address_2":"Unit 2000",
         "city":"Toronto",
         "country":"CA",
         "province":"ON",
         "postal_code":"M1M1M1"
      },
      "cc_total":"252.00",
      "gift":[
         {
            "balance_remaining":"0.00",
            "Description":"Gift Fixed Reload",
            "first4last4":"************0214",
            "pan":"0211020000001000214",
            "cvd":"123",
            "balance_used":"200.00"
         }
      ],
      "cc":{
         "first6last4":"4761735637",
         "expiry":"0121",
         "cardholder":"bill smith"
      },
```

```
    "mcp":{
       "merchant_settlement_amount":"452.00",
       "cardholder_currency_code":"978"
    },
    "ticket":"157556871510pAi0PJRFZaOISm4DHQvigFswDnET ",
    "cust_id":" chkt - cust - 0303",
    "dynamic_descriptor":"dyndesc",
    "order_no":null,
    "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":"1572443908zc9cIHRI6aJTwZU",
       "cust_id":null,
       "transaction_no":"5246-0_14",
       "reference_no":"660187030012750100",
       "transaction_code":"00",
       "transaction_type":"200",
       "transaction_date_time":"2019-10-30 09:59:14",
       "corporateCard":"false",
       "amount":"252.00",
       "response_code":"027",
       "iso_response_code":"01",
       "approval_code":"851268",
       "card_type":"V",
       "dynamic_descriptor":"dyndesc",
       "invoice_number":null,
       "customer_code":null,
       "eci":"7",
       "cvd_result_code":"1M",
```

```
"avs_result_code":"M",
"cavv_result_code":"2",
"first6last4":"4761735637",
"expiry_date":"0121",
"recur_success":null,
"issuer_id":"570106735143835",
"is_debit":"true",
"ecr_no":"66018703",
"batch_no":"275",
"sequence_no":"010",
"result":"a",
"tokenize":{
   "success":"true",
   "firstlast4":"4761***5637",
   "datakey":"lRNJbQUpcsw4qRkd3LpZ5Lpi7",
   "status":"001",
   "message":"Successfully registered 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":{
   "cvd":{
      "decision_origin":"Merchant",
      "result":"1",
      "condition":"1",
      "status":"success",
      "code":"1M",
      "details":""
   },
   "avs":{
      "decision_origin":"Merchant",
      "result":"1",
      "condition":"1",
      "status":"success",
      "code":"M",
      "details":""
```

```
            },
            "3d_secure":{
               "decision_origin":"Moneris",
               "result":"1",
               "condition":"1",
               "status":"success",
               "code":"5",
               "details":{
                  "VERes":"Y",
                  // Only returned for 3DS 1.0 transactions"PARes":"true",
                  // Only returned for 3DS 1.0 transactions"transStatus":"Y",
                  // Only returned for 3DS 2.0 transactions"message":"Successful Payer Authentic-
ation",
                  "cavv":"AAACADlzFBchc3F5NHMUEwAAAAA=",
                  "loadvbv":true,
                  "eci":"5"
               }
            },
            "kount":{
               "decision_origin":"Merchant",
               "result":"2",
               "condition":"0",
               "status":"failed_optional",
               "code":"001",
               "details":{
                  "responseCode":"001",
                  "message":"Success",
                  "receiptID":"1567780065NjErgHzsUubOkoIA31gaSUtLkxKseT",
                  "result":"D",
                  "error":[
                     "DECLINE when Device Country is not the same as the BIN Country for VISA and
MC &amp;amp; Data Collector is present",
                     "DECLINE when the BIN country is not the same as the Billing Country for
VISA and MC"
                  ]
               }
            }
         }
      }
   }
}
}
```

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 configure 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.

If you want to tokenize credentials in Moneris Checkout transactions, you select the **Tokenize Card** 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

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.

If you are using Moneris's basic fraud service package and do not have your own Kount enterprise account, you do not require this information.

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

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 window ed 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

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/chkt/request/request.php`

3. Reference the testing JavaScript library:
   ```
   <script src="https://gatewayt.moneris.com/chkt/js/chkt_
   v1.00.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**

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/chkt/request/request.php`

3. Reference the production JavaScript library:
   `<script src="https://gateway.moneris.com/chkt/js/chkt_v1.00.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

6.2 AVS Response Codes – Moneris Checkout

Code	Visa	Mastercard/Discover	American Express/ JCB
A	Street address matches, zip/postal code does not; acquirer rights not implied	Address matches, zip/ postal code does not	Billing address matches, zip/postal code does not
B	Street address matches; zip/postal code not verified due to incompatible formats (acquirer sent both street address and zip/postal code)	N/A	N/A
C	Street address not verified due to incompatible formats	N/A	N/A

Code	Visa	Mastercard/Discover	American Express/ JCB
	(acquirer sent both street address and zip/postal code)		
D	Street address and zip/-postal code match	N/A	Customer name incorrect; zip/postal code matches
E	N/A	N/A	Customer name incorrect, billing address and zip/-postal code match
F	*Applies to UK only*: Street address and zip/postal code match	N/A	Customer name incorrect; billing address matches
G	Address information not verified for international transaction Any of following may be true: • 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	N/A	N/A
I	Address information not verified	N/A	N/A
K	N/A	N/A	Customer name matches

Code	Visa	Mastercard/Discover	American Express/ JCB
L	N/A	N/A	Customer name and zip/-postal code match
M	Street address and zip/-postal code match	N/A	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 code and street address Also used when acquirer requests AVS but sends no AVS data	Neither address nor zip/-postal code matches	Billing address and zip/-postal code do not match
O	N/A	N/A	Customer name and billing address match
P	Zip/postal code match; acquirer sent both zip/-postal code and street address, but street address not verified due to incompatible formats	N/A	N/A
R	Retry; 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.	Retry; system unable to process	System unavailable; retry
S	N/A	AVS currently not supported	AVS currently not supported

Code	Visa	Mastercard/Discover	American Express/ JCB
T	N/A	Nine-digit zip code matches; address does not match	N/A
U	Address not verified for domestic transaction, for any of the following reasons: • 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	No data from issuer-/authorization system	Information is unavailable
W	Not applicable; if present, replaced with Z by Visa *Available for U.S. issuers only*	For U.S. addresses, nine-digit postal code matches, address does not For addresses outside the U.S., postal code matches, address does not	Customer name, billing address, and zip/postal code are all correct matches
X	N/A	For U.S. addresses, nine-digit postal code and address match For addresses outside the U.S., postal code and address match	N/A
Y	Street address and zip/-postal code match	Billing address and zip/-postal code both match	Billing address and zip/-postal code both match

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

Result Code	Description
U	AEVV Unchecked