https://verifone.cloud/docs/online-payments/checkout/accepting-payments
Updated: 28-Mar-2024

Accepting payments

Overview

Checkout can be used in several different ways to start accepting payments. These solutions share the same Checkout APIs and have a lot in common in terms of integration. You can find the integration paths on this page.

To generate checkout links via Checkout API, **you will need a Verifone Central user with one of the following roles** assigned to it:

- Merchant Supervisor
- Merchant Cashier

Check the Supported acquirers documentation for more information on payment methods, card brands and acquirer details.

Requirements

In order to start accepting eCommerce transactions, you will need to generate a Secure Card Capture Key.

Get your API endpoints here!

Hosted Payments Page (HPP)

With this approach, the shopper is redirected to a payment page hosted by Verifone to complete the payment. The shopping cart details, products and quantities can also be sent when doing HPP. Upon completion of the payment process, the shopper is redirected back to the URL of the merchant.

For Israel ABS, no card expiration check will be done.

Iframe

Iframe allows for merchants to display a payment form component as part of their own website, without having to redirect the shopper. The form is still securely hosted by Verifone, so there is no additional PCI scope required for the merchant with this solution.

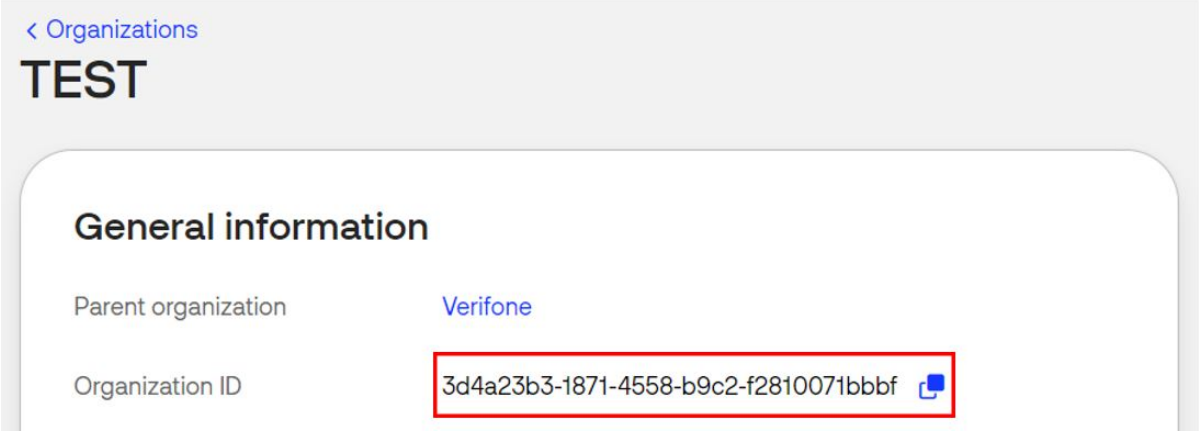
Required fields

To configure Checkout for accepting payments, the following fields are required:

- `entity_id` - This value can be found in the Verifone Central portal or provided to you by a Verifone employee.

To obtain the entity_id value from Verifone Central, go to Administration > Organizations > [Organization] > Organization ID value.

Organization ID (Entity ID) in Verifone Central



- `currency_code`
- `amount`
- `configurations` - Object with one or more payment method configurations.

Optional Checkout fields

- `customer` - ID of a Customer created via the Customer API. A customer object can be created and attached to a Checkout. The customer object can store relevant customer details. Some of these details might be required, depending on the payment method and/or authentication mechanisms (e.g., 3DS) used in the Checkout.
- `expiry_time` - Expiry time of the Checkout page. Defaults to 15 min from creation time. If a user tries to visit or use an expired Checkout, an error will be shown. Maximum value is 30 days.
- `merchant_reference` - Reference provided by Merchant to identify the Checkout and the transaction initiated from it.
- `return_url` - URL belonging to the Merchant website, where the Customer would be redirected after the Checkout has been completed. If this parameter is used, the customer will be redirected after spending 10 seconds on the payment confirmation page provided by Verifone in case of HPP and PAYMENT LINK transactions. For IFRAME transaction the redirect is performed right after the checkout page without passing through the payment confirmation page.
- `shop_url` - URL belonging to the Merchant website, where the Customer would be redirected in case of cancelled Checkout.
- `i18n` - Language preferences for the page. Read Localisation for details.
- `interaction_type` - Indicates the type of integration. Allows for distinguishing payments accepted via different channels. If this parameter is not used, then the createCheckout API call will generate an HPP

payment link.
    - HPP - For payments done as Hosted Payments Page integration.
    - IFRAME - For payments done as an Iframe integration.
    - PAYMENT_LINK - For payments done through the Payment link feature in the merchant portal.

A privacy policy link in the HPP, IFRAME, PBL will be visible in case a PPC used is set up with Intercard acquirer.

- `notification_methods` - This parameter is used for the PAYMENT_LINK interaction type in order to specify the payment link delivery method.
    - email
    - sms
- `display_line_items` - Indicates whether line items should be displayed on the page. Defaults to true.
- `line_items` - List of shopping cart items, to be displayed on the page. Read Line items for details.
- `theme_id` - The parameter that was created within the theming API call that applies the customizations upon the checkout interface. Read Theming for details.
- `receipt_type` - This parameter can be used to send different invoices to the customer.
    - INVOICE - requires line_items to be specified in Israel
    - FULL_RECEIPT - requires line_items to be specified in Israel
    - SIMPLE_RECEIPT
    - INVOICE_RECEIPT

Creating a Checkout

- HPP
- Iframe

Here is an example body creating a Checkout to be used as a Hosted Payments Page (default interaction_type).

```
{
  "amount": 74.55,
  "currency_code": "EUR",
  "entity_id": "{{entity_id}}",
  "configurations": {
    "{{payment_method}}": {
      // varies per payment_method
    }
  },
  "merchant_reference": "SNKRS-7001",
  "return_url": "https://merchantwebsite.com/order/1234",
  "interaction_type": "HPP" // also default value
}
```

The response for creating the Checkout will look like this.

```
{
  "id": "38615263-ed67-4774-bddd-7407edc0b700",
  "url": "{{host}}/v2/checkout/38615263-ed67-4774-bddd-7407edc0b700/view"
}
```

Here is an example body for creating a Checkout to be used as Iframe.

```
{
  "amount": 74.55,
  "currency_code": "EUR",
  "entity_id": "{{entity_id}}",
  "configurations": {
    "{{payment_method}}": {
      // varies per payment_method
    }
  },
  "merchant_reference": "SNKRS-7001",
  "return_url": "{{merchant_return_url}}",
  "interaction_type": "IFRAME"
}
```

The response for creating the checkout will look like this.

```
{
  "id": "38615263-ed67-4774-bddd-7407edc0b700",
  "url":
"{{host}}/v2/loader.js?checkoutId=38615263-ed67-4774-bddd-7407edc0b700"
}
```

In the card field, a digits limitation has been set based on the active payment methods configured on the Payment Provider Contract.

Displaying payment to the shopper

-
-
- HPP
-
- Iframe

Redirect the shopper to the URL received as a response to creating a Checkout. They will see a ready-to-use payment page, displaying a summary of the payment details and the option to do a payment.

1. Inject <script> inline

The URL from the response above can be used to place a <script> tag as direct child of an HTML element where the Iframe should be displayed. This way, the script would automatically render the payment form in the same place of the HTML document as it is included.

```
<html>
    <head>
    ...
    </head>
    <body>
    ...
    <div id="payment_form_container">
         <script defer src="{{host}}
/v2/loader.js?checkoutId=38615263-ed67-4774-bddd-7407edc0b700"></script>
    </div>
    </body>
</html>
```

2. Inject script anywhere on the page

The script can also be included anywhere in the merchant's HTML, as long as there is a containerId query parameter provided pointing to the ID of the parent HTML element in which the form should be displayed. So, if the desired parent element is a with id "payment_form_container", this can look like:

```
<html>
    <head>
    ...
        <script defer src="{{host}}
/v2/loader.js?checkoutId=38615263-ed67-4774-bddd-7407edc0b700&containerId=paymen
></script>
    </head>
    <body>
    ...
    <div id="payment_form_container">
    </div>
    </body>
</html>
```

Handling payment response

Whenever a payment is performed by a shopper through an HPP or IFRAME, it will result in a payment confirmation page.

If a `return_url`is specified by the merchant when creating a Checkout, the customer will be redirected to the `return_url` after the confirmation page. Depending on the result, the redirection would include **transaction_id** if a transaction is created and/or **error_code** if an error has occurred.

The redirection URL for a successful transaction would look like this:

```
https:
//merchantwebsite.com/order/1234?transaction_id={transaction_id}&checkout_id={ch
```

Both success and failure Checkout outcomes result in the customer being redirected to the confirmation page/ failed payment page. If a `return_url` is used, the customer will be redirected to that specific page afterwards.

Note: The `return_url` you have provided needs to link the Checkout ID with the order on your system, to be able to reliably handle the response.

Example:

- Customer places order and your system generates order ID `1234`
- You create a Checkout for this order and set the `return_url` to `https://merchantwebsite.com/order/1234`
- The API responds with `id` and `url`. You ensure that this `id` is linked to the `order_id` in your system, so that you can match them later.
- You redirect the customer to the `url`
- The customer completes the Checkout and is redirected to the `return_url` with query string parameters appended to it by Verifone. For a successful transaction, the final redirection URL would look like: https://merchantwebsite.com/order/1234?transaction_id={transaction_id}&checkout_id={checkout_id}
- Because your return URL uniquely identifies the `order_id`, you know which checkout belongs to the order
- To determine the outcome, you can read back the Checkout by calling the `GET /v2/checkout/{checkout_id}` using the `id` of the Checkout stored earlier and inspect the `events` field

Note: To ensure that the redirection request was not tampered with, always check that the `transaction_id` received as a query parameter in the redirection matches the `transaction_id` property of the retrieved Checkout. If those are not matching, this is an indication of either an incorrect integration, that the redirection to your `return_url` did not originate from Verifone, or the `transaction_id` was tampered with.

- You can now store the `transaction_id` value together with the order `1234` in your system to link the two together

Read Checkout

To determine the outcome of the Checkout, the merchant should query the checkout by doing a Server-to-Server call to **GET /v2/checkout/{checkout_id}**. In the response, they would find the **events** field.

Successful transaction response:

```
[
  ...
  {
      "type": "TRANSACTION_SUCCESS",
      "id": "f2041250-4fc2-4b3a-bc94-651ba099541a",
      "timestamp": "2020-07-08T12:42:37.974Z",
      "details": {
         // varies per payment method
      }
  }
  ...
]
```

Failed transaction response:

```
[
  ...
  {
      "type": "TRANSACTION_FAILED",
      "id": "f2041250-4fc2-4b3a-bc94-651ba099541a",
      "timestamp": "2020-07-08T12:42:37.974Z",
      "details": {
         // varies per payment method
      }
  }
  ...
]
```

The exact list of event types and the keys in the **details** object vary per payment method.

Next steps

- Card payments
- Card payments with 3-D Secure
- Alternative payment methods
- Theming

APIs used to update the transaction status

- Actions that can be performed on a transaction after it was initiated are listed in the Verifone eCommerce API.