PRISMAproduction Behavior Driven

Test Automation

This document summarizes the behavior driven framework installation steps, test definition, implementation and execution workflows.

# Deployment

1. Install Oracle Java jdk 8; add to PATH system environment the location of the jdk\bin
2. Add certificate to java cacerts. Password = changeit. Cmd started with Admnistrator

* JDK mandatory

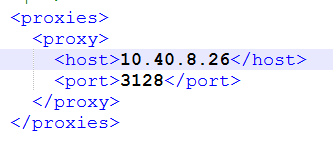
“C:\Program Files\Java\jdk1.8.0\_131\bin\keytool.exe” -import -trustcacerts -file "<certificate\_file\_path>" -alias "cert\_alias" -keystore "C:\Program Files\Java\jdk1.8.0\_xxx\jre\lib\security\cacerts"

* JRE if exists on the system

C:\Program Files\Java\jdk1.8.0\_131\jre\lib\security\keytool.exe -import -trustcacerts -file "<certificate\_file\_path>" -alias "cert\_alias" -keystore " C:\Program Files\Java\jre1.8.0\_xxx\lib\security\cacerts"



1. Install IntelliJ (from JetBrains – the community edition)
2. Install Maven (3) & configure it
   1. provide proxy info in settings file (./conf/settings.xml)

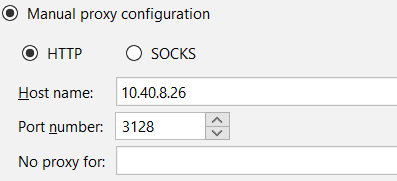


Set the following environment variables: M2\_HOME = path to Maven; M2 = %M2\_HOME%\bin

Add also %M2% to the PATH environment variable

(\*) Configuration proxy.pac script cannot be used; proxy has to be manually entered

1. Install GIT
2. Start IntelliJ
3. Go to File -> Settings
   1. Maven and provide the path to Maven previously installed
   2. Server Certificates and add Oce certificate
   3. Appearance & Behavior -> System Settings -> HTTP Proxy



(\*) Proxy.pac can be used

* 1. Set jdk into IntelliJ
  2. Add the following plugins: “Cucumber for java” and “Gherkin”

1. In “Mvn Projects” view go to the ppautomated project (View -> tool windows -> maven project)-> Lifecycle and do a Clean, Install, Deploy
2. Open Project View to access project resources
3. In order to be able to also execute Selenium tests, Selenium WebDriver needs to be installed. For this, download the latest Selenium Web Driver – Java, unzip and copy the executable into a folder. Now add that folder to the PATH variable. Open cmd and type chromedriver -> should start. Chrome browser needs to be installed.
4. For image comparison tools go to [\\sro-gsx35\TaV\_CP\Automation](file:///\\sro-gsx35\TaV_CP\Automation) and install gstools 9.22 & imagemagick 7.0.7 tools. Within the installation step “Select Additional tasks” choose the option to install legacy utilities.

Hint: To update maven dependencies (after pom.xml update): mvn dependency:copy-dependencies

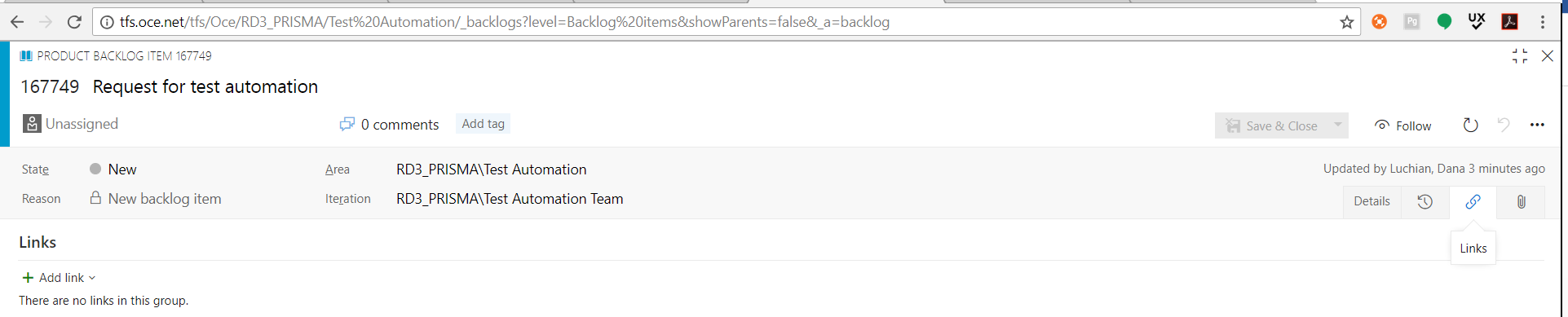
# Good practices

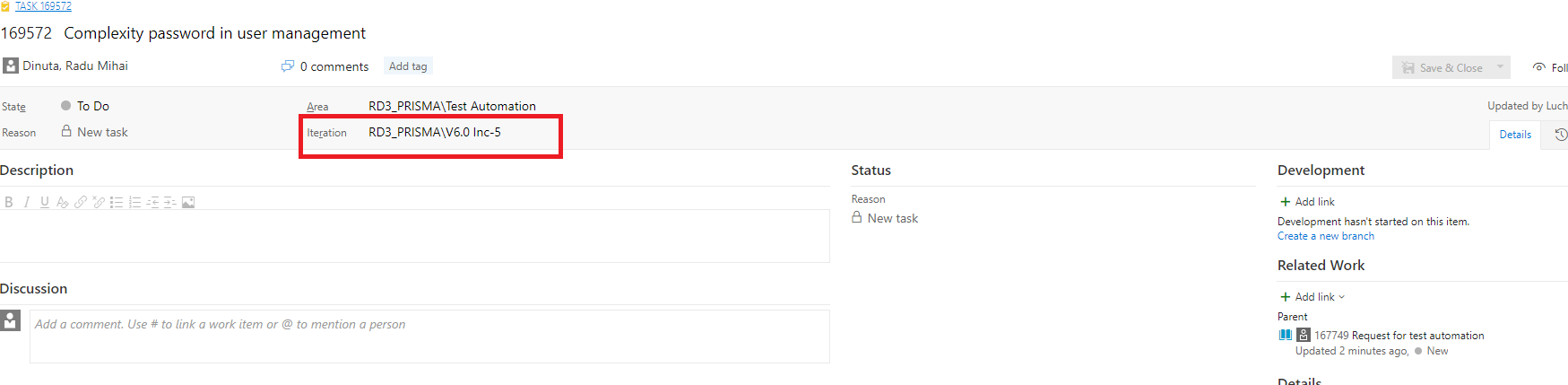
* Every morning or several times per day or at least before committing changes, do a PULL

# Test definition

The tests are written in Gherkin language by the test team. When new scenarios are available and ready to be implemented, then a task in TFS is to be created. The steps are below:

* Go to the Automation Team current iteration (left side -> current): <http://tfs.oce.net/tfs/Oce/RD3_PRISMA/Test%20Automation/_backlogs?level=Backlog%20items&showParents=false&_a=backlog>
* Search for generic PBI called “Request for test automation”, open it
* Go to links, click Add link -> New item
* Provide information about the tests to be automated in the title (BD/DD + component name mandatory) and if needed in the comments; make sure the link type is Child and the work item type is Task. Assign the task to the current iteration





**Selenium tests (UI)** – add **@component** and **@undefined** tags to make sure that the tests are assigned to the corresponding component (first tag) and that they are not executed without being implemented (second tag).

**Karate tests (API)** - since most of the keywords are already provided by Karate, for API tests, no further implementations should be needed.

For these tests the **@component** tag should be added and the test should be executed before checking it in to make sure that it is functional.

In case additional implementations are needed then @undefined tag should be added.

Used component tags: **@enhancer, @workflowdesigner, @sms, @viewer, @impose, @simple\_submit, @web\_explorer, @user\_management**

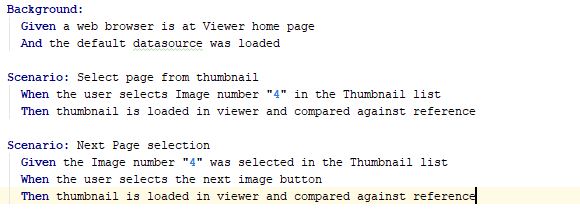
If needed to execute the scenario on the integrated version (not standalone mode) then a **@system** tag should be added.

Based on these tags, the tests are scheduled for execution within Jenkins daily jobs.

Tags at feature level are inherited by the tests included.

If a test doesn’t have a @undefined tag it will executed in the night, even if it is unimplemented.

Each test is independent of the other tests. For example, test nr.2 in the feature file does not start from where test nr.1 finished, but it starts from the background step of the feature file (or the given step if there is no background). In the example below the first scenario finishes, and then the next test starts from scratch (from the background step).



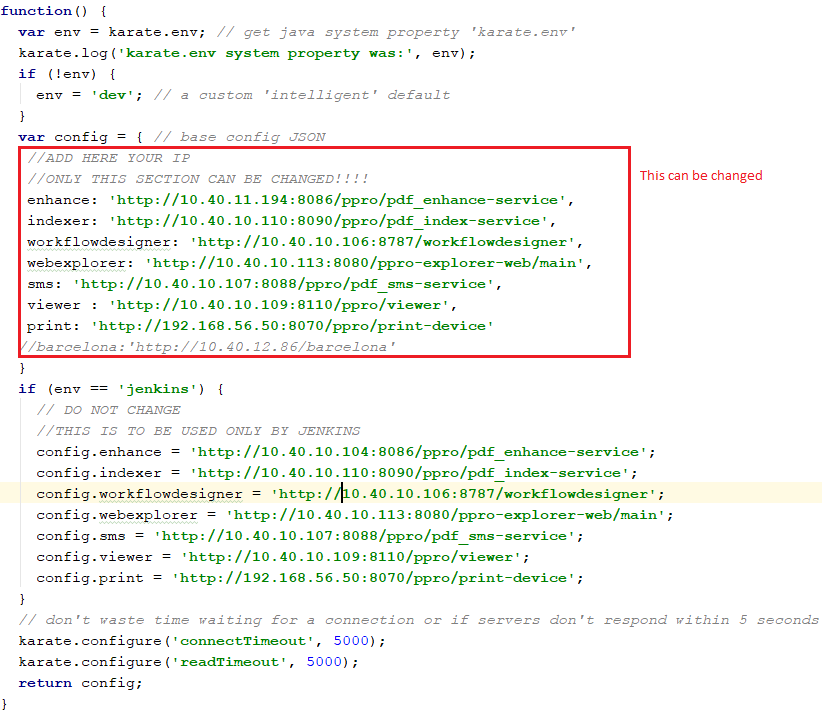
When implementing new tests put @*TestName* above the scenario. This will help us when we want to run a single test.

For example:

@NextPageSelection

Scenario: Next Page selection

Default URL’s were configured to be used. These are available in the karate-config.js file. This configuration file has two parts: one with default IP’s (that is to be changed) and another one with the IP’s that are to be used during nightly execution (from Jenkins).



For PDF verification: first run on a Windows (local machine) there has to be a rsa key exchange between Windows and Linux, so the following command must be run in command prompt (cmd.exe): **<Project\_Path>\src\main\resources\pscp.exe -pw .pwroot root@<Linux\_IP>:/home/slave.jar C:\temp**

**Output verification**

Additionally a pdf comparison function was implemented and can be added to the test if needed. Usage example can be found in enhancer API minimal feature. This is to be used for output validation. Once the first outputs are available then these should be manually checked and if ok copied to [\\sro-gsx35.sro.oce.net\TaV\_CP\Test Files\Automation\ReferenceFiles](file:///\\sro-gsx35.sro.oce.net\TaV_CP\Test%20Files\Automation\ReferenceFiles)

This function needs the following mapped drive: J -> \\sro-gsx35.sro.oce.net\TaV\_CP\Test Files\Automation



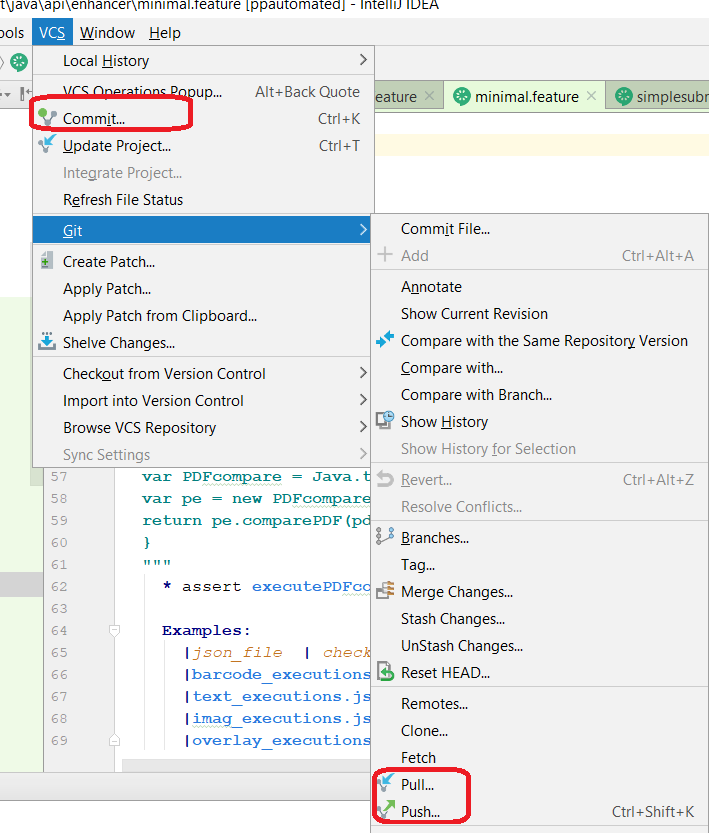
Name of the folders in which the tiffs are generated and afterwards used as references. In this example the tiffs are in: C:\CompareFiles\_ppauto\newFiles\barcode2\_APIexecitions. These have to be copied to the specified location on gsx35. During nightly execution these are automatically copied on the machine

Service output file. Every page is afterwards covered to a tiff file and compared against reference

**Version Control**

After making changes (creating a new test, adding new tags, etc.) in order to make the change persistent and available to all the team members, this change has to be checked in. The steps are:

* Pull – to get the latest changes
* Commit change – provide a suggestive text & check appears as changed
* Push the change into GIT



As best practice: every day do a Pull in the morning before starting work. Do NOT put comments before the feature name.

When new tests are defined and implementations needed, the automation team should be informed.

# Test implementation

The test implementation is done by the automation team member.

General coding best practices apply.

* Comment your code
* Use proper indentation
* Provide descriptive names to the functions
* Make sure functions are reusable; always think on reuse possibilities

When a new scenario/feature is implemented, it’s @undefined tag should be erased and implementations should be reviewed by a peer before committing.

# Test execution

Tests can be executed on the local machine in IntelliJ as follows: select the feature file, right click and choose “Run feature ..”. For Selenium tests provide the following VM options in the Run Configuration:

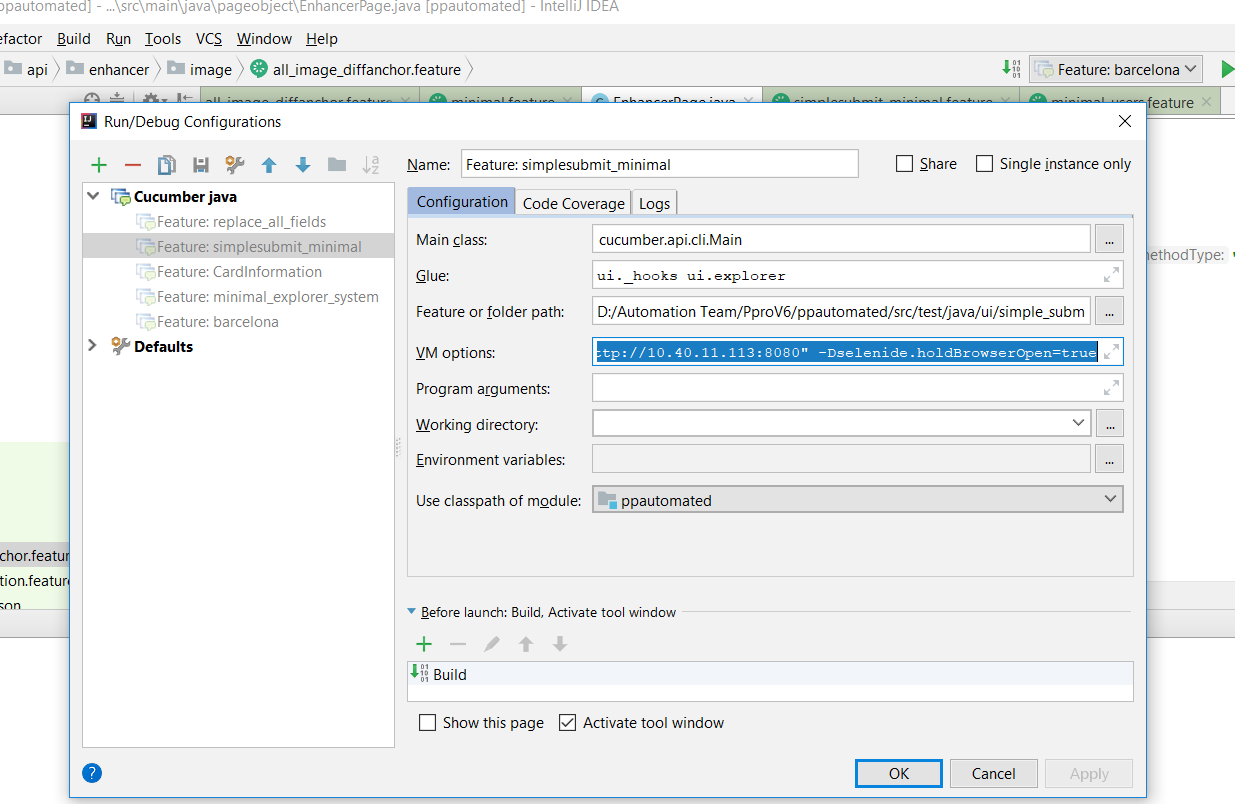
-Dselenide.browser=utils.CustomWebDriverProvider -Dcucumber.options="--tags @runMe" -Dselenide.baseUrl="http://10.40.11.113:8080" -Dselenide.holdBrowserOpen=true

Where: @runMe – tag of the scenario to be executed

Choose Edit Configurations; select the feature file and change VM options

baseUrl – component URL

holdBrowserOpen = true or false = after execution browser remains open or not

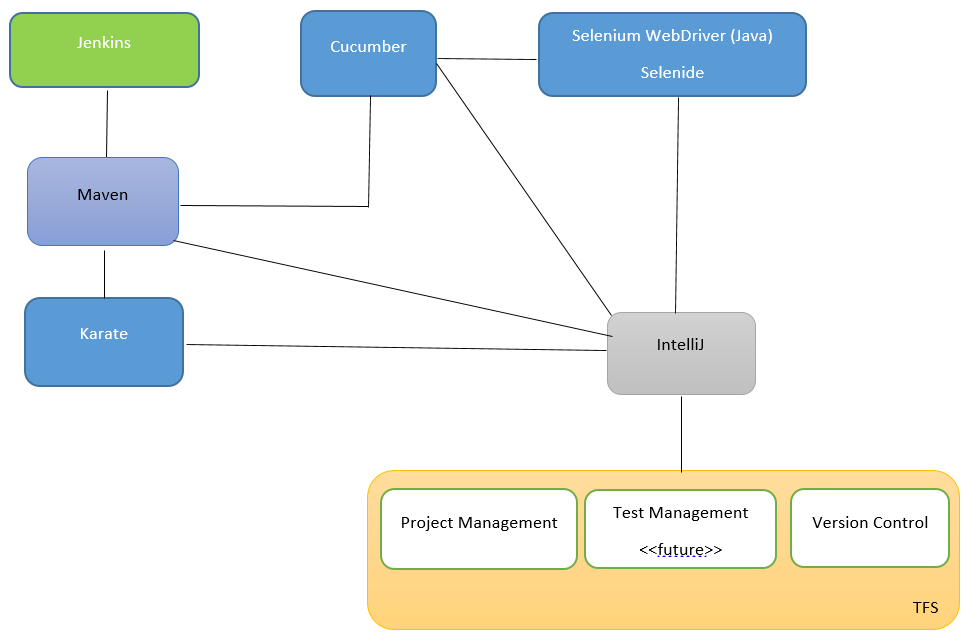


Tests are executed on a daily basis in Jenkins. Links to execution reports are submitted via email to the involved functional responsible, tester, developer & automation team.

Jenkins link: <http://pp-jenkinss.sro.oce.net:8080/jenkins/>

Screenshots are available here: [\\pp-client\Utils\workspace\<<job\_name](file:///\\pp-client\Utils\workspace\%3c%3cjob_name)>>; user=pp-client\admin; pass=ocevm$10x64

# Used tools



* **Jenkins** - web application that handles building, test running and reporting through several plugins) – free for corporate
* **IntelliJ (\*)** Idea Community edition - used as IDE for automation – free for corporate (.java associations)
* **Maven** - project building tool – free for corporate
* **TFS Git** - source code management tool, automation team planning tool and in the future test management tool
* **Karate (\*)** - tool used for API testing – free for corporate
* **Selenium (\*)** - tool used for UI testing – free for corporate
* **Selenide** (tool that creates a framework on top Selenium to help with some know issues and ease of use) – free for corporate