**Rx CC Test Automation Guide**

By Software Quality Automation

**Contents**

[1. Purpose 1](#_Toc94014908)

[1.1. Foundational test framework 1](#_Toc94014909)

[1.2. References 1](#_Toc94014910)

[2. Prerequisite 1](#_Toc94014911)

[3. Configuration and setup 1](#_Toc94014912)

[3.1. Code Repository 2](#_Toc94014913)

[3.2. Import to Eclipse 2](#_Toc94014914)

[4. Test Execution 5](#_Toc94014915)

[4.1. Eclipse IDE 5](#_Toc94014916)

[4.2. Maven 6](#_Toc94014917)

[4.3. Test Parameters 6](#_Toc94014918)

[5. Roles 6](#_Toc94014919)

[6. Test Tags 7](#_Toc94014920)

[7. Azure DevOps 8](#_Toc94014921)

[7.1. GIT Pull Request 8](#_Toc94014922)

[7.2. Pipeline 8](#_Toc94014923)

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Author** | **Description** | **Date** |
| G. Cosmiano | Initial version | 01/25/2022 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Purpose

The purpose of this document is to define how-to for test automation of Rx CC Intervention Web UI and API.

## Foundational test framework

The project will use the SQA utilities

1. [Selenium project starter](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/SeleniumProjectStarter_UserGuide.docx?d=w793cfa72d01f423b8d8fcf13e2143c2c&csf=1&web=1&e=dleO7q)
2. [JUnit reporter](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/JUnit%20Utility.docx?d=w79273e2e68cc447ea99b075ab2377ced&csf=1&web=1&e=o19rny)
3. [Workflow](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/Workflow%20Implementation.docx?d=wfce8decaeeac43c289fe679e4a7dccd2&csf=1&web=1&e=V8f3oh) / [Step](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/StepImplmentation.docx?d=w59edb26fc16145a282d7d4c2dd0bef74&csf=1&web=1&e=RJKdBm) process
4. [Data test object](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/Data%20Test%20Object.docx?d=w3f7af232ec304bfe9a1f91704e409d13&csf=1&web=1&e=jN5TSJ)

In addition, it will follow the best practices as follows

1. [Selenium and Java programming](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/Selenium%20and%20Java%20Programming%20Best%20Practices.docx?d=w69955b107ddd4f2694f36578860c3f0c&csf=1&web=1&e=oD6giD)
2. [Java coding best practices](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/A%20short%20summary%20of%20Java%20coding%20best%20practices.docx?d=w0222cc62693d4dc08b914844e456164c&csf=1&web=1&e=fHC0uY)
3. [Distributed version control](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Source%20Control/Distributed%20Version%20Control%20with%20Mercurial%20or%20Git.docx?d=w2135c2154b5f4144a2cc85d0595a7db6&csf=1&web=1&e=mA2KDH)
4. [Azure DevOps GIT](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Source%20Control/SQA%20-%20Azure%20DevOps%20GIT.docx?d=w76c9f8497a7740468cc160a4fef861f1&csf=1&web=1&e=oDBTw2)

## References

* [Product documentations](https://lifethc.sharepoint.com/sites/RxConciergeTeam_PHE_PRV)
* [Agile stories and tasks](https://azdoprd.excellus.com/DefaultCollection/EHP/_backlogs/backlog/Rx%20Concierge%20Team/Backlog%20items?showParents=true)
* [Automation test plan](https://lifethc.sharepoint.com/:w:/r/sites/RxConciergeTeam_PHE_PRV/Shared%20Documents/Testing/RxCC_Test_Automation_Plan.docx?d=wd2e2120e532c43e384ef15944d37c29e&csf=1&web=1&e=HPl2xo)

# Prerequisite

Install and setup Java, Eclipse and Maven using this [documentation](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/_layouts/15/Doc.aspx?sourcedoc=%7B7DEC1BBA-0B76-4223-841D-D47655A8ABB5%7D&file=PCS%20Local%20Software%20Test%20Automation%20Install%20Guide.docx&action=default&mobileredirect=true&cid=da0f1f1a-be1e-44f7-9fb0-72c474295f8b).

# Configuration and setup

The test automation will utilize maven parent-module to take advantage of sharing resources for Web UI and API. The project will be structured as follows

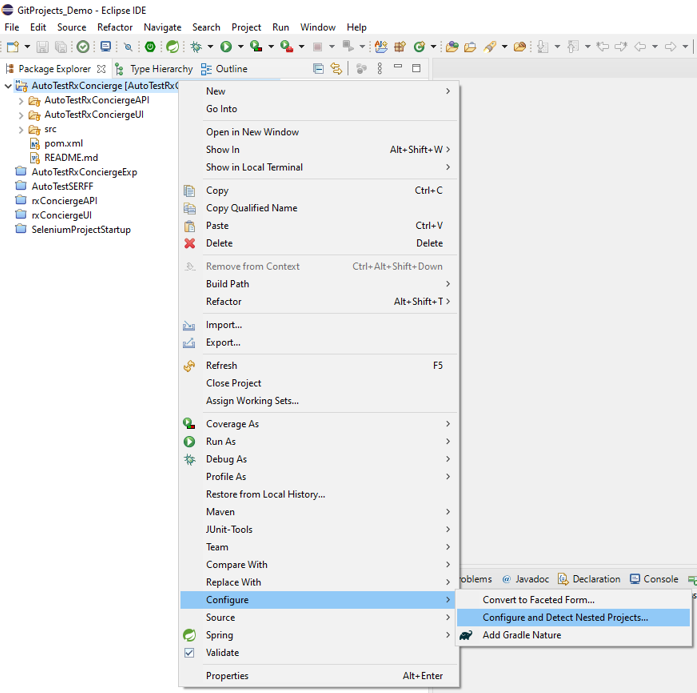


## Code Repository

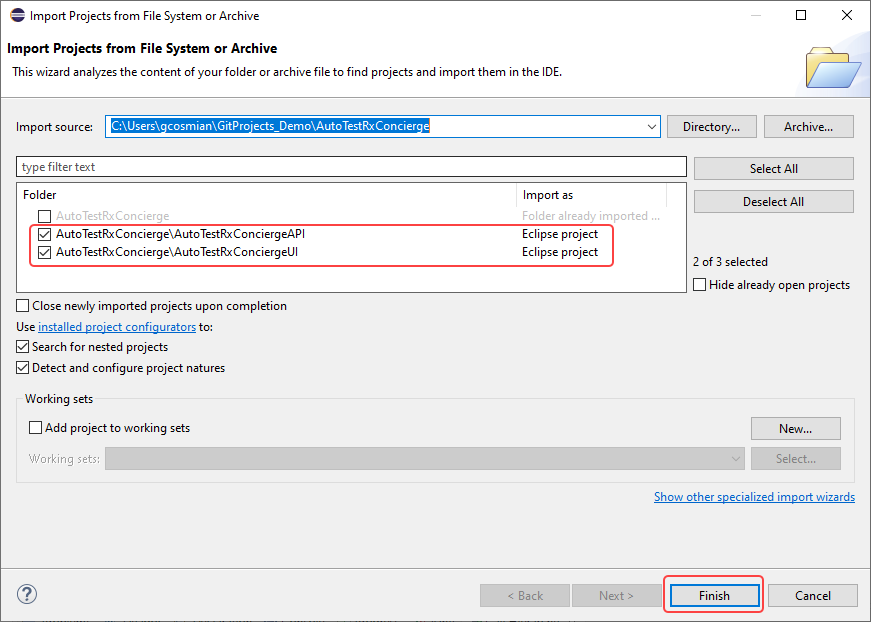
The code repository will be in Azure GIT, <https://azdoprd.excellus.com/DefaultCollection/EHP/_git/AutoTestRxConcierge>.

## Import to Eclipse

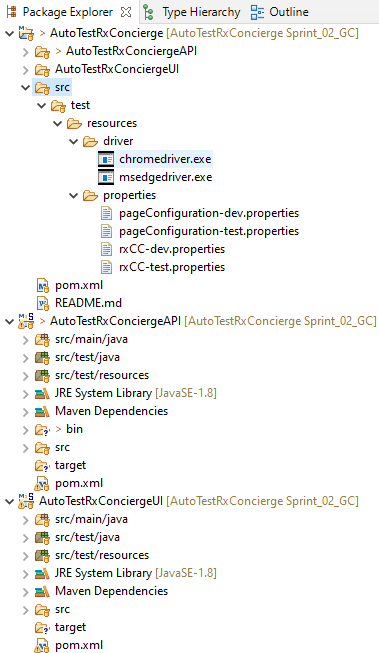
1. Follow the instruction in [SQA – Azure DevOps GIT](https://lifethc.sharepoint.com/:w:/r/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Source%20Control/SQA%20-%20Azure%20DevOps%20GIT.docx?d=w76c9f8497a7740468cc160a4fef861f1&csf=1&web=1&e=wdjKZc) section 4.3 to import the project and setup AutoTestRxConcierge as maven project.
2. Right click on the project AutoTestRxConcierge and select Configure > Configure and Detect Nested Projects.



1. Enable the projects AutoTestRxConciergeUI and AutoTestRxConciergeAPI then click Finish.



1. Make sure your project is setup as follows



# Test Execution

Test execution can be done either in IDE or using maven command line.

## Eclipse IDE

1. See <https://www.eclipse.org/community/eclipse_newsletter/2017/october/article5.php> for reference
2. Select the test to run, examples
   1. AutoTestRxConciergeUI - com.excellus.sqa.rxcc.tests.ApiTest
   2. AutoTestRxConciergeAPI - com.excellus.sqa.rxcc.tests.LoginTest
3. Run the test with these VM arguments

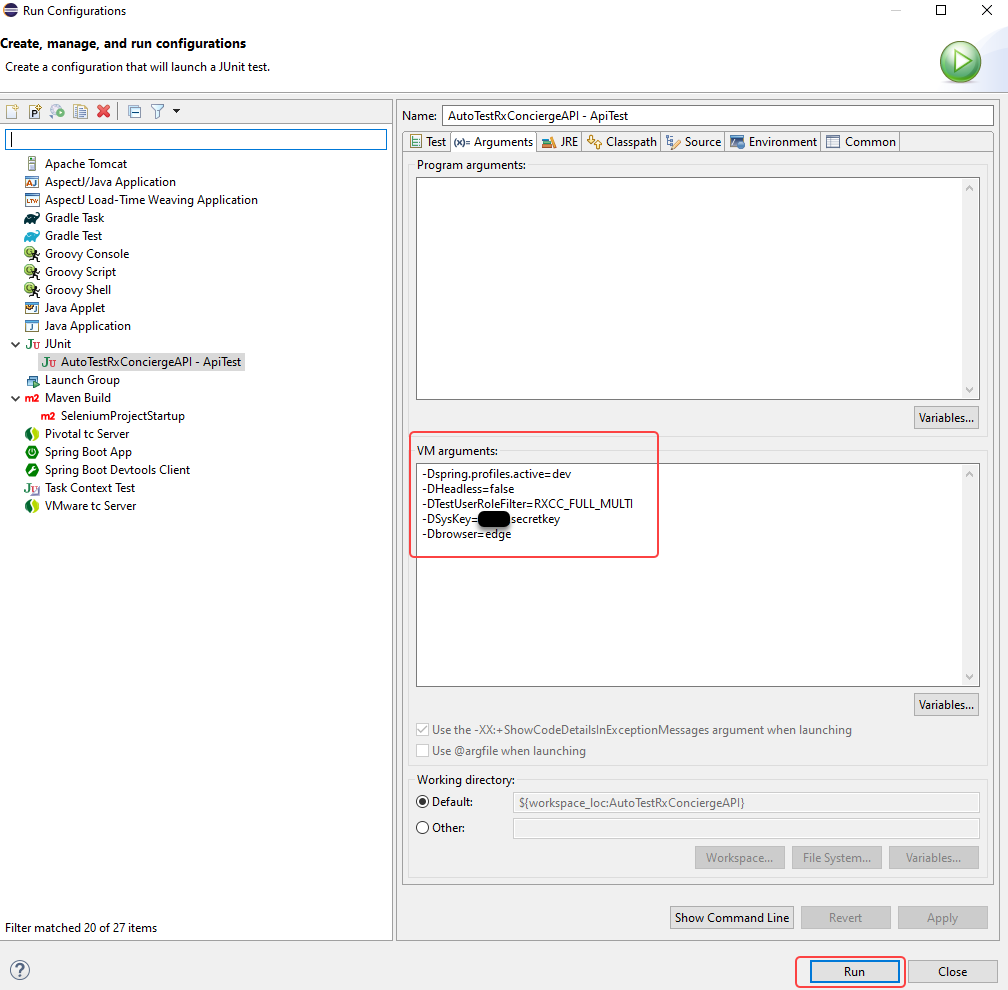
-Dspring.profiles.active=dev

-DHeadless=false

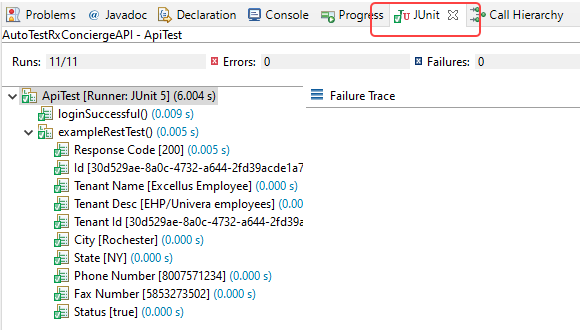
-DTestUserRoleFilter=RXCC\_FULL\_MULTI

-DSysKey=XXXXXX,secretkey

-Dbrowser=edge



1. Use JUnit view (tab) to test results



## Maven

TBD

## Test Parameters

There are parameters from JUnit utilities that can be used in this project, see [JUnit utility user guide](https://lifethc.sharepoint.com/sites/SQAArchitectureAndAutomationTeam_EIT_GRP/Shared%20Documents/General/Technical%20Testing%20References/Java/JUnit%20Utility.docx?web=1) section 5. Specific parameters for this project are as follows

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Description** | **Example** |
| -Dspring.profiles.active | Required: Environment (test or dev) | -Dspring.profiles.active=test |
| -DHeadless | Optional: Execute browser headless (true or false)  Default: false | -DHeadless=false |
| -DTestUserRoleFilter | Required: The user role to run test (see [roles section](#_Roles)) | Ex 1:  -DTestUserRoleFilter=SSO    Ex 2:  -DTestUserRoleFilter=SSO,RXCC\_FULL\_MULTI |
| -DSysKey | Required: *See test lead for the value* |  |
| -Dbrowser | Required: Web browser (either Chrome or Edge) | -Dbrowser=edge |

# Roles

|  |  |  |
| --- | --- | --- |
| **Test Suite ROLE** | **Username** | **Role description** |
| SSO | User running the test | Most likely it would have admin role |
| RXCC\_FULL\_MULTI | rxcc\_full\_multi@excellus.com | Admin access for multiple tenants |
| RXCC\_FULL\_SINGLE | rxcc\_full\_single@excellus.com | Admin access for single tenants |
| RXCC\_OPS\_MULTI | rxcc\_ops\_multi@excellus.com | Operational access for multiple tenants |
| RXCC\_OPS\_SINGLE | rxcc\_ops\_single@excellus.com | Operational access for single tenants |
| RXCC\_READ\_MULTI | rxcc\_read\_multi@excellus.com | Read access for multiple tenants |
| RXCC\_READ\_SINGLE | rxcc\_read\_single@excellus.com | Read access for single tenants |
| RXCC\_REPORT\_MULTI | rxcc\_report\_multi@excellus.com | Report access for multiple tenants |
| RXCC\_REPORT\_SINGLE | rxcc\_report\_single@excellus.com | Report access for single tenants |

# Test Tags

The annotation tag in test classes/methods is part of JUnit where it allows testers to group the test by features, examples:

-Dinclude-tag="SEARCH | FILTER" *will run any test that are tag with either SEARCH or FILTER*

-Dinclude-tag="MEMBER\_NOTES & MEMBER\_AUDITS" *will run any test that are tag with both MEMBER\_NOTES and MEMBER\_AUDITS*

|  |  |
| --- | --- |
| **Name** | **Definition** |
| **All entities** | |
| @Tag("ALL") | Entire regression test suite |
| @Tag("CHROME") | Chrome browser |
| @Tag("EDGE") | Edge browser |
| @Tag("ADMIN\_ROLE") | Admin role |
| @Tag("OPERATION\_ROLE") | Operation role |
| @Tag("READ\_ROLE") | Read role |
| @Tag("REPORT\_ROLE") | Report role |
| @Tag("SEARCH") | Any test related to searching |
| @Tag("FILTER") | Any test related to filter |
| @Tag("GLOBAL\_SEARCH") | Global search across multiple tenant |
| @Tag("FUZZY\_SEARCH") | Cognitive searching of members |
| @Tag("TENANT\_SEARCH") | Tenant search (on main page) |
| @Tag("FAX") | Any test related to fax |
| @Tag("NOTES") | Any test related to notes |
| @Tag("PRESCRIPTION") | Any test related to prescription |
| @Tag("AUDITS") | Any test related to audits |
| @Tag("MEMBER") | Any test related to members |
| @Tag("MEMBER\_DEMOGRAPHICS") | Test related to member demographics |
| @Tag("MEMBER\_INTERVENTIONS") | Test related to member interventions |
| @Tag("MEMBER\_PRESCRIPTION\_CLAIMS") | Test related to member prescription claims |
| @Tag("MEMBER\_PROVIDERS") | Test related to member providers |
| @Tag("MEMBER\_PHARMACY") | Test related to member pharmacies |
| @Tag("MEMBER\_CORRESPONDENCE") | Test related to member correspondence |
| @Tag("MEMBER\_NOTES") | Test related to member notes |
| @Tag("MEMBER\_SUMMARY") | Test related to member summary |
| @Tag("MEMBER\_AUDITS") | Test related to member audits |
| @Tag("PHARMACY") | Any test related to pharmacy |
| @Tag("PHARMACY\_DIRECTORY") | Test related to pharmacy directory |
| @Tag("PROVIDER") | Any test related to provider |
| @Tag("PROVIDER\_VIEW") | Test related to provider view |
| @Tag("PROVIDER\_SEARCH") | Test related to provider search |
| @Tag("PROVIDER\_FILTER") | Test related to provider filter |
| @Tag("PROVIDER\_EDIT") | Test related to provider edit |
| @Tag("PROVIDER\_NOTES") | Test related to provider notes |
| @Tag("INTERVENTION") | Test related to intervention |
| @Tag("INTERVENTION\_SEARCH") | Test related to intervention search |
| @Tag("INTERVENTION\_FILTER") | Test related to intervention filter |
| @Tag("INTERVENTION\_NEW\_FORM") | Test related to intervention new form |
| @Tag("INTERVENTION\_CLONE") | Test related to intervention clone existing intervention |
| @Tag("INTERVENTION\_SIMULATION") | Test related to intervention simulation run |
| @Tag("INTERVENTION\_FAX") | Test related to intervention fax |
| @Tag("INTERVENTION\_SETTING") | Test related to intervention settings |

# Azure DevOps

## GIT Pull Request

Add the individuals below when submitting pull request

* Garrett Cosmiano
* Husnain Zia
* Sowmya Machana
* Brian Keenan
* Nigel Powell

## Pipeline

TBD