Testing Stripe API endpoints with Postman

Introduction

Testing objectives

This guide demonstrates how to test Stripe's payment endpoints using Postman.

The following are the Stripe API endpoint testing objectives:

- Verify that all API endpoints function as documented
- Validate that request/response formats match specifications
- Confirm that error scenarios produce expected responses
- Test authentication mechanisms

Types of testing

Overview of tests covered in this guide.

Note: These instructions have been tested on macOS. Other operating systems may not be fully supported.

Authentication

Verify that valid API keys are accepted and invalid keys are rejected.

Creating a Payment Intent

A Payment Intent is the core object for managing the complete payment process in Stripe. Each Payment Intent represents a single payment attempt and automatically handles authentication requirements and payment method variations.

Retrieving a Payment Intent

Once a Payment Intent is created, Stripe assigns it an ID. You can test that you can perform a GET request to retrieve the ID.

Customers

A Customer object is a representation of a customer in your Stripe account. You can store multiple payment methods, shipping addresses, and more on the Customer object, and all fields are optional.

Currency Handling

Stripe uses ISO currency codes and amounts specified in the smallest currency unit, for example, cents for `usd`.

Configuration

Get a Stripe secret API key

Get a Stripe secret API key for testing purposes.

Before you can test the Stripe API, get a secret API key from your Stripe account.

1. Sign up for Stripe if you do not already have an account.
   Go to https://dashboard.stripe.com/register.
2. Skip the business information questions.
3. Select **Go to sandbox**.
4. Select **Got it** under **Verify your business**.
5. Select the **Secret key** to copy it.
6. Store the copied secret key in a secure location.

Configure Postman

Set up Postman for testing Stripe API endpoints.

You must have a Stripe secret API key before configuring Postman.

1. Download and install Postman.
   Navigate to https://www.postman.com/downloads/.
2. Store your secret key securely.
   a) Go to **Vault** on the bottom menu.
   b) Enter a **Key** name, for example `STRIPE_SECRET_KEY`.
   c) Enter the Stripe secret API key in **Value**.
   d) Leave **Allowed domains** blank.

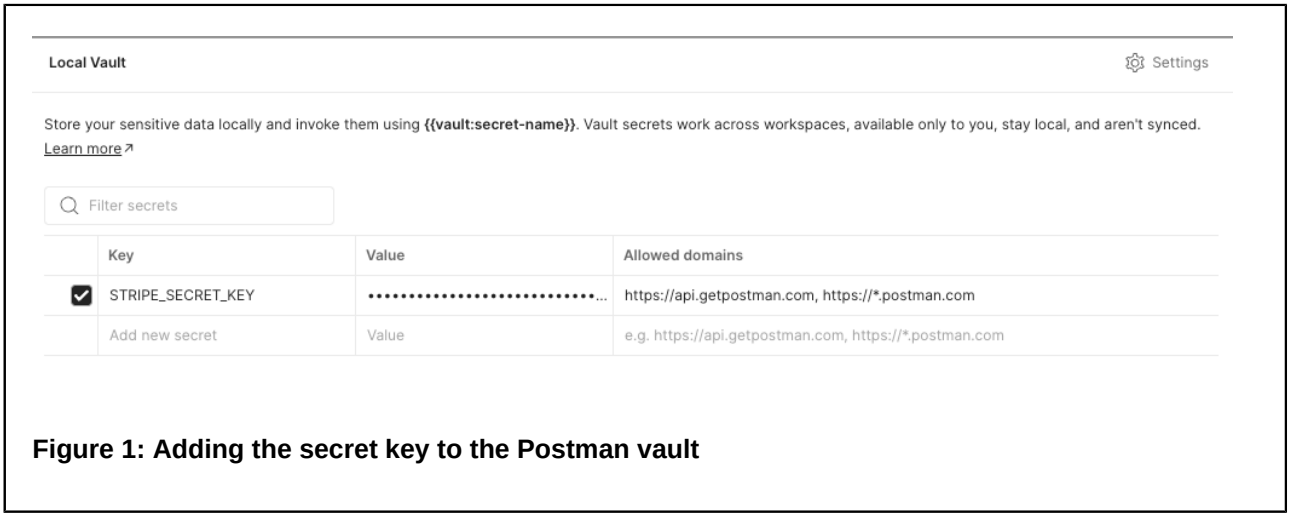

Figure 1: Adding the secret key to the Postman vault

   Your secret key is now stored securely and can be referenced in the format `{{vault:KEY_NAME}}`, where *KEY_NAME* is your Key name. For example, `{{vault:STRIPE_SECRET_KEY}}`.
3. Configure collection authorization.

a) Select **New** and then select **Collection**. Rename if desired, for example, `Stripe API testing`.
b) Select your collection and then select **Auth**.
c) Set **Type** to Basic Auth.
d) Set **Username** to `{{vault:STRIPE_SECRET_KEY}}` to access the key stored in your vault.
e) Leave **Password** blank.

4. Set the base URL variable.
   a) Select your collection and then select **Variables**.
   b) Enter a **Variable** named `base_url`.
   c) Set **Value** to `https://api.stripe.com/v1`.

Testing

Test authentication

Create a Postman request to verify that your Stripe API key is working correctly.

Testing authentication verifies that your API key is valid and properly configured. This test retrieves your account balance, which requires authentication.

1. Create a new request in your collection.
   a) Select your Stripe API testing collection.
   b) Select **Add a request**.
   c) Name the request `Authentication`. This step is optional, but helpful to keep requests organized.
2. Configure the request method and URL.
   a) Set the HTTP method to GET.
   b) Enter the URL: `{{base_url}}/balance`

   The balance endpoint requires authentication but no additional parameters.
3. Add test scripts to validate the response.
   a) Select the **Scripts** tab.
   b) Enter the following JavaScript to test for authentication:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains balance object", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.object).to.eql("balance");
});

pm.test("API key is valid", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.available).to.exist;
    pm.expect(jsonData.pending).to.exist;
});
```

4. Send the request and verify the results.
   a) Select **Send**.
   b) Wait for the response to appear.
   c) Select the **Test Results** tab.
   d) Confirm that all tests pass.

   If all tests pass, your authentication is configured correctly. If tests fail, refer to the Troubleshooting section.

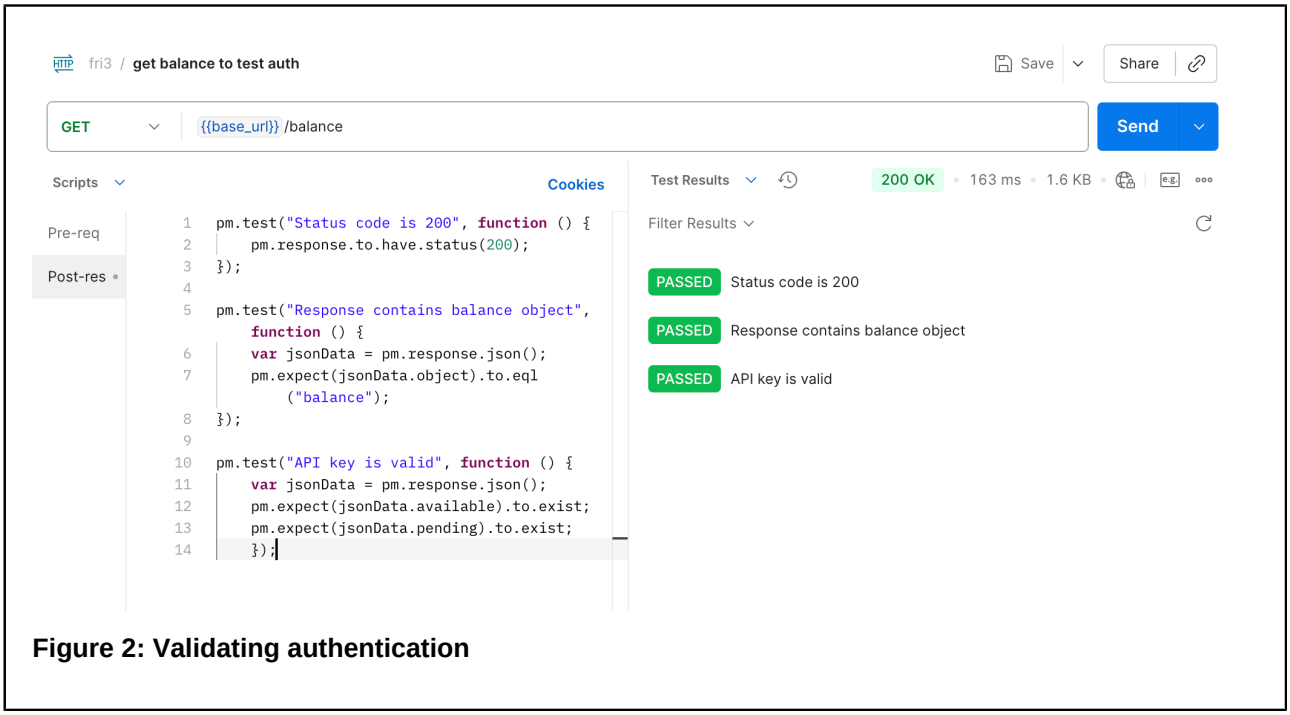

Figure 2: Validating authentication

You have successfully tested authentication with the Stripe API. The response will show your account balance in various currencies.

Create Payment Intent

Create a Postman request to generate a new Payment Intent in Stripe.

A Payment Intent guides you through the process of collecting a payment from your customer. Creating a Payment Intent is the first step in accepting a payment.

1. Create a new request in your collection.
   a) Select **Add a request**.
   b) Name the request `Create Payment Intent`.
2. Configure the request method and URL.
   a) Set the HTTP method to POST.
   b) Enter the URL: `{{base_url}}/payment_intents`
3. Add the request body parameters.
   a) Select the **Body** tab on the request pane.
   b) Select x-www-form-urlencoded.
   c) Enter the following example data:

Key	Value
amount	5000
currency	usd

The amount is in cents (the smallest currency unit), so 5000 represents $50.00.

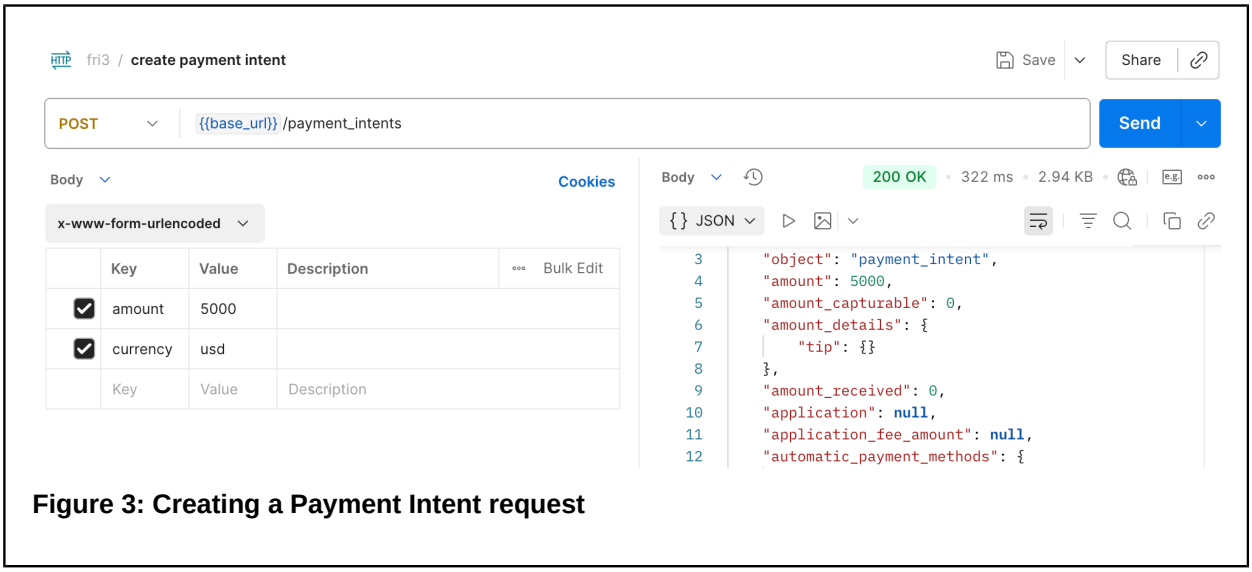

Figure 3: Creating a Payment Intent request

4. Add test scripts to validate the response.
   a) Select the **Scripts** tab on the request pane.
   b) Select Post-res.
   c) Enter the following JavaScript to test creation of the Payment Intent endpoint:

```
// Test for Create Payment Intent endpoint
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains payment_intent object", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.object).to.eql("payment_intent");
});

pm.test("Amount is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.amount).to.eql(5000);// This should match the
 amount set in the body
});

pm.test("Currency is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.currency).to.eql("usd");// This should match the
 currency set in the body
});

// Save the payment intent ID for subsequent tests
pm.environment.set("payment_intent_id", pm.response.json().id);
```

   The last line saves the Payment Intent ID to an environment variable so you can use it in other requests.

5. Send the request and verify the results.
   a) Select **Send**.
   b) Wait for the response to appear.
   c) Select the **Test Results** tab on the response pane.
   d) Confirm that all tests pass.

   If all tests pass, You have successfully created a Payment Intent. If tests fail, refer to the Troubleshooting section.

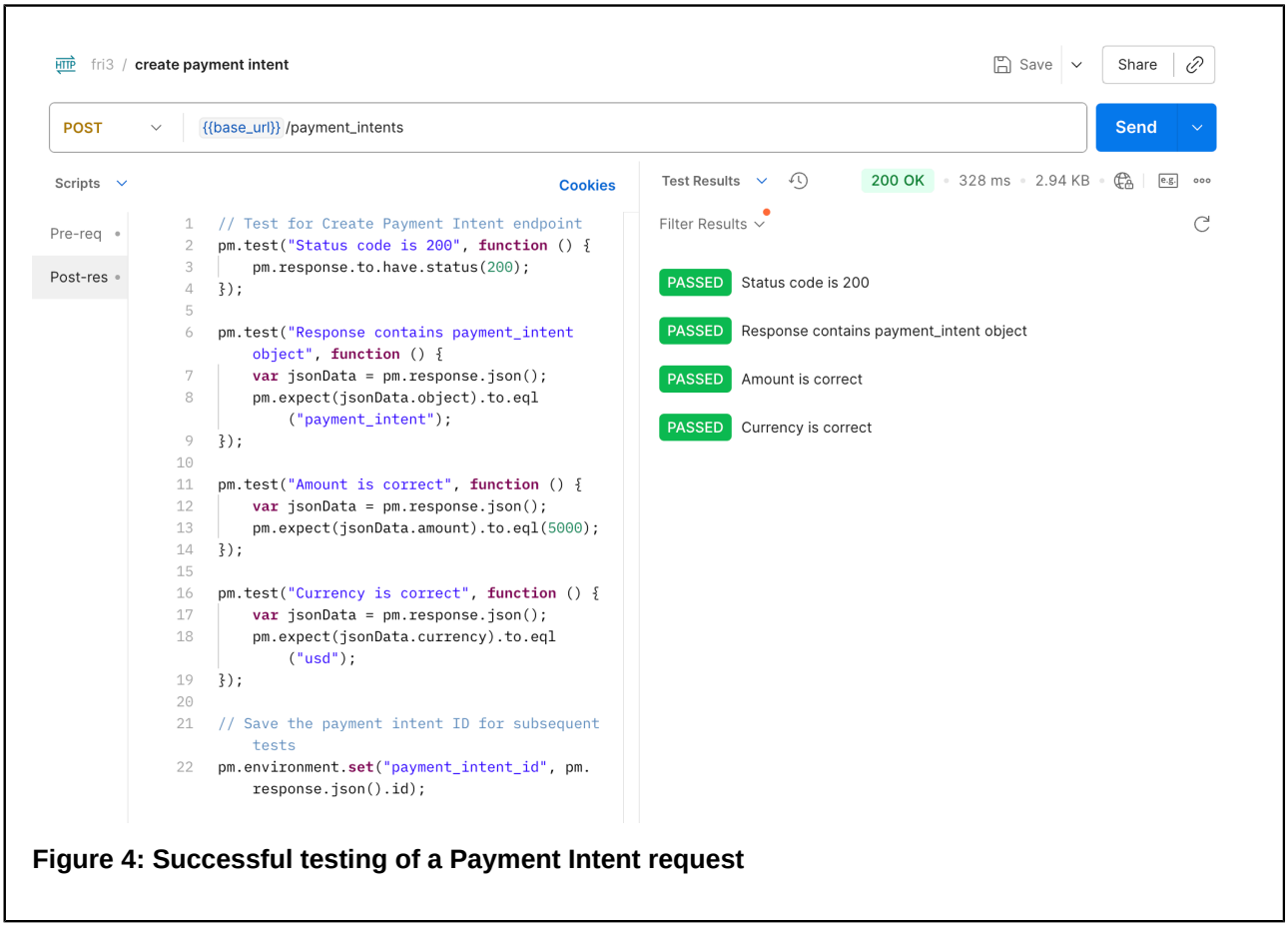

Figure 4: Successful testing of a Payment Intent request

You have successfully created a Payment Intent. The response contains a unique ID that you can use to continue the payment process.

Troubleshooting

Authentication fails

Resolve authentication errors when testing Stripe API endpoints.

Symptoms

All requests return 401 Unauthorized.

Possible Causes

Authentication failures can occur due to incorrect API key configuration or account issues.

Solutions

1. Verify you're using test mode key that starts with `sk_test_`.
2. Check for extra spaces in the API key.
3. Ensure colon (`:`) is present after key in cURL.
   Example: `-u sk_test_key:`
4. Verify Stripe account is active.

Payment Intent creation fails

Resolve errors when creating Payment Intent objects.

Symptoms

400 Bad Request when creating Payment Intent.

Possible Causes

Payment Intent creation can fail due to incorrect parameter formatting or missing required fields.

Solutions

1. Verify amount is a positive integer.
2. Check currency is a valid 3-letter ISO code.
   For example: `usd`, `eur`, `jpy`.

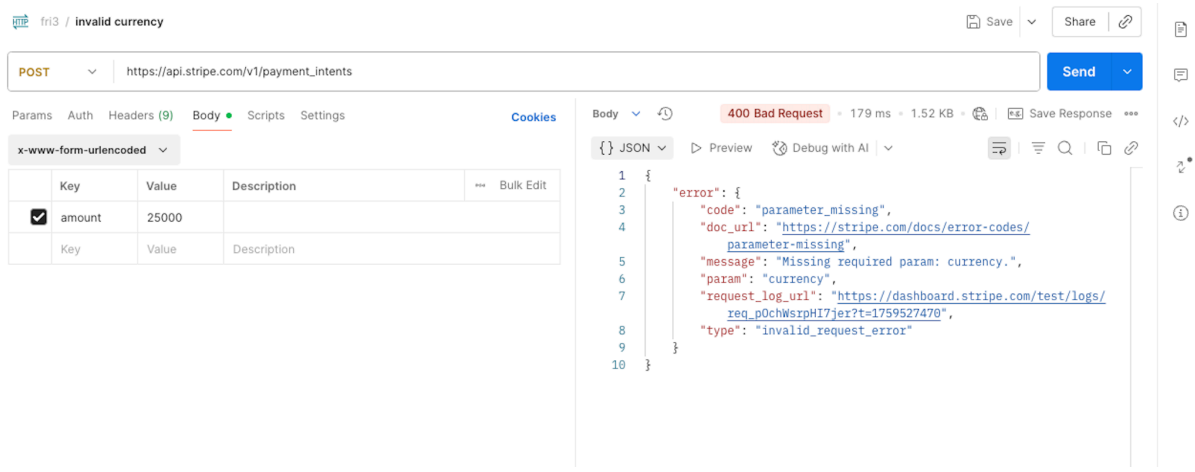

Figure 5: Failure due to missing required currency