Optimize 25 Testing Automation: Setup Guide



Prepared by

**Hari Yarapotina**

Solution Architect

Contributors

**List of Contributors**

Revision and Signoff Sheet

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
|  | Hyarapotina | 1.0 | Initial draft for review/discussion |
|  |  |  |  |
|  |  |  |  |

Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Table of Contents

[1 Introduction 5](#_Toc131590365)

[1.1 Pre-requisites 5](#_Toc131590366)

[1.2 Installation 5](#_Toc131590367)

[1.3 Dynamics 365 – Trail Environment Setup 7](#_Toc131590368)

[1.4 Setting up Automation Test Project 8](#_Toc131590369)

[1.5 Running Tests 10](#_Toc131590370)

[1.6 Resources 10](#_Toc131590371)

1. Introduction

Automated end-to-end testing is an important process during Dynamics 365 implementations. Automation helps to test the code quickly by replicating the business process flows and deploying the code faster with high confidence. Historically, setting the automation testing in Dynamics 365 projects is hard and various frameworks were available like EasyRepo based on Open-Source Selenium web drivers.

In this guide, we are introducing Playwright for end-to-end automation testing with Dynamics 365. Playwright Library provides unified APIs for launching and interacting with browsers, while Playwright Test provides all this plus a fully managed end-to-end Test Runner and experience.

* 1. Pre-requisites

**Playwright:** An open-source library from Microsoft that is built for developers and testers to write end-to-end tests that run on all modern web browsers. Supports JavaScript, TypeScript, C# and Python.

* Native Installation:
  + <https://playwright.dev>
* [Playwright Test for VSCode - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-playwright.playwright)
  + Version – v1.19+ or newer
  + Playwright NPM Package

Playwright supports testing on Windows, Linux, and macOS, locally or on CI, headless or headed with native mobile emulation of Google Chrome for Android and Mobile Safari.

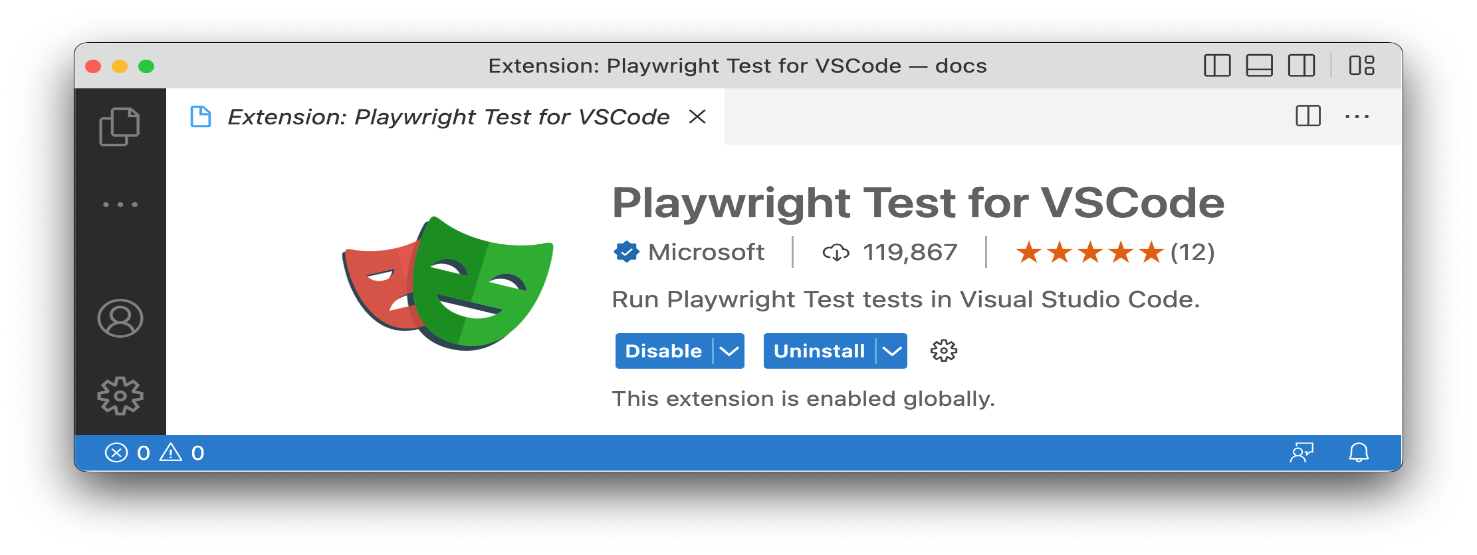
**Azure Subscription (optional):** TBR. An active Microsoft Azure subscription is recommended. This is a requirement if you plan to execute automated tests in Microsoft Azure or integrate load testing into your CI/CD pipelines.

* 1. Installation

Playwright can be installed in two ways i.e., natively using npm or yarn. Alternatively, you can also get started and run your tests using the VS Code Extension.

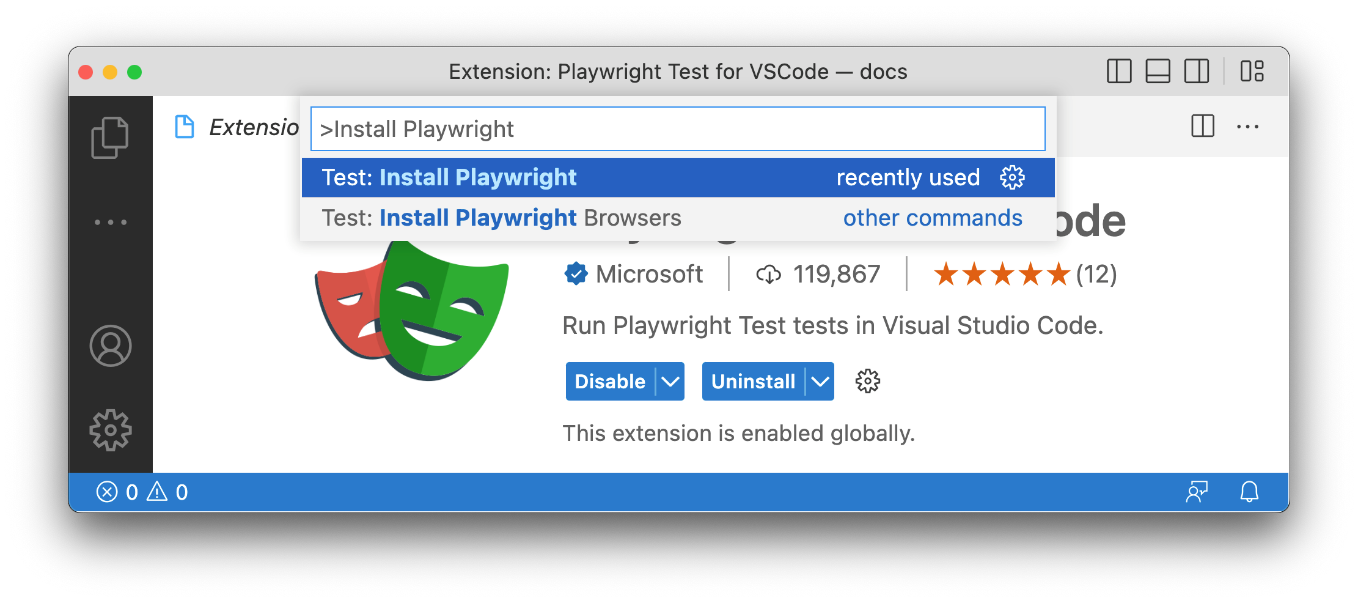
For Automation testing here, we will choose the second option i.e., Playwright VS Code extension.

1. Install the [VS Code extension from the marketplace](https://marketplace.visualstudio.com/items?itemName=ms-playwright.playwright) or from the extensions tab in VS Code.

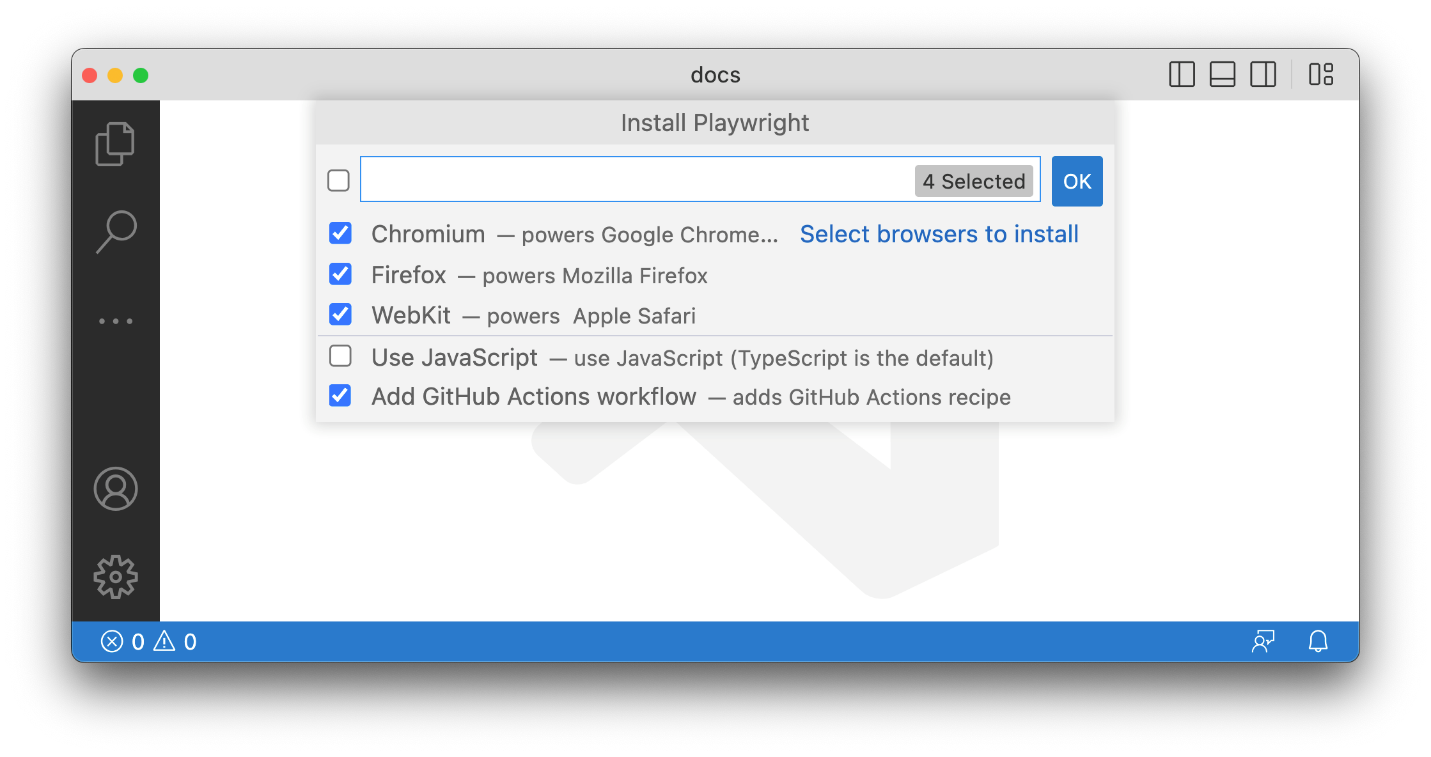


1. After installation, open the command panel and type:

*Install Playwright*



1. Select **Test: Install Playwright** and choose the browsers you would like to run your tests on. These can be later configured in the playwright.config file.



This will complete the installation needed for setting up the Automation Test project.

* 1. Dynamics 365 – Trail Environment Setup

1. Create a Trial environment by navigating to
   1. [Start a Free Trial for Microsoft Dynamics 365| Microsoft Dynamics 365](https://dynamics.microsoft.com/en-us/dynamics-365-free-trial/)
2. Select Dynamics 365 Customer Service under Service

Graphical user interface

Description automatically generated

This prompts to enter the “Email Address” to continue.

Graphical user interface, application

Description automatically generated

1. Enter your email address and start your feel trail. (Trial is valid for 30 days)
2. Customer Service environment will be created.
3. Turn off security defaults. For automation testing, it is
   1. [Security defaults and Conditional Access - Microsoft 365 Business Premium | Microsoft Learn](https://learn.microsoft.com/en-us/microsoft-365/business-premium/m365bp-conditional-access?view=o365-worldwide&tabs=secdefaults)
   2. [Introducing security defaults - Microsoft Community Hub](https://techcommunity.microsoft.com/t5/microsoft-entra-azure-ad-blog/introducing-security-defaults/ba-p/1061414)

1. Go to the Azure Active directory and add additional users as needed then assign security roles.
   1. At minimum, create a test automation account to use it with testing.
      1. test.automation@domain.com
   2. Setting up Automation Test Project

To facilitate the test automation in Dynamics 365 Implementations, FastTrack team has provided a sample automation project and below are the instructions to set it up. This project is based on Playwright test library.

Follow the steps below to set up a Test Automation:

1. **Download/Clone the sample Automation project** from the github repository link below.
   1. <https://microsoft-my.sharepoint.com/:u:/p/hyarapotina/EfVv6YdR3epJl0vrqZOOKC4B01h57SeMub0jgFa1waVeSw?e=L8wRJe>
2. **Extract the project folder** and open this **in Visual Studio Code**.
3. Here is the folder structure:

Text

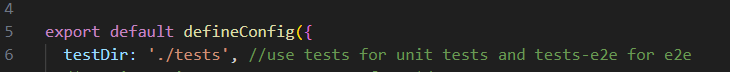
Description automatically generated

1. **Open the .env File and edit the file to include your environment details.** 
   1. *DYN365\_ORGURL='https://org212fbf02.crm.dynamics.com/'*
   2. *DYN365\_USERNAME=’<<<Replace with the Username>>>’*
   3. *DYN365\_PASSWORD='<<<Replace with the password'>>>’*
   4. *CSH\_APPID=’<<<Replace with the Customer Service Hub App Id>>>’*
   5. *CASE\_NUMBER=’<<<Replace with a Case Number that you already have>>>’ 🡪 This is only used if you are to run the unit tests. By default, the project setup to run the end-to-end tests in the tests-e2e folder.*
2. Please note that, this sample test project has end to end tests for the Case/Incident covering Case Create, Retrieve, Update and Resolve. These cases are based on the out of the box forms. Recommended to run this on a trial environment to test. Once you were able to configure and test this, you can switch the configuration for your org.
   1. Running Tests

By default, the sample project has a configuration setting pointing to end-to-end tests folder. The instructions below are to execute the end-to-end tests from the UI using VS Code. To execute the unit tests, you can change the setting “testDir” within playwright.config.ts file as needed.

For end-to-end tests the testDir must be ‘./tests-e2e’

For unit tests the testDir value must be ‘./tests’



End-To-End test executions (from VS Code UI):

1. From the Visual Studio, navigate to the Testing area on the left menu. Click to open “Test Explorer”

A screenshot of a computer

Description automatically generated with medium confidence

1. Test Explorer would display the folder structure and the end-to-end test cases as shown below.

A screenshot of a computer

Description automatically generated with medium confidence

1. To run the test cases you can click on the play button next to the casecrud.spec.ts file as shown below. By default, VS Code test runner runs your tests on the default browser of chrome.

Graphical user interface, text

Description automatically generated

1. To run on different browsers, click the play button on the top of the test explorer and choose another browser you wish to run the tests on.

Graphical user interface, text

Description automatically generated

1. After successful test executions, there will be a check mark next to each test. Graphical user interface, text

   Description automatically generated
   1. Resources

Playwright Overview - <https://playwright.dev/docs/intro>

Playwright with Visual Studio Code - <https://playwright.dev/docs/getting-started-vscode>