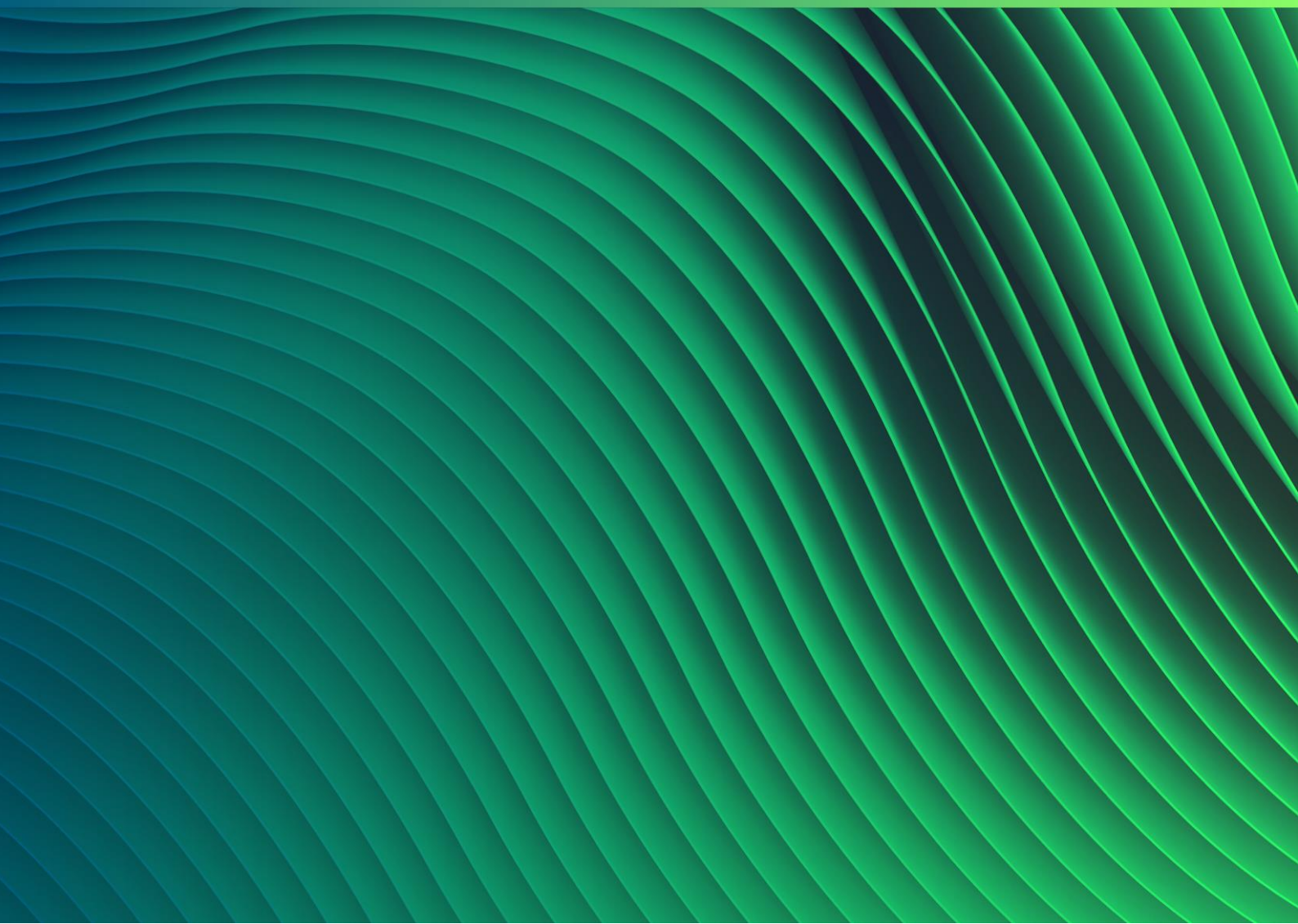
worldpay
from FIS

Salesforce SFRA testing guide for Access Worldpay

Version 21.1.0 – March 2021



Contents

Summary

This guide will help you to write test cases for all Worldpay services on the SFCC platform that use the RefArch Site reference application (the default SFCC ecommerce site). You can use the test cases to ensure all the main functionality operates correctly.

It also includes references to automation test suites for functional testing, integration testing and unit testing. These test suites enable you to verify your implementation.

Finally, the guide includes instructions for using the CodeceptJS testing automation suite for acceptance testing.

Test cases

This cartridge has successfully passed through QA and UAT. The test cases referred to below will help you implement the functional and integration test cases. See the Excel files in the *Consolidated_SFCC_AWP_TestCases* folder at <*Workspce*>*\link_access_worldpay\test\testCases.*

Test data

Test data is in the file *SFCC_AWP_TestData.xlsx* at <*Workspce*>*\link_access_worldpay\test\testData.*

Automation test suite for functional testing

Integration testing

Prerequisites to run Integration tests:

1. Make sure the dw.json file is present in the root of the project. Integration tests would use the sandbox URL given in dw.json file.
2. Integrations are currently running against RefArch site, make sure to create a storefront customer account from MyAccount with the following test credentials. These are test credentials, used for testing purpose only.

```
username: 'test@worldpay.com'
password: 'Worldpay@123'
```

3. Once the customer is created, make sure to add a credit card in the MyAccount section.
4. Make sure to have the following OCAPI settings are present for RefArch site context.

```
{
  "_v":"19.1",
  "clients":
  [
    {
      "client_id":"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
      "allowed_origins":[],
      "resources":
      [
        {
          "resource_id":"/customers/auth",
          "methods":["post"],
          "read_attributes":"(**)",
          "write_attributes":"(**)"
        },
            {
          "resource_id":"/customers/*/payment_instruments",
          "methods":["get","post"],
          "read_attributes":"(**)",
          "write_attributes":"(**)"
        }
      ]
    }
  ]
}
```

To run integration testing:

1. Open the terminal and run the command `npm run test:integration` to run the test and integration files one by one. You can see the status (success or failure) of each test script in the console.

The test scenarios are:

- guestCreditCardPaymentMethod
- loginDirectCreditCardPaymentMethod

- loginSavedCardPaymentMethod

Automation test suite (CodeceptJs)

To configure and run automated acceptance testing on CodeceptJs, ensure that the *selenium-standalone* node package is installed on your machine.

Then do the following:

1. Add Codecept and update the selenium standalone dependencies in package.json to version 5.8.0 or above as shown below:

"@wdio/selenium-standalone-service": "^5.8.0", "codeceptjs": "^2.1.0"

2. To run acceptance testing, run the command `npm run test:acceptance` on the terminal.

The CodeceptJS configuration is set in codecept.conf.js file. The configuration file looks like this:

```
var RELATIVE_PATH = './test/acceptance';

var OUTPUT_PATH = RELATIVE_PATH + '/report';

var HOST = '<host url>';

var webDriver = {

    url: HOST,

    browser: 'chrome',

    //desiredCapabilities: {

        chromeOptions: {

          args: [ "--headless", "--disable-gpu", "--no-sandbox", "--window-
size=1920,1080" ]

        }

    },//

  smartWait: 5000,

    waitForTimeout: 5000,

    windowSize: 'maximize',

    timeouts: {

        script: 60000,

        'page load': 10000
```

```
    }
};


exports.config = {
    output: OUTPUT_PATH,
    helpers: {
        WebDriver: webDriver
    },
    plugins: {
        wdio: {
            enabled: true,
            services: ['selenium-standalone']
        },
        allure: {
            enabled: true
        },
        retryFailedStep: {
            enabled: true,
            retries: 1
        }
    },
    include: {
        homePage: RELATIVE_PATH + '/pages/HomePage.js',
        /*accountPage: RELATIVE_PATH,*/
        worldpayPaymentTestRegistered: RELATIVE_PATH +
'/pages/worldpayPaymentTestRegistered.js',
        worldpayPaymentTestRegisteredcredit: RELATIVE_PATH +
'/pages/worldpayPaymentTestRegisteredcredit.js',
        uriUtils: RELATIVE_PATH + '/utils/uriUtils.js'
```

```
    },
    gherkin: {
        features: RELATIVE_PATH + '/features/PaymentMethods/**/*.feature',
        steps: [
            RELATIVE_PATH + '/features/steps/wpPaymentRegisteredApm.steps.js',
            RELATIVE_PATH + '/features/steps/wpPaymentRegisteredcredit.steps.js'
        ]
    },
    tests: RELATIVE_PATH + '/tests/**/*.test.js',
    name: 'link_worldpay'
};
```

Features

A feature describes the current test script as executed. Every *.feature file conventionally consists of a single feature. Lines starting with the keyword **Feature:** start a feature. A feature usually contains a list of scenarios, for example the `RegisteredCCDirectAuthorised.feature`.

Scenarios

A scenario describes the steps and expected outcome of a specific test case. Every scenario starts with the Scenario: keyword. Each feature can have one or more scenarios, and every scenario consists of one or more steps. See the screenshot below.

```
@DirectOrder_MasterCard
   Scenario: Registered User is able to place order via Master Card in Direct Method with Authorised Status
      Given shopper selects yes or no for tracking consent
      Then Shopper click on login button displaying on left header side
      And Shopper fills the correct login details and click on Login Button
         |email|password|
         |code1auto1@yopmail.com|Test@123|
      Then Shopper searches for "Shirt"
      Then Selects size "15R"
      Then User add the product to cart and click to Checkout
      Then Verify that user has navigated to Shipping Page
      Then Select Country "UnitedStates"
      Then Select State "California"
      And Fill the Shipping address
         |firstName|lastName|streetAddress1|streetAddress2|city|postalCode|phoneNumber|
         |code|auto|27 RUE PASTEUR|52 RUE DES FLEURS|CABOURG|14390|(33) 1 43 12 48 65|
      And User fills email and phone number and click on Add Payment Button
         |email|password|
         |code1auto1@yopmail.com|(33) 1 43 12 48 65|
      Then User add a new Master card details
         |Email|Phone Number |Name on Card |Card Number |Expiration Month|Expiration Year|Security Code|
         |code1auto1@yopmail.com|3333333333|code auto|5454545454545454|02|2022|545|
      Then Verify that added card should be Master Card
      Then User Click on Next Place Order Button
      Then User Click on Place Order
      Then User Click on Challenge OK Button
      And Print the Order Number
```

Steps

Features consist of steps, also known as Givens, Whens and Thens. Let's look at them in detail:

- Given: Specifies the context of the text to be executed
- When: Specifies the test action that must be done
- Then: Represents the expected outcome of the test.

Testing starts by running the features mentioned in Gherkin (codecept.conf.js). It then looks for the steps to be executed. See the code example below:

```
include: {

        homePage: RELATIVE_PATH + '/pages/HomePage.js',

        /*accountPage: RELATIVE_PATH,*/

        worldpayPaymentTestRegistered: RELATIVE_PATH +
'/pages/worldpayPaymentTestRegistered.js',

        worldpayPaymentTestRegisteredcredit: RELATIVE_PATH +
'/pages/worldpayPaymentTestRegisteredcredit.js',

        uriUtils: RELATIVE_PATH + '/utils/uriUtils.js'

    },

    gherkin: {
```

```
        features: RELATIVE_PATH + '/features/PaymentMethods/**/*.feature',
        steps: [
            RELATIVE_PATH + '/features/steps/wpPaymentRegisteredApm.steps.js',
            RELATIVE_PATH + '/features/steps/wpPaymentRegisteredcredit.steps.js'
        ]
    },
```

You can see the test scenarios in SFCC_AWP_CodeceptJS_Automation_Scenarios.xlsx at <Workspce>\link_access_worldpay\test\testScenarios

Unit testing

For the unit test case libraries go to *<Workspce>\link_access_worldpay\test\unit*.

To run the unit tests, go to *<Workspce>\link_access_worldpay*.

Open the terminal and run the command `npm run test` to run the test/unit files.

Contact Us

Please contact your Worldpay Relationship Manager or Worldpay Support.

Tel: 0800 096 3997.