Moneris™

BE PAYMENT READY

Moneris Checkout Integration Guide

Version: 1.0.0

Review Draft

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 Hours: Monday – Friday, 8:30am to 8 pm ET	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 Hours: 8am to 8pm ET	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 separate back-end server with HTTPS certificates for communicating with the Moneris Gateway via JSON

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

- JavaScript
- JSON
- Server-side programming

1 About Moneris Checkout

Moneris Checkout gives merchants conducting e-commerce transactions a secure way to process transactions by integrating a Moneris-hosted checkout page into your merchant e-commerce website. Moneris Checkout uses server-to-server communications on the back end with your merchant server that identifies each transaction with a unique ticket number, without exposing this information in the browser.

2 Building Your Moneris Checkout Integration

2.1 Integrating Your Website With Moneris Checkout

Integrating your e-commerce website with the Moneris Checkout solution involves a few steps:

1. Configuring the options for your Moneris Checkout page in the Merchant Resource Center
2. Customizing your website's code to work with Moneris Checkout
3. Implementing back-end server logic for requests and responses to and from the Moneris Check-outserver

2.2 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 will be using the testing configuration, but later, when you move to production, you will also need to set up a configuration in the production MRC that reflects the one you used for testing.

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

2.3 Implementing Server Logic for Moneris Checkout

In addition to your front-end e-commerce website, you also need a back-end server to make a server-to-server post to the Moneris Checkout server in order to process a transaction.

You can code in any language you choose, but your server must send and receive messages to the Moneris Checkout server in JSON format.

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

On your website, make sure to include the corresponding JavaScript calls:

Testing:

```
<script src="https://gatewayt.moneris.com/chkt/js/chkt_v1.00.js"></script>
```

Production:

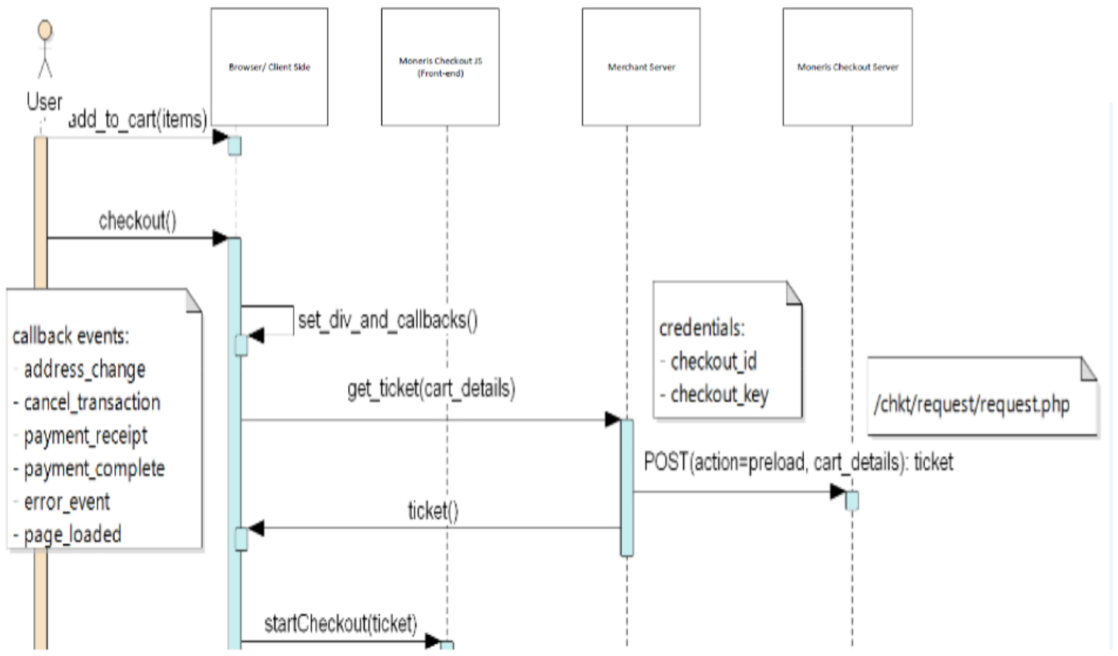
```
<script src="https://gateway.moneris.com/chkt/js/chkt_v1.00.js"></script>
```

2.4 Moneris Checkout Transaction Process Overview

Moneris Checkout transactions follow a general sequence:

1. When the customer clicks to start the checkout process on your merchant e-commerce site, your back-end server sends a Preload request to the Moneris Checkout server, containing the store ID, API token, checkout ID and other transaction details
2. Moneris responds with a transaction ticket number
3. Your merchant site initiates the `startCheckout` function call, with the transaction ticket number received in the previous step included, to create an iFrame that will be inserted into a div on your website
4. The customer now sees the Moneris Checkout page, where they enter their cardholder details
5. When the customer clicks Checkout on the Moneris Checkout page, the transaction is processed directly by Moneris
6. When transaction is finished, the JavaScript callback `paymentComplete` will be triggered, and depending on how you chose to configure your Moneris Checkout integration:
    a. If you have chosen to have Moneris Checkout handle the receipt, Moneris will display the receipt in the browser
    b. If you will handle the receipt, your back-end server sends a Receipt request to the Moneris Checkoutserver

2.5 Moneris Checkout Process Flow



2.6 Moneris Checkout Requests

There are two types of server requests in Moneris Checkout:

- Preload requests, which ask the Moneris Checkout server to respond with a unique transaction ticket number
- Receipt requests, which ask the Moneris Checkout server to respond with the receipt details once a transaction is completed

The transaction ticket number is valid for that transaction only.

2.6.1 Preload Request in Moneris Checkout

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

Required Preload Request Variables for Moneris Checkout

Attribute or Variable	Size/Type	Description
store ID `store_id`	10-character alphanumeric	Unique identifier provided by Moneris upon merchant account set up
API token `api_token`	20-character alphanumeric	Unique alphanumeric string assigned upon merchant account activation
checkout ID `checkout_id`	30-character alphanumeric	Identifies your Moneris Checkout configuration; this is given to you when you configure your page in the Merchant Resource Center
transaction amount `txn_total`	10-character decimal	The total dollar amount of the transaction
integrator `integrator`	50-character alphanumeric	Your merchant name

developmental mode
`environment`
	alphabetic
	`qa` or `prod`
		Indicates the stage of development you are sending the request for:
		testing = qa
		production = prod

request type	alphabetic	Type of request being made to Moneris Checkout server; is either pre-

Attribute or Variable	Size/Type	Description
`action`	`preload` or `receipt`	load or receipt request

Optional Preload Request Variables

Attribute or Variable	Size/Type	Description
order number `order_no`	30-character alphanumeric	
customer ID `cust_id`	30-character alphanumeric	Merchant-defined field that can be used as an identifier
dynamic descriptor `dynamic_descriptor`	maximum 20-character alphanumeric total of 24 characters including the merchant name	Merchant-defined description sent on a per-transaction basis that will appear on the credit card statement appended to the merchant's business name

Optional Objects for Preload Requests in Moneris Checkout

Attribute or Variable	Size/Type	Description
Shopping Cart `cart`	Object	The virtual shopping cart and its contents
Contact Details `contact_details`	Object	Customer contact information
Shipping Details `shipping_details`	Object	Customer shipping information
Billing Details `billing_details`	Object	Customer billing information
Shipping Rates `shipping_rates`	Object	List of shipping rates to be applied

2.6.1.1 Example Preload JSON Request

This example reflects a Preload request with all optional objects.

```
{
   "store_id":"example_apiToken",
   "api_token":"example_apiToken",
   "checkout_id":"example_checkoutId",
   "integrator":"cr_dev",
   "txn_total":"50.00",
   "environment":"qa",
   "action":"preload",
   "order_no":"",
   "cust_id":"chkt - cust - 0228",
   "dynamic_descriptor":"dyndesc",
   "language":"en",
   "billing_details":{
      "address_1":"1 main st",
      "address_2":"Unit 2000",
      "city":"Toronto",
      "province":"ON",
      "country":"CA",
      "postal_code":"M1M1M1"
   },
   "shipping_details":{
      "address_1":"1 main st",
      "address_2":"Unit 2012",
      "city":"Toronto",
      "province":"ON",
      "country":"CA",
      "postal_code":"M2M1M1"
   },
   "contact_details":{
      "first_name":"bill",
      "last_name":"smith",
      "email":"emailaddress@email.com",
      "phone":"4165551234"
   },
   "credential_on_file":{
      "payment_indicator":"C"
   }
}
```

2.6.2 Receipt Request in Moneris Checkout

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

Required Receipt Request Variables for Moneris Checkout

Attribute or Variable	Size/Type	Description
store ID `store_id`	alphanumeric	Unique identifier provided by Moneris upon merchant account set up
API token `api_token`	alphanumeric	Unique alphanumeric string assigned upon merchant account activation
checkout ID `checkout_id`	30-character alphanumeric (maximum)	Identifies your Moneris Checkout configuration; this is given to you when you configure your page in the Merchant Resource Center
transaction ticket number `ticket`	50-character alphanumeric (maximum)	The unique ticket number that identifies a particular transaction; this returned in the response to the preload request

developmental mode
`environment`
	alphabetic
	`qa` or `prod`
		Indicates the stage of development you are sending the request for:
		testing = qa
		production = prod

request type `action`	alphabetic `preload` or `receipt`	Type of request being made to Moneris Checkout server; is either preload or receipt request

2.6.2.1 Example Receipt JSON Request

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

2.6.2.2 Example Receipt JSON Response

```
{
  "response": {
    "success": "true",
    "request": {
      "txn_total": "7.00",
      "cust_info": {
        "first_name": null,
        "last_name": null,
        "phone": null,
        "email": null
      },
      "shipping": {
        "address_1": "Bloor st",
        "address_2": "3300",
        "city": "Toronot",
        "country": "CA",
        "province": "ON",
        "postal_code": "P0P0P0"
      },
      "billing": {
        "address_1": "3300 Billing Add",
        "address_2": "BAdd2",
        "city": "Toronto",
        "province": "ON",
        "country": "Canada",
        "postal_code": "B0B0B0"
      },
      "cc_total": "7.00",
      "cc": {
        "first6last4": "4012001112",
        "expiry": "1220",
        "cardholder": "card holder"
      },
      "ticket": "1542048678550wGk2AOqCNIb8VqOuf4a9iDoR1YV",
      "cust_id": "Test Customer ID",
      "dynamic_descriptor": "Test Dynamic Desc",
      "order_no": "TestKP694998976",
      "eci": "7"
    },
    "receipt": {
      "result": "a",
      "cc": {
        "order_no": "TestKP694998976",
        "cust_id": "Test Customer ID",
        "transaction_no": "15160-0_11",
        "reference_no": "660125050013190080",
        "transaction_code": "00",
        "transaction_type": "200",
        "transaction_date_time": "2018-11-12 13:51:23",
        "corporateCard": "false",
        "amount": "7.00",
        "response_code": "005",
        "iso_response_code": null,
        "approval_code": "078680",
        "card_type": "V",
        "dynamic_descriptor": "Test Dynamic Desc",
        "invoice_number": null,
        "customer_code": null,
        "eci": "7",
        "cvd_result_code": "1P",
        "avs_result_code": null,
        "first4last4": "4012001112",
        "expiry_date": "1220",
        "recur_success": null,
        "ecr_no": "66012505",
        "batch_no": "319",
        "sequence_no": "008",
        "result": "a"
      }
    }
  }
}
```

2.6.3 Optional Request Objects

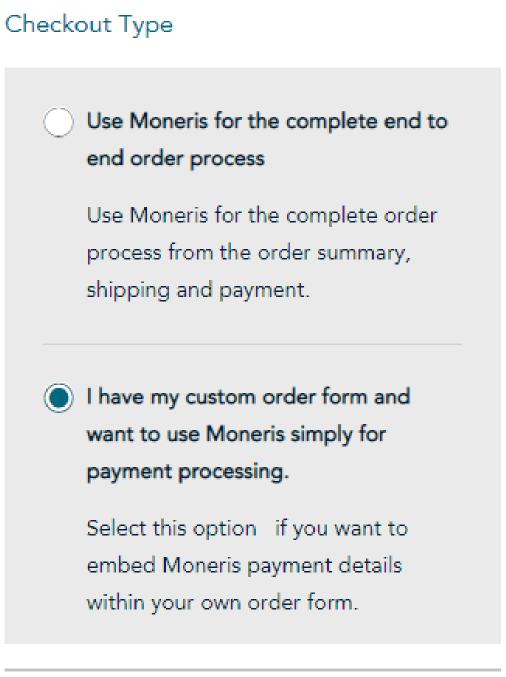
Moneris Checkout also allows you to send optional objects in the Preload request that reflect additional information entered by the customer at checkout, enable additional features, or meet transaction processing requirements.

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:

- Credential on File Object
- Recurring Billing Object
- Shopping Cart Object
- Contact Details Object
- Shipping Details Object
- Billing Details Object
- Shipping Rates 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:


Checkout Type

○ Use Moneris for the complete end to end order process

Use Moneris for the complete order process from the order summary, shipping and payment.

● I have my custom order form and want to use Moneris simply for payment processing.

Select this option if you want to embed Moneris payment details within your own order form.

2.6.3.1 Credential on File Object

Object is required when storing cardholder credentials for use in future transactions.

In Moneris Checkout, this applies to transactions where the customer has agreed to store their cardholder information for future transactions or the transaction is part of a Recurring Billing series managed by Moneris.

Attribute or Variable	Size/Type	Description
Credential on File Info `credential_on_file`	*Object* contains following variables in blue	Required when storing cardholder credentials or using these credentials in subsequent transactions.
payment indicator **`credential_on_file.payment_indicator`**	*String* 1-character alphabetic	**NOTE:** For Moneris Checkout transactions, the payment indicator values will either be C or R Indicates the current or intended use of the credentials Possible values for first transactions: C - unscheduled Credential on File (first transactions only) R - recurring Possible values for subsequent trans-

Attribute or Variable	Size/Type	Description
		actions: R - recurring U - unscheduled merchant-initiated transaction Z - unscheduled customer-initiated transaction In Credential on File transactions where the request field e-commerce indicator is also being sent, the acceptable values for e-commerce indicator are dependent on the value sent for payment indicator, as follows: if payment indicator = R, then allowable values for e-commerce indicator: 2, 5 or 6 if payment indicator = C, then allowable values for e-commerce indicator: 1, 5, 6 or 7 if payment indicator = U, then allowable values for e-commerce indicator: 1 or 7 if payment indicator = Z, then allowable values for e-commerce indicator: 1, 5, 6 or 7

2.6.3.2 Recurring Billing Object

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

Attribute or Variable	Size/Type	Description
Recurring Billing `recur`	*Object* contains following variables in blue:	Contains fields related to Recurring Billing
number of recurs **`recur.number_of_recurs`**	*String* numeric 1-99	The number of times that the transaction must recur
period	*String*	Number of recur unit intervals that must pass between recurring billings

Attribute or Variable	Size/Type	Description
`recur.recur_period`	numeric 1-999	
recurring amount `recur.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.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 `recur.start_date`	*String* YYMMDD 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

2.6.3.3 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 field

`cart`

Required Variables for Shopping Cart

Attribute or Variable	Size/Type	Description
shopping cart items `items`	sub-object nested within `cart` contains following items in blue	Encapsulates the entire array of items in the shopping cart
URL **`items.url`**	alphanumeric	URL that corresponds to the image Moneris Checkout shopping cart item
description **`items.description`**	alphanumeric	Describes the item in the shopping cart
product code **`items.product_code`**	20-character alphanumeric	The SKU for the item
unit cost **`items.unit_cost`**	free form alphanumeric	Per-unit cost of the item
quantity **`items.quantity`**	numeric 6 characters maximum	Number of individual instances of the given item in the shopping cart
subtotal `subtotal`	free form alphanumeric	Total dollar amount of the shopping cart before taxes
tax `tax`	sub-object nested within `cart` contains following items in blue	Contains items related to taxes
tax amount **`tax.amount`**	free form alphanumeric	Dollar amount of taxes
tax description **`tax.description`**	free form alphanumeric	Describes type of tax being applied
tax rate **`tax.rate`**	free form alphanumeric	Percentage tax rate charged

2.6.3.4 Contact Details Object

Optional object

Top Level Field

`contact_details`

Required Variables for Contact Details Object

Attribute or Variable	Size/Type	Description
first name `first_name`	30-character alphanumeric	Customer first name
last name `last_name`	30-character alphanumeric	Customer last name
e-mail `email`	255-character alphanumeric	Customer email
phone number `phone`	30-character alphanumeric	Customer phone number

2.6.3.5 Shipping Details Object

Optional object

Top Level Field

`shipping_details`

Required Variables for Shipping Details Object

Attribute or Variable	Size/Type	Description
shipping address line 1 `address_1`	50-character alphanumeric	Customer shipping address
shipping address line 2 `address_2`	50-character alphanumeric	Customer shipping address
shipping city	30-character alphanumeric	Customer shipping address city

Attribute or Variable	Size/Type	Description
`city`		
shipping province `province`	30-character alphanumeric	Customer shipping address province
shipping country `country`	30-character alphanumeric	Customer shipping address country
shipping postal code `postal code`	30-character alphanumeric	Customer shipping address postal code

2.6.3.6 Billing Details Object

Optional object

Top level field

`billing_details`

Required Variables for Billing Details Object

Attribute or Variable	Size/Type	Description
billing address line 1 `address_1`	50-character alphanumeric	Customer billing address
billing address line 2 `address_2`	50 character alphanumeric	Customer billing address
billing city `city`	30-character alphanumeric	Customer billing address city
billing province `province`	30-character alphanumeric	Customer billing address province
billing country `country`	2-character alphabetic	Customer billing address country
billing postal code	30-character alphanumeric	Customer billing address postal code

Attribute or Variable	Size/Type	Description
`postal code`		

2.6.3.7 Shipping Rates Object

Optional object

Top level field

`shipping_rates`

Required Variables for Shipping Rates Object

Attribute or Variable	Size/Type	Description
shipping rate code `code`	20-character alphanumeric	Unique code you create to identify a shipping rate
shipping rate description `description`	20-character alphanumeric	Description of the shipping rate category
shipping time `date`	free form alphanumeric Suggested form: `X days`	Describes the turnaround time for shipping, described as number of days (suggested)
shipping cost `amount`	free form alphanumeric	Total dollar amount of shipping, as per the rate category, on transaction
shipping tax amount `txn_taxes`	free form alphanumeric	Total dollar amount of taxes on this transaction
transaction total amount `txn_total`	free form alphanumeric	Total dollar amount of transaction

Attribute or Variable	Size/Type	Description
default shipping rate `default_rate`	`true` or `false`	Sets a default rate category

2.6.4 Callbacks

Moneris Checkout uses JavaScript callbacks to enable additional actions or events.

These callbacks are required:

- Cancel Transaction
- Payment Complete

The following callbacks are optional:

- Address Change
- Error Event
- Page Loaded
- Payment Receipt

2.6.4.1 Cancel Transaction

Callback Use

When cardholder cancels transaction.

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

JavaScript Format for Callback

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

JSON Message Format

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

2.6.4.2 Payment Complete

Callback Use

Payment is complete and Moneris Checkout should be closed by calling the `closeCheckout()` method

JavaScript Format for Callback

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

JSON Message Format

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

2.6.4.3 Address Change

Callback Use

Used when the shipping address changes, specifically the values for address line 1, postal code and country.

If you choose to set up Moneris Checkout to change shipping rates when the address changes, you need to include the optional JavaScript for `.setNewShippingRates` in the address change function.

JavaScript Format for Callback

```
myCheckout.setCallback("address_change",myAddressChange);
```

JavaScript for Shipping Rate Change (optional)

```
myCheckout.setNewShippingRates(json_rate);
```

Example Address Change JavaScript Function with Shipping Rate Change

```
function myAddressChange(msg)
{
       rates = {};
       rates["action"] = "set_shipping_rates";
       rates["data"] = {};

       rate = {};
       rate[0] = {};
```

```
        rate[0]["code"] = "rateM1";
        rate[0]["description"] = "Standard R";
        rate[0]["date"] = "4 days";
        rate[0]["amount"] = "Free";
        rate[0]["txn_taxes"] = "0.01";
        rate[0]["txn_total"] = "1.00";
        rate[0]["default_rate"] = "false";

        rate[1] = {};
        rate[1]["code"] = "rateM2";
        rate[1]["description"] = "Express R";
        rate[1]["date"] = "0.5 day";
        rate[1]["amount"] = "11.01";
        rate[1]["txn_taxes"] = "2.01";
        rate[1]["txn_total"] = "13.01";
        rate[1]["default_rate"] = "true";

        rates["data"] = rate;

        json_rate = JSON.stringify(rates)
        console.log(json_rate);

        myCheckout.setNewShippingRates(json_rate);
}
```

JSON Message Format

```
{
   "handler":"address_change",
   "response":"001",
   "ticket":"1539961059DdrvGG3Yj7rxvMAgvRlc4nqKXF7YjT",
   "address":{
      "address_1":"1 Main Ave",
      "address_2":"",
      "city":"Toronto",
      "country":"Canada",
      "province":"Ontario",
      "postal_code":"M1M1M1"
   }
}
```

2.6.4.4 Error Event

Callback Use

When an error occurs during the checkout process. This requires the Moneris Checkout session to be closed and for a new Preload request to be sent to the Moneris Checkout server in order to get a new transaction ticket number.

JavaScript Format for Callback

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

JSON Message Format

```
{
   "response":{
      "success":"false",
      "error":{
         "cust_id":"Invalid data",
         "shipping_details":{
            "address_1":"Invalid data",
            "city":"Invalid data",
            "province":"Invalid data"
         },
         "billing_details":{
            "city":"Invalid data"
         },
         "recur":{
            "recur_amount":"Invalid amount",
            "recur_unit":"Invalid data",
            "start_date":"Invalid date"
         },
         "credential_on_file":{
            "payment_indicator":"Invalid data"
            }
        }
    }
}
```

2.6.4.5 Page Loaded

Callback Use

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

JavaScript Format for Callback

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

JSON Message Format

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

2.6.4.6 Payment Receipt

Callback Use

Transaction is complete and receipt is ready to be collected.

If you have chosen to have Moneris Checkout display the receipt, this callback is triggered when the receipt is displayed.

JavaScript Format for Callback

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

JSON Message Format

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

2.7 Window Sizing Behaviour in Moneris Checkout

2.7.1 About Window Sizing in Moneris Checkout

You can customize the appearance of the Moneris Checkout window that is presented to the customer on their web browser, including the size of the window.

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. Opting for a window view requires a few extra steps; for more information on what to do, see 2.7.2 Implementing Window View in Moneris Checkout.

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

2.7.2 Implementing Window View in Moneris Checkout

If you want the Moneris Checkout window to be presented as a separate window instead of taking over the entire web 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. On the left side of the page, expand the **Branding & Design** tab
4. Under the Customizations heading, uncheck the box next to **Enable Fullscreen**
5. In your web page code, create a container `div` element with attributes that define the size that the window will be
6. Inside the container `div`, create the `div` that corresponds to the Javascript call `myCheckout.setCheckoutDiv("MonerisCheckout");`

2.8 Tokenization of Credentials With Moneris Checkout

2.8.1 About Tokenization With Moneris Checkout

When performing transactions with Moneris Checkout, you can opt to store the cardholder's credentials using the Moneris Vault.

If you want to tokenize credentials in Moneris Checkout transactions, you configure this behaviour in your settings in the Moneris Merchant Resource Center. For more on how to do this, see 2.8.2 Tokenizing Credentials in Moneris Checkout.

2.8.2 Tokenizing Credentials in Moneris Checkout

Tokenizing cardholder credentials to store in the Vault requires you to select this behaviour in the Merchant Resource Center.

To tokenize cardholder credentials in Moneris Checkout transactions:

1. In a web browser, go to the Merchant Resource Center page corresponding 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 **Edit** button next to the store you are configuring
4. In the Transaction Type section, check the box next to **Tokenize Card** to turn on tokenization

2.9 Fraud Tools in Moneris Checkout

2.9.1 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 configuration settings in the Moneris Merchant Resource Center. For more on how to do this, see 2.2 Configuring Moneris Checkout in Merchant Resource Center.

2.9.2 Selecting Fraud Tools in Moneris Checkout

You select which fraud tools to use in Moneris Checkout transactions in your configuration settings in the Merchant Resource Center.

To select which fraud tools to use with Moneris Checkout transactions:

1. In a web browser, go to the Merchant Resource Center page corresponding 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 **Edit** button next to the store you are configuring
4. Expand the **Payment** tab on the left side of the page
5. In the Payment Security section, use the check boxes to select which fraud tools to use
6. (*optional*) To have Monerisautomatically decide whether to approve or deny transactions based on fraud tool risk information, select **Auto Decision by Moneris**

2.9.3 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.

2.9.4 Fraud Tools and Auto Decision-Making

You can choose to have Moneris Checkout automatically approve or deny transactions based on the risk information compiled by the selected fraud tools.

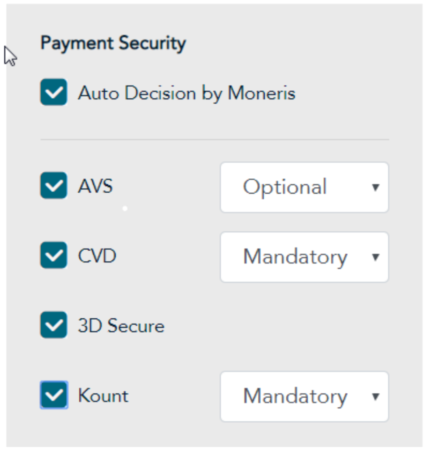
When you enable auto decision-making, you also can choose whether each fraud tool's analysis will be treated as an optional or mandatory factor in the decision to approve or deny the transaction. This information applies to all fraud tools with the following exceptions:

- 3-D Secure, which is always mandatory *if* enabled
- CVD is always mandatory and is enabled for all transactions, regardless of whether auto decision-making is turned on or not
- Auto decision-making is not

2.9.5 Enabling Auto Decision-Making in Moneris Checkout

Auto decision-making is enabled in your store's Moneris Checkout configuration settings in the Merchant Resource Center:

1. In the Payments tab under the heading Payment Security, check the **Auto Decision by Moneris** box
2. For each fraud tool, choose whether that tool's report will be considered an optional or mandatory factor for approving or denying a transaction


Payment Security

☑ Auto Decision by Moneris

☑ AVS — Optional

☑ CVD — Mandatory

☑ 3D Secure

☑ Kount — Mandatory

For more information on how, see 2.9.2 Selecting Fraud Tools in Moneris Checkout

2.9.6 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 (acquirer sent both street address and zip/postal code)	N/A	N/A
D	Street address and zip/-postal code match	N/A	Customer name incorrect; zip/postal code matches

Code	Visa	Mastercard/Discover	American Express/ JCB
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
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	Neither address nor zip/-postal code matches	Billing address and zip/-postal code do not match

Code	Visa	Mastercard/Discover	American Express/ JCB
	only, or • street address only, or • both postal code and street address Also used when acquirer requests AVS but sends no AVS data		
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 incom-patible 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 sup-ported	AVS currently not sup-ported
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 reas-ons: • Issuer is not an AVS participant, or	No data from issuer-/authorization system	Information is unavailable

Code	Visa	Mastercard/Discover	American Express/ JCB
	• 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	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
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

2.9.7 CVD Response Codes – Moneris Checkout

2.9.8 3-D Secure Response Codes – Moneris Checkout

Code	Message	Significance to Merchants
0	CAVV authentication results invalid	For this transaction, you may not receive protection from chargebacks as a result of using VbV because the CAVV was considered invalid at the time the financial transaction was processed. Check that you are following the VbV process correctly and passing the correct data in our transactions.
1	CAVV failed validation; authentication	Provided that you have implemented the VbV process correctly, the liability for this transaction should remain with the Issuer for chargeback reason codes covered by Verified by Visa.
2	CAVV passed validation; authentication	The CAVV was confirmed as part of the financial transaction. This transaction is a fully authenticated VbV transaction (ECI 5)
3	CAVV passed validation; attempt	The CAVV was confirmed as part of the financial transaction. This transaction is an attempted VbV transaction (ECI 6)
4	CAVV failed validation; attempt	Provided that you have implemented the VbV process correctly the liability for this transaction should remain with the Issuer for chargeback reason codes covered by Verified by Visa.
7	CAVV failed validation; attempt (US issued cards only)	Please check that you are following the VbV process correctly and passing the correct data in your transactions. Provided that you have implemented the VbV process correctly the liability for this transaction should be the same as an attempted transaction (ECI 6)
8	CAVV passed validation; attempt (US issued cards only	The CAVV was confirmed as part of the financial transaction. This transaction is an attempted VbV transaction (ECI 6)

Code	Message	Significance to Merchants
9	CAVV failed validation; attempt (US issued cards only)	Please check that you are following the VbV process correctly and passing the correct data in our transactions. Provided that you have implemented the VbV process correctly the liability for this transaction should be the same as an attempted transaction (ECI 6)
A	CAVV passed validation; attempt (US issued cards only)	The CAVV was confirmed as part of the financial transaction. This transaction is an attempted VbV transaction (ECI 6)
B	CAVV passed validation; information only, no liability shift	The CAVV was confirmed as part of the financial transaction. However, this transaction does not qualify for the liability shift. Treat this transaction the same as an ECI 7.
C	CAVV was not verified — attempted authentication	If 3-D Secure Authentication Results Code value is 07 in the CAVV and the issuer did not return a CAVV results code in the authorization response
D	CAVV was not verified — cardholder authentication	If 3-D Secure Authentication Results Code value is 00 in the CAVV and the issuer did not return a CAVV results code in the authorization response

3 Testing Your Moneris Checkout Integration

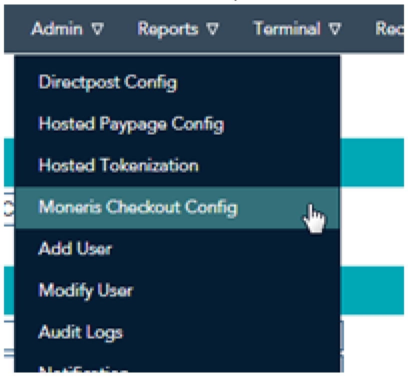
In the testing stage of development:

1. Use the testing Merchant Resource Center at https://esqa.moneris.com/mpg to configure your Moneris Checkout page for testing purposes
2. In all Preload requests use the value "**`qa`**" for the `environment` variable
3. In all Preload requests, make sure that you are using the testing version of your credentials for **Store ID**, **API token** and **Checkout ID**

4 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:
   a. Log in to your production MRC store at https://www3.moneris.com/mpg
   b. In the Admin menu, select **Moneris Checkout Config**



   c. Find your Checkout ID in the list and click **Edit**
2. In all Preload requests, use the value `prod` for the `environment` request variable
3. In all Preload requests, make sure that you are using the production version of your credentials for **Store ID**, **API token** and **Checkout ID**