Card payments

Overview

Checkout can be used to accept card payments with minimal PCI requirements for merchants.

This guide requires familiarity with ***Accepting payments***.

Our checkout module utilizes Checkout API. For a full understanding on how to accept payments, use the Checkout API documentation.

Read more

Card processing fields

The `configurations.card` object carries the fields used for making a card payment. In this object, the fields are:

- `mode` - Determines the use of the Checkout
  - CARD_CAPTURE - only card details are captured, and no payment is processed; no transaction ID is generated, and the payment confirmation page is not displayed
  - 3DS - card details are captured and 3DS will be processed; no transaction ID is generated, and the payment confirmation page is not displayed
  - PAYMENT - non 3DS successful transaction
  - 3DS_PAYMENT - 3DS successful transaction if 3DS body provided with 3DS enabled true
- `dynamic_descriptor` - Value is displayed as short text on the bank statement of the cardholder
- `account_validation` - Indicates whether the card transaction should be processed as Account Validation request instead. If set to true, the amount provided for the Checkout would be ignored and the transaction be processed with amount: 0.
- `card_capture_mode` - Method of capturing card data. v2 is used by default as the recommended method. v1 is considered deprecated and is not supported for new integrations.
  - `stored_credential` and `card_capture_mode` - are only supported in v2
  - The fraud protection service as this feature is only supported in v2
- `capture_now` - Used for separate authorization and capture

For an account created on the NZ environment the capture_now value should only be set as true because of regional restrictions.

- `cvv_required` - Used to make CVV input field mandatory on the payment form. Defaults to false.
- `authorization_type` - Used to indicate what type of authorization is done. Support for this feature varies based on the Payment Contract used.
- `shopper_interaction` - Indicates the channels used by the shopper to send the card data for transactions:
  - ECOMMERCE
- `payment_contract_id` - This parameter can be found in the portal on the Payment Provider Contract attached to your organization or given to you by a Verifone employee. It is used to retrieve MID and other merchant configurations necessary for a card payment. (**Required**)
  - if different payment contract IDs are created and separated per card brand, then individual card brands can be listed under this parameter to include them into the checkout logic

```
"payment_contract_id": {
"VISA": "{{payment_contract_card}}",
"CB": "{{payment_contract_card}}",
"MASTERCARD": "{{payment_contract_card}}"
},
```

- `token_preference` - Read ***Tokenization*** for additional information
    - token_scope - ID taken from the Linked token scope section under the organization in Verifone Central, after a token scope was created
    - token_expiry_date - allows the limitation of the expiry date for the token created
    - token_type - type of token created
        - REUSE
        - REUSE_AND_ANALYTICS
        - ANALYTICS
- `threed_secure` - Used for *3-D Secure payments* . 3DS contract ID is validated against entity ID in checkout API.
- `credit_term` - Determines the transaction type. Only applicable for ABS Acquirer.
    - STANDARD - Sale
    - PREFERRED - Sale - Preferred Isracard
    - DEBIT - Sale - Direct Debit
    - INSTALMENT_STANDARD - Instalment standard
    - INSTALMENT_CREDIT - Instalment credit
- `instalment` - The details on the instalment scheme that should be enacted by the issuer. Only applicable for ABS Acquirer.
- `stored_credentials` - Used for transactions with ***Stored Credentials***
- `additional_business_data` - Merchant defined additional data fields for ABS Acquirer
- `reuse_token` - Identifier used to represent the cardholder data
- `fraud_protection_score` - Identifier used to submit Verifone Fraud Protection ID. Usable with mode PAYMENT.

Authorization and capture

Checkout can be used to do a sale (capture_now = true) or to authorize without capturing immediately (capture_now = false). An authorized payment reserves the money in the cardholder's account and allows you to capture the funds at a later stage. Authorizing the payment can be done using Checkout while capturing the payment is done through a separate API call or through the portal.

You cannot capture more or less than has been authorized by the cardholder per transaction.

Example: If a payment has been authorized for 20.00 EUR, it is only possible to perform a full capture for 20.00 EUR.

Requirement

Set the `configurations.card.capture_now` value to `false`. Setting it to `true` will immediately capture the payment.

> Captures can only be done on transactions with the status AUTHORIZED.

After you have used Checkout to authorize a payment, you will receive the ID of the transaction in the `transaction_id` query string parameter appended to the `return_url`. This ID needs to be referenced for ***capturing the payment*.**

Account verification

Sometimes, a merchant wants to ensure that the customer's card details are valid and can be used to make a payment, without making the actual payment at the time. This can be done via a transaction referred to as an Account validation.

Account validation transactions do not have an amount, so there is no money being blocked or moved from the customer's card. However, processing such a transaction tells the merchant whether their card is valid or not, but can also confirm CVV validity and AVS information if your acquirer supports this feature. Account verification is also known as zero-value transaction.

Set up

Checkout pages can be set up for processing transactions. This is done by providing the `amount` of the Checkout as `0`. If `configurations.card.capture_now` as `true` is provided, it will be overridden with `false` because it is not possible to capture an account verification payment.

3-D Secure

Account verification can be used in combination with 3-D Secure. Follow the steps in the ***Accepting 3-D Secure payments*** document to configure Checkout for 3-D Secure, then set the `amount` to `0` (zero). No money will be reserved on the cardholder's account and authentication will be done.

Strong Customer Authentication (SCA)

Strong Customer Authentication (SCA) is a European regulatory requirement to reduce fraud and make online and contactless offline payments more secure. We support SCA that applies to customer-initiated online and contactless offline payments within Europe.

Find out more on SCA in our eCommerce payments section and in-store payments section.

Handling responses

Whenever a card payment is processed via the Checkout, the responses events would contain additional fields specific to card payments in the **details** object.

Example of successful checkout via the Checkout:


```
[
{
"type": "TRANSACTION_SUCCESS",
"id": "f2041250-4fc2-4b3a-bc94-651ba099541a",
"timestamp": "2020-07-08T12:42:37.974Z",
"details": {
"id": "927bdeb4-afb9-44ff-9bf2-6348e2080c82",
"payment_provider_contract": "8d3c506b-5585-4c1c-8530-2319237f6385",
"amount": 12322,
"blocked": false,
"customer": null,
"merchant_reference": "ORDER-9633",
"payment_product": "CARD",
"status": "AUTHORIZED",
"authorization_code": "5669 ",
"created_by": "ffa1ac64-d04c-4af1-9e71-c7aad3c854d5",
```

```
"cvv_result": "0",
"details": {
"auto_capture": true
},
"reason_code": "0000",
"rrn": "ORDER-9633",
"shopper_interaction": "ECOMMERCE",
"stan": "041796",
"reversal_status": "NONE",
"geo_location": [
51.9336,
4.4888
],
"city": "Rotterdam",
"country_code": "NLD",
"additional_data": {
"acquirer_response_code": "00",
"initiator_trace_id": "041796"
}
}
}
]
```


In case v1 is used as card capture mode, the response will contain also the event and card config information below.

Event

- "type": "CARD_TOKEN_SUCCESS",
- "id": "c0e894c6-7a1f-4af7-9a11-2c34b2731649",
- "timestamp": "2022-03-21T08:11:05.559Z"

Card config information

- "card_expiry_month": "XX",
- "card_expiry_year": "XX",
- "card_last_four": "XXXX"

Example of failed card payment via Checkout:


```
[
{
"type": "TRANSACTION_FAILED",
"id": "f2041250-4fc2-4b3a-bc94-651ba099541a",
"timestamp": "2020-07-08T12:42:37.974Z",
"details": {
"id": "927bdeb4-afb9-44ff-9bf2-6348e2080c82",
"payment_provider_contract": "8d3c506b-5585-4c1c-8530-2319237f6385",
"amount": 12322,
"blocked": false,
"customer": null,
"merchant_reference": "ORDER-9633",
"payment_product": "CARD",
"status": "FAILED",
"created_by": "ffa1ac64-d04c-4af1-9e71-c7aad3c854d5",
"cvv_result": "0",
"details": {
"auto_capture": true
},
"shopper_interaction": "ECOMMERCE",
```

```
"stan": "041796",
"reversal_status": "NONE",
"geo_location": [
51.9336,
4.4888
],
"city": "Rotterdam",
"country_code": "NLD",
"additional_data": {
"acquirer_response_code": "00",
"initiator_trace_id": "041796"
}
}
}
]
```


When a transaction has been declined, the `events` would look like this:


```
[
{
"type": "TRANSACTION_DECLINED",
"id": "f2041250-4fc2-4b3a-bc94-651ba099541a",
"timestamp": "2020-07-08T12:42:37.974Z",
"details": {
"id": "927bdeb4-afb9-44ff-9bf2-6348e2080c82",
"payment_provider_contract": "8d3c506b-5585-4c1c-8530-2319237f6385",
"amount": 12322,
"blocked": false,
"customer": null,
"merchant_reference": "ORDER-9633",
"payment_product": "CARD",
"status": "DECLINED",
"created_by": "ffa1ac64-d04c-4af1-9e71-c7aad3c854d5",
"cvv_result": "0",
"details": {
"auto_capture": true
},
"shopper_interaction": "ECOMMERCE",
"stan": "041796",
"reversal_status": "NONE",
"geo_location": [
51.9336,
4.4888
],
"city": "Rotterdam",
"country_code": "NLD",
"additional_data": {
"acquirer_response_code": "05",
"initiator_trace_id": "041796"
}
}
}
]
```


- **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.