verifone

Your first transaction

This tutorial takes you through the steps required to do your first transcation through the API with Verifone. At the end of the tutorial you will have basic understanding of how the Verifone API works and how to perform transactions.

Resources

Before we get started here is a list of valuable resources for developing with Verifone you should add to your bookmarks:

- Verifone integration documentation - Where you are now
- Verifone API reference - An up-to-date reference of the API
- Verifone portal
- devdocs.io - Documentation for various development frameworks
- Mozilla Developer Network (MDN) Web Docs

Step 1: Set up

First, you need to be set up in Verifone. You'll need an API key, organisation ID and an account ID to get started.

API key

Log into the portal and click on your name in the top right of the application. Go the "API keys" section, if you don't see an API key or option to create one reach out to our Verifone contact and confirm you have the right permissions.

If you see your API key; click on the edit button, copy the full key from the next page and store it somewhere safe. Alternatively, create one with a date in the future and follow the same process.

Organisation ID

You can search for your organisation ID by navigating to the organisations page and clicking on your organisation. On the detail page you will see the organisation ID at the top or you can copy it directly from the URL. Store this ID somewhere safe.

Account ID

Go the accounts page and click on your account. On the detail page you will see the ID at the top or you can grab it in the URL. Store this ID somewhere safe.

Step 2: Tokenizing a card

We have our essentials now and we will start with our first API call. For our first API call we will be tokenizing a card. We will use this token later on in the tutorial to initiate the transaction.

JSON

The API accepts and responds with JSON, therefore we need to set the 'Content-Type' header of your API call to 'application/json'.

Formatting the API call

First we need to verify what the correct format is for the createCard call. We can find that information in the API reference. Copy the JSON object beneath this paragraph for this tutorial. Replace `organisation_id` with your own organisation ID from step 1. The other values can be copied as-is. `4000000000001000` is a Visa test card we will be using for this tutorial. Any `cvv` will be accepted as long as it is 3 digits long. Any `expiry_month` and `expiry_year` will be accepted as long as they are in the future.

```
{
"card_number": "4000000000001000",
"cvv": "123",
"expiry_month": "01",
"expiry_year": "23",
"organisation": "replace_this_with_your_own_organisation_id"
}
```

POST the body to `https://sandbox.omni.verifone.cloud/v1/card` and await the response with status code `200`:

```
{
"_id": "113321451149632557",
"bin": 400000,
"brand": "Visa",
"cvv_verified": true,
"expiry_month": 1,
"expiry_year": 22,
"issuer_country": "US",
"issuer_name": "",
"last_four": "1000",
"organisation": "organisation_id",
"currency": "USD",
"prepaid": false,
"type": "",
"variant": "",
"updated_at": "2019-12-10T13:31:55.000Z",
"created_at": "2019-10-14T13:46:00.059Z",
"token": "113321451149632557"
}
```

We have succesfully tokenized the card! As you can see we return quite some details for the card. The most important parameter we are looking for is the `_id` parameter. This contains the token we need to create a transaction. Store this somewhere safe.

Step 3: Creating a transaction

In this section we will use the API key and account ID from step 1 and the token from step 2. After this section we will have successfully completed a transaction and we can view it in the portal.

Authentication

To create transactions you will need to set your API key from step 1 in a customer header called `X-APIKEY`. The value must be the API key. Again, set the `Content-Type` to `application/json`.

Formatting the API call

Again, first we need to format what the correct format is for the createCardTransaction API call. Let's go to the API reference and navigate to the right endpoint. We'll only grab the required fields and ignore the rest for now:

- `account` - Set this to the ID from step 1.
- `amount` - Set this to 1234, the currency and therefore the decimal location is determined by the account. For a Euro account this will translate to €12,34.

- `card` - Use the ID from step 2 here.
- `customer_ip` - You can use your own IP address here or copy this one in: `127.0.0.1`
- `dynamic_descriptor` - The value in this parameter will show up on the cardholders bank statement. Use default value `Verifone Merchant` here.
- `merchant_reference` - This is a field that you can use to link the transaction to an internal ID, for now set it VF-001.
- `user_agent` - You can use your own user agent here or copy this one in: `Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0`

```
{
"account": "account_id",
"amount": 1234,
"card": "card_id",
"customer_ip": "127.0.0.1",
"dynamic_descriptor": "Verifone Merchant",
"merchant_reference": "VF-001",
"user_agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0"
}
```

POST the body to `https://sandbox.omni.verifone.cloud/v1/transaction` and await the response with status code `200`:

```
{
"reason_code": "00",
"processor": "dummy",
"account": "5d2efcf2628e0432e2826c6a",
"customer_ip": "127.0.0.1",
"dynamic_descriptor": "Verifone Merchant",
"user_agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0",
"merchant_reference": "VF-001",
"amount": 1234,
"details": {
"auto_capture": true
},
"card": "113321451149632557",
"cvv_present": true,
"payment_product": "card",
"payment_product_type": "Visa",
"_id": "5df106bb49a05a39a1a502aa",
"shopper_interaction": "ecommerce",
"fraud_predictions": [],
"status": "SETTLEMENT_REQUESTED",
"last_status_update": "2019-12-11T15:09:47.774Z",
"geo_location": [],
"payment_method_fees": [],
"fees": [],
"created_at": "2019-12-11T15:09:47.758Z",
"blocked": false,
"actions": [
"authorize"
]
}
```

Congratulations! You can see that the transaction was created successfully by looking at the `status` parameter. It says `SETTLEMENT_REQUESTED`, which is one of the states where we consider a transaction successful. You can view a full state diagram here.