01. Getting Started

Last updated by | Warren Duffin | Jun 7, 2023 at 1:08 PM GMT+5:30

Contents

Tool Chain

Below is a list of things required to deploy automation framework (both Webapps and API) on your system.

Azure QA/Dev VM

Automation should always be executed from an Azure VM in order to minimise latency between automation and the Database.

Azul Zulu

- All Azure QA/Dev VMs should have the correct version of Azul Zulu Java installed.
  - for CTMS version 15.04 and earlier this is Java 8
  - for CTMS version 15.05 and later this is Java 11
- Otherwise, install the relevant Azul Zulu Java see Upgrade Java and Tomcat Versions

IntelliJ - Community/Enterprise Edition

Ensure a version of IntelliJ is installed, either the Ultimate edition, if you have a license, or the Community edition, if you don't have a license. Both can be downloaded from the IntelliJ download page .

Git

The Azure VM should already have Git installed
Use git clone to obtain the CTMS code, which includes automation. i.e.

```
git clone https://Perceptive-Cloud@dev.azure.com/Perceptive-Cloud/Calyx%20CTMS/_git/CTMS
```

Xvfb Installation

Xvfb is an application which allows a virtual display to be used. Some automation projects use this to allow the scenarios to be executed on the virtual display so that the machine can still be used.

In order to install this on Ubuntu, execute `sudo apt install xvfb`

Opening Automation in IntelliJ

There are two ways in which IntelliJ can be used to to access the automation code.

Use the default CTMS IntelliJ project

Using this method allows full access to all of the CTMS and Automation code in a single IntelliJ project. Simply open the IntelliJ project in the `intellij_config/dev2021_linux` folder.

Create a new Project just for Automation

This has the advantage that you can just concentrate on the automation code, but has the disadvantage that you cannot see the CTMS code unless you also open the full project, as above, in a different window of IntelliJ
If this is done, it is not advisable to edit CTMS and Automation code in two different IntelliJ windows for the same GIT repository. IntelliJ can become confused about which files are edited in each window/change set

This can be done by doing the following

1. Select `File | Open` and open the `automation/automation-parent/pom.xml` . When asked select `open as project`
2. In the Maven Project window select `Add Maven Projects` and select the `automation/automation-impact-webapps/pom.xml`
3. In the Maven Project window select `Add Maven Projects` and select the `automation/automation-impact-services/pom.xml`
4. Ensure that the Java version for the project is configured as the same Azul Zulu Java version as the main CTMS project.

Configuring Automation to use the correct JDK

Using Java 8

If Automation needs to use Java 8 then all of the Automation modules should be configured to use the Project JDK.
Maven must also be configured to use the Project JDK, `File | Settings | Build | Build Tools | Maven | Runner`

Using Java 11

If automation needs to use Java 11 then this will required that all of the Automation modules are configured for a specific JDK Version.

Specific Automation IntelliJ Project

If an IntelliJ project has been created specifically for automation, then the Projects JDK can be set to the relevant Java 11 and no additional changes are required.

Combined Automation and CTMS IntelliJ Project

As the main CTMS code uses Java 8 and automation requires Java 11 then the following changes need to be made.

- Goto `File | Project Structure | Modules`
- For all of the automation modules
  - Open the 'Dependencies' tab
  - Change the Module SDK to the configured Java 11 JDK
- Configure IntelliJ to run maven using Java 11, `File | Settings | Build | Build Tools | Maven | Runner`

Once this has been done ONLY automation should be built directly using Maven target in IntelliJ. Other projects e.g. Impact-services should be built using ant targets.

Jenkins

Ensure couple of team members have access to Jenkins and that Members are added to e-Mail notifications
Note: Jenkins will send notification on a regular basis (Test Results and Build Artifacts).

Third Party Jars

All of the third party jars should be monitored for updates to ensure the latest versions are being used.

Selenium

Selenium is the library which is used to communicate with the Browser. This has regular updates which need to be monitored. When an update is available then the automation project must be updated to use it.

Cucumber

Cucumber is the library which allows the use of Gherkin in the feature files and processes them so that they can be executed by TestNg.

Rest Assured

Rest Assured is the library which handles the Http requests/responses in the Api automation scenarios.

TestNg

TestNg is the underlying framework which is used to execute the Scenarios.