• verifone

https://verifone.cloud/docs/uk-gateway/checkout_overview Updated: 10-Jun-2022

Checkout

Checkout is an end-to-end solution, which facilitates payments through a dedicated payment page which is hosted on a secure PCI compliant environment. It allows for a merchant to provide rich custom styling and layout to determine the exact looks of the page. Read more about PCI and how your integration methods influence that <u>here</u>. Want to know more about the differences between the different integration methods, read more about that <u>here</u>.

The following payment methods support processing transactions through Checkout:

• Card



Creation

Creating a Checkout is done via a single API call to POST /v1/checkout

Example body of the request:

```
{
    "account": "54884a22e1e6573d1d1ee001",
    "amount": 1750,
    "customer": "54686a32e3e6773d1d1ee006",
    "merchant_reference": "ORDER-1234",
    "template": "https://merchantwebsite.com/order/1234/template",
    "return_url": "https://merchantwebsite.com/order/1234/return",
    "css_framework": "bootstrap-3.3.7",
    "configurations": {
        "card": {
            "capture_now": true,
            "dynamic_descriptor": "Supermerchant Amsterdam",
            "threed_secure": {
                "enabled": true,
                "description": "Short description, shown on 3DS page"
            }
        }
    }
}
```

The request body consists of several components, which are explained in more detail below.

Example body of a successful response:

```
{
    "_id": "37285b25e5a65d3d1d1ea0ac",
    "url":
    "https://sandbox.omni.verifone.cloud/v1/checkout/view/37285b25e5a65d3d1d1ea0ac.html
```

}

- url Points to the newly created Checkout page. The Checkout page uses the provided template and contains a payment form with card and paypal payment methods available.
- _id Checkout identifier, useful for later on lookups

Template

A Checkout cannot exist without a Checkout template (template). The Checkout template is not forced to use any specific layout or styling, this freedom is left to the Merchant. This freedom also extends to external assets, such as images and CSS styling.

HTML

- Cannot contain DOM elements with duplicated id attribute
- The <title> tag is not allowed in the template.
- DOM elements cannot contain any event listeners such as onclick, onhover. See Constraints section

• Must contain exactly one <form> element inside of the <body> tag. This form element is used as the location where the payment form would be placed when the Checkout page gets created.

CSS

Styling the page can be done by using CSS provided in the <head> of the Checkout template. The CSS will be fetched, sanitized and available on the resulting Checkout page.

CSS can be provided in two ways:

- inline as: <style type="text/css"></style>
- externally linked: <link rel="stylesheet" href="/stylesheets/my_style.css"> The General constraints section

CSS frameworks In addition to being able to provide custom CSS, it is also possible to make use of a list of popular CSS frameworks. These CSS frameworks allow merchants to style the page to be consistent with their existing website.

Currently, the supported CSS frameworks as:

- <u>bootstrap</u>
- <u>materialize</u>

Supported css_framework values are:

- bootstrap-3.3.7
- materialize-0.100.1

In order to include a framework, you should use the css_framework property (see above).

For example, including the bootstrap framework would have the following body property:

```
{
    ...
    "css_framework": "bootstrap-3.3.7"
    ...
}
```

NOTE: Including one of those libraries can only be done via the css_framework property and not by manually adding them on the Checkout template page.

Form Styling applied on the page can also used to style the payment form which is injected into the <form> tag. Whenever a css_framework is used, all the elements within the form will have the corresponding classes assigned to them, so they can have consistent look and feel. It is always recommended that you try out different templates in sandbox before using in production.

Constraints All provided styling needs to be valid against the whitelist of allowed attributes, which can be found <u>below</u>.

Images

Templates can include images. Images can be added using and tags:

```
<img src="https://merchant.website/static/docs/images/myimage.jpg">
```

Images can only be provided with one of the following format: jpeg, png

Constraints

- Maximum size of template: **10MB** Calculated after all assets have been fetched. While this is the hard limitation, it is highly recommended try and minimize the total size by optimizing page assets. This ensures the Checkout page load time as fast as possible for the customer.
- Maximum size per asset: **1MB** Each image and stylesheet cannot be larger than 1MB
- Maximum time for fetching an asset: 15 sec
- Same domain All images and stylesheets should be served from the same domain, as the template. Usage of both relative and absolute paths is accepted. Any usage of other external links or assets not from the same domain will result in an error.
- Javascript The template cannot contain **any** javascript code. This is to ensure the security of sensitive customer details provided in the form, such as credit card details.

Caching

Checkout template pages provided in the template are Cached to ensure they load as fast as possible. This ensures the images and stylesheets are not fetch each time a Checkout request is made. If a Checkout request uses a template which has been used previously, the Checkout page loading time is sped up by using Cached version.

If changes are made to the Checkout template code, a unique URL is required to enforce obtaining the code changes. Is is recommended to make the template unique by passing a query string parameter or having a dynamic path parameter. All Transaction and Payment method specific details would always be applied as provided in the payload regardless of Caching. This means that payment methods made available on the form can change, but also all details about the actual transaction (amount, customer, etc).

Dynamic values

Checkouts can display a set of values on the page, specific to the given payment. This allows to make use of a cached template, while showing actual information about the payment on the page. Making use of those values is as simple as setting the id attribute of an HTML element to one from the table below.

id	
amount	amount
merchant_reference	merchant_reference
customer_first_name	customer.first_name
customer_last_name	customer.last_name
customer_company_name	customer.company_name

account.organisation.name

Default

A default template is hosted and ready to use <u>on this link</u>. This template is made to work with either the bootstrap or materialize css frameworks, but it is not suitable to being used without any css framework. The html and css which the template is made out of are free to use by merchants as a starting point for their own custom template.

Transaction parameters

This part of the body concerns common details about the Transaction

- **account** The account id that you intend to use for receiving the payment. The currency of the transaction will be inferred from the currency of this account.
- **amount** The amount to be charged, in digits only, stated in the minor units of the currency, without any decimal point or other punctuation. For example, €1.00 should be stated as "100", while ¥100 should also be stated as "100".
- customer The customer id of an existing customer.
- **merchant_reference** Reference used by the Merchant to identify the payment. Typically this value is an order ID or similar reference from the Merchant's system.

These parameters would be used when initiating an actual transaction, once the Customer has submitted the form with a chosen payment method.

Payment method specific details

The configurations parameter is used to determine which payment methods should be available on the page.

One or more configurations can be included in the configurations object as:

```
{
    ...
    "configurations": {
        "card": {
            // Card configuration goes here
        }
    }
    ...
}
```

For each payment method, there is a different set of configuration values which should be provided, visit the Checkout page per payment method to view their specific requirements:

• <u>Card</u>

Payment

Once a Checkout has been successfully created, a payment can be executed with any of the payment methods configured. There are no more actions required by the Merchant in order for the payment to be completed as it's handled within the Checkout life cycle.

Form submissions

A checkout is intended as a single use payment page. The URL of every checkout would become inactive, as soon as a payment has been successfully completed through it. A total of 3 payment attempts are allowed per checkout, after which the page would become inactive.

Handling results & errors

Both success / failure outcomes result in the customer being redirected back to the redirect_url provided when creating the Checkout.

Success

If a transaction was completed successfully, a transaction_id will be appended to the url as a query string parameter.

A transaction initiated through Checkout will be considered successful if it has SETTLEMENT_REQUESTED, SETTLEMENT_SUBMITTED, SETTLEMENT_COMPLETED or AUHTORIZED as a status. Other statuses will be considered unsuccessful.

Example:

```
https:
//merchantwebsite.com/order/1234/return?transaction_id=15238b36f5e2273d1d1ee001
```

Additionally, it's best practice for Merchants to have Webhook URL configured on their Account, such that they are notified as soon as a Transaction has been created.

Failure

In case that the something went wrong within the payment flow, an error query parameter will be appended to the url. The value of the error is a url encoded error message, explaining the reason for the failure.

Example:

https:

//merchantwebsite.com/order/1234/return?error=The%20transaction%20has%20been%20de

3DS Scenarios

Unsuccessful 3D authentication automatically blocks authorisation

If a 3D authentication is unsuccessful (where the pares_status is N, R or C) the Checkout page will not continue with the authorisation. In this situation is that the return_url is appended by the authentication ID and the error string for not continuing with the transaction.

Example

```
{{return_url}}
?authentication_id=5d5e9cc0d6ef09600ed160d5&error=error%20string
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to true and configurations.card.threed_secure.enabled to true.
- 2. Use PAN 400000000001018 with month 01 and the year set to the current year + 3. If the current year is 2019 than set the year to 22.

Unsuccessful standalone 3D authentication

For unsuccessful standalone 3D authentications using Checkout the authentication Id is returned along with an error string appended to it in the return_url.

Format

```
{{return_url}}
?authentication_id=5d5e9cc0d6ef09600ed160d5&error=error%20string
```

Scenarios:

1. Authentication has failed due to Signature Verification failure ("signature_verification": "N"
)

Example

```
{{return_url}}?authentication_id=5d8cbc0a0f2f2c5669e8dad5&error=%27
Transaction%20cannot%20be%20initiated%20because%20signature%20
verification%20in%20the%20authentication%20process%20failed.%27
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to false and configurations.card.threed_secure.enabled to true.
- 2. Use PAN 400000000000000000 with month 01 and the year set to the current year + 3. If the current year is 2019 than set the year to 22.
- 2. Authentication has failed ("pares_status": "N")

Example

```
{{return_url}}
?authentication_id=5d8cbd830f2f2c5669e8db00&error=%27Transaction%20cannot%20
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to false and configurations.card.threed_secure.enabled to true.
- 2. Use PAN 40000000001018 with month 01 and the year set to the current year + 3. If the current year is 2019 than set the year to 22.
- 3. Authentication has failed ("pares_status": "R")

Example

```
{{return_url}}?authentication_id=5d8cc58f0f2f2c5669e8dc64&error=%27
Transaction%20cannot%20be%20initiated%20because%20issuer%20rejected%20
the%20authentication.%27
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to false and configurations.card.threed_secure.enabled to true.
- 2. Use PAN 400000000001042 with month 01 and the year set to the current year + 3. If the current year is 2019 than set the year to 22.
- 4. An error happened during the authentication ("error_no": "value")

Example

```
{{return_url}}
?authentication_id=5d8cc58f0f2f2c5669e8dc64&error=%27An%20error%20took
place%20during%20the%20authentication.%20Please%20refer%20to%20the%20Reason%
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to false and configurations.card.threed_secure.enabled to true.
- 2. Use PAN 400000000001067 with month 01 and the year set to the current year + 3. If the current year is 2019 than set the year to 22.

Successful standalone 3D authentication

For successful standalone 3D authentications using Checkout the authentication Id is appended to it in the return_url.

Example

```
\{\{\texttt{return\_url}\}\}? \texttt{authentication\_id=5d5e9cc0d6ef09600ed160d5}
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to false and configurations.card.threed_secure.enabled to true.

Successful 3D authentication and unsuccessful authorisation

If a 3D authentication is successful the Checkout page will automatically continue with the authorisation. If the authorisation is unsuccessful (the transaction has the status DECLINED, FAILED, UNKNOWN or it is not possible to create the transaction due to an internal server error) than

Example

```
{{return_url}}
?authentication_id=5d5e9cc0d6ef09600ed160d5&error=error%20string
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to true and configurations.card.threed_secure.enabled to true. Set the merchant_reference to please 123.

Successful 3D authentication and successful authorisation

If a 3D authentication is successful the Checkout page will continue with the authorisation. After successful authorisation both the authentication ID and the transaction ID are appended to the return_url.

Example

```
{{return_url}}
?authentication_id=5d5e9cc0d6ef09600ed160d5&transaction_id=5w839cc0d3h8d0600ed16
```

Steps for reproducing

- 1. Create a Checkout and set configurations.card.process_transaction to true and configurations.card.threed_secure.enabled to true.

Transaction Scenarios

Unsuccessful authorisation

If an authorisation fails than an error string will be appended to the return_url.

Example

{{return_url}}?error=The%20transaction%20has%20been%20declined%20by%20the% 20processor.%20The%20request%20reached%20the%20processor,%20and%20was%20 valid,%20but%20it%20was%20not%20accepted.%20A%20decline%20reason%20code%20i s%20available%20in%20the%20details%20property.

Successful authorisation

If an authorisation is successful the transaction_id will be appended to the return_url.

Example

{{return_url}}?transaction_id=5d5e9cc0d6ef09600ed160d5

Allowed stylesheet rules

```
"-moz-border-radius-bottomleft"
"-moz-border-radius-bottomright",
"-moz-border-radius-topleft",
"-moz-border-radius-topright",
"animation",
"animation-delay",
"animation-direction",
"animation-duration",
"animation-fill-mode",
"animation-iteration-count",
"animation-name",
"animation-play-state",
"animation-timing-function",
"appearance",
"azimuth",
"backface-visibility",
"background",
"background-attachment",
"background-color",
"background-image",
"background-position",
"background-repeat",
"background-size",
"border",
"border-bottom",
"border-bottom-color",
"border-bottom-left-radius",
"border-bottom-right-radius",
"border-bottom-style",
"border-bottom-width",
"border-collapse",
"border-color",
"border-left",
"border-left-color",
"border-left-style",
"border-left-width",
"border-radius",
"border-right",
"border-right-color",
"border-right-style",
"border-right-width",
"border-spacing",
"border-style",
"border-top",
"border-top-color",
"border-top-left-radius",
"border-top-right-radius",
"border-top-style",
"border-top-width",
"border-width",
"bottom",
"box",
"box-shadow",
"box-sizing",
"caption-side",
"clear",
"clip",
"color",
"content",
"cue",
```