D:\Logos\CG_logoReflect.tif

Description: C:\Users\244345\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\MobileTestingLogonew.png

**QE&A Technology CoE**

**7/31/2017**

C.R.A.F.T MAVEN

**Table of Contents**

[**1. Purpose** 3](#_Toc489363901)

[**2. Pre-requisite** 4](#_Toc489363902)

[**3. Desktop Automation Settings** 4](#_Toc489363903)

[**3.1. Working with Global Settings Properties** 4](#_Toc489363904)

[**3.2. Working with Run Manager** 4](#_Toc489363905)

[**3.3. Working with TestConfigurationID & Sheet** 5](#_Toc489363906)

[**3.4. Points to be noted while using Actions, JavaScriptExecutor, findElements** 7](#_Toc489363907)

[**4. Appium Automation Settings** 7](#_Toc489363908)

[**4.1. Working with Mobile Automation Settings properties** 7](#_Toc489363909)

[**4.2. Run Manager** 8](#_Toc489363910)

[**4.3. Parallel Execution** 9](#_Toc489363911)

[**4.4. Points to be noted** 9](#_Toc489363912)

[**5. Perfecto Automation Settings** 9](#_Toc489363913)

[**5.1. Working with Mobile Automation Settings** 9](#_Toc489363914)

[**5.2. Working with Run Manager** 10](#_Toc489363915)

[**6. SeeTest Automation Settings** 10](#_Toc489363916)

[**6.1. Mobile Automation Settings Properties** 10](#_Toc489363917)

[**6.2. Run Manager Settings** 11](#_Toc489363918)

[**7. Sauce Lab Automation Settings** 12](#_Toc489363919)

[**7.1. Mobile Automation Settings** 12](#_Toc489363920)

[**7.2. Run Manager** 12](#_Toc489363921)

[**8. Browser Stack Automation Settings** 12](#_Toc489363922)

[**8.1. Mobile Automation Settings** 12](#_Toc489363923)

[**8.2. Run Manager** 13](#_Toc489363924)

[**9. Mint Automation Settings** 13](#_Toc489363925)

[**9.1. Mobile Automation Settings** 13](#_Toc489363926)

[**9.2. Run Manager** 13](#_Toc489363927)

[**10. HP Mobile Center Automation Settings** 14](#_Toc489363928)

[**10.1. Mobile Automation Settings** 14](#_Toc489363929)

[**10.2. Run Manager** 14](#_Toc489363930)

[**11. Integration of Amazon Device Farm** 14](#_Toc489363931)

[**11.1. Configure ADF settings in Framework** 14](#_Toc489363932)

[**11.2. Execute CRAFT Maven in ADF** 15](#_Toc489363933)

[**12. C.R.A.F.T QC INTEGRATION** 16](#_Toc489363934)

[**12.1. C.R.A.F.T and QC setup** 16](#_Toc489363935)

[**12.2. Execution in Test Plan** 21](#_Toc489363936)

[**12.3. Executing QC from local without opening QC** 22](#_Toc489363937)

[**13. Jenkins Integration** 23](#_Toc489363938)

[**13.1. Pre-Requisites for Jenkins CRAFT Integration** 23](#_Toc489363939)

[**13.2. Executing Allocator from POM file** 23](#_Toc489363940)

[**13.3. Executing TestNG from POM file** 24](#_Toc489363941)

[**13.4. Invoking from Jenkins** 25](#_Toc489363942)

[**13.5. Report update to Jenkins from CRAFT** 26](#_Toc489363943)

[**14. JIRA Integration** 26](#_Toc489363944)

[**14.1. Global Settings** 26](#_Toc489363945)

[**14.2. Updating the JIRA API’s** 27](#_Toc489363946)

[**15. C.R.A.F.T – TestNG Implementation** 27](#_Toc489363947)

[**15.1. One-to-one Mapping for TestNG classes** 27](#_Toc489363948)

[**15.2. One-to-one Mapping for TestNG classes** 28](#_Toc489363949)

[**15.3. Batch Execution with TestNG** 29](#_Toc489363950)

[**15.4. Parameterization of Test Parameters** 29](#_Toc489363951)

[**15.5.1. Rerunning the Failed Test Case at that moment** 31](#_Toc489363952)

[**15.5.2. Rerunning the Failed Test case at the end of execution** 31](#_Toc489363953)

[**16. Modular Driven Approach within CRAFT** 32](#_Toc489363954)

[**16.1. Download Modular Driven Samples** 32](#_Toc489363955)

[**16.2. Configuration for Modular Driven Approach** 32](#_Toc489363956)

[**17. Leverage the Reusable functions with Framework** 32](#_Toc489363957)

[**17.1. Working with Selenium, Appium & Perfecto Reusable Functions** 33](#_Toc489363958)

[**17.2. Working with additional Data Access Functions** 33](#_Toc489363959)

[**17.3. Working with Broken Links** 34](#_Toc489363960)

# **1. Purpose**

The purpose of this document is to provide description of advanced features in CRAFT Maven Framework. This document also provides step by step instruction for implementing the respective features.

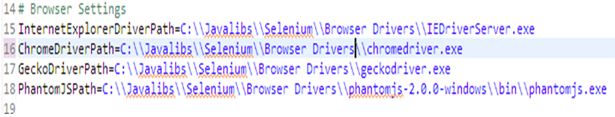
# **2. Pre-requisite**

The user is expected to have basic knowledge on Automation Testing. Having a working knowledge of CRAFT Framework (Basic workflow), Jenkins, ALM/QC and TestNG knowledge will be an added advantage.

# **3. Desktop Automation Settings**

## **3.1. Working with Global Settings Properties**

* Provide the respective Browser executable paths like InternetExplorerDriverPath/ChromeDriverPath/GeckoDriverPath under Browser Settings



## **3.2. Working with Run Manager**

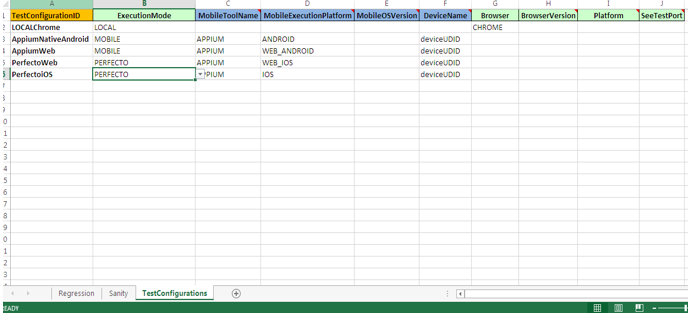
* Configure the Run manager with all required details such as Test Scenario, TestCase, Test Instance, Iteration, and TestConfigurationID.



* The TestConfigurationID’s are prepopulated from **TestConfigurations** Sheet. All Configuration Details are abstracted in separate sheet. Please refer to section 3.3 for Parameters & possible values.
* For Desktop automation, set the following as,
  + **Execution Mode** – LOCAL
  + **Browser** – CHROME/ FIREFOX/ SAFARI/ GHOST\_DRIVER/ INTERNET\_EXPLORER/ OPERA
  + **BrowserVersion** – This option is applicable only when ExecutionMode=GRID.

## **3.3.** **Working with TestConfigurationID & Sheet**

* TestConfigurations Sheet contains all execution details mapped to a **TestConfigurationID**.
* Any New Test Configuration can be added.
* The TestConfigurationID should be Unique Name. The TestConfigurationID can be referred within your Suite Sheets like Regression, Sanity.



|  |  |  |
| --- | --- | --- |
| **Column Name** | **Possible values** | **Additional Details** |
| TestConfigurationID | Any Valid Unique Name – Should Explain the Respective Configuration |  |
| ExecutionMode | |  | | --- | | LOCAL | | REMOTE | | LOCAL\_EMULATED\_DEVICE | | REMOTE\_EMULATED\_DEVICE | | GRID | | MOBILE | | PERFECTO | | SEETEST | | MOBILELABS | | SAUCELABS | | MOBILECENTER | | BROWSERSTACK | | MINT | |  |
| MobileToolName | |  | | --- | | DEFAULT | | APPIUM | | REMOTE\_WEBDRIVER | | These Values Depend on the ExecutionMode |
| MobileExecutionPlatform | |  | | --- | | ANDROID | | IOS | | WEB\_ANDROID | | WEB\_IOS | |  |
| MobileOSVersion | Version of the Device | Optional value |
| DeviceName | Device Name/UDID |  |
| Browser | |  | | --- | | CHROME | | FIREFOX | | GHOST\_DRIVER | | HTML\_UNIT | | INTERNET\_EXPLORER | | OPERA | | SAFARI | | For Mobile it will take default CHROME for Android & SAFARI for iOS. |
| BrowserVersion | Version of the browser | Applicable only if ExecutionMode is GRID, also wanted to test for different versions of Browser |
| Platform | Platform of OS | Applicable only if ExecutionMode is GRID |
| SeeTestPort | SeeTestPort like 8889 etc., | Applicable only if ExecutionMode is SEETEST |

## **3.4. Points to be noted while using Actions, JavaScriptExecutor, findElements**

As Craft implements dynamic polymorphism and we use CraftDriver on Top of all the WebDriver implementation. We would some time required to pass original driver object for few of the implementation like Actions, JavaScriptExecutor.

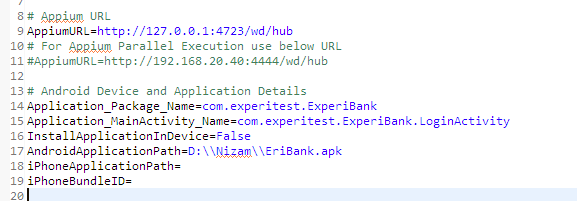
* To achieve this, use ***driver.getWebDriver()*** instead of driver.



# **4. Appium Automation Settings**

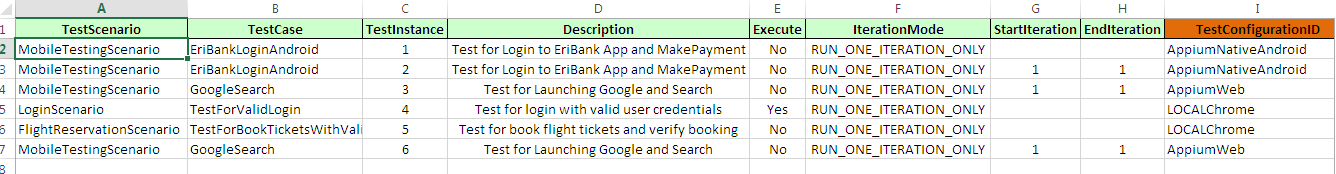
## **4.1. Working with Mobile Automation Settings properties**

All details of Application & Appium has to be mentioned under Mobile Automation Settings property file as below,

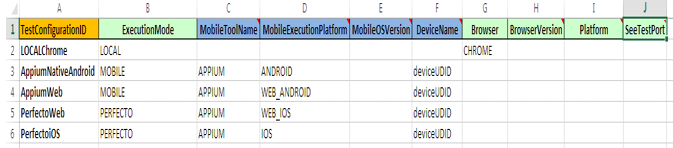
* **AppiumURL**: Please provide the Appium running URL, generally it will be http://127.0.0.1:4723/wd/hub
* **Application\_Package\_Name:** Provide the Package Name of Android Application
* **Application\_MainActivity\_Name:** Provide the Main Activity Name of Android Application.
* **InstallApplicationInDevice**: Helps to install App from below Path into the respective Device
* **AndroidApplicationPath**: Provide the Application path of Android Application.
* **iPhoneApplicationPath**: Provide the Application path of iOS Application.
* **iPhoneBundleID**: Provide the iOS application bundle ID.

## **4.2. Run Manager**

* Configure the Run manager with all required details such as Test Scenario, TestCase, Test Instance, Iteration, and TestConfigurationID.



* For Appium automation, please change configure as following within TestConfigurations Sheet
  + **ExecutionMode** – MOBILE
  + **MobileToolName** – APPIUM
  + **MobileOSVersion** - Optional
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



## **4.3. Parallel Execution**

In Global Settings, mention the number of threads based on the number of devices.

* We would need to implement Grid concept to achieve Parallel Execution. Download the Document for settings up Hub and Node from <http://acoetoolkit.cognizant.com/setup/CRAFTVersions/V3/Addons/Appium_Parallel_executionFiles.zip>

## **4.4. Points to be noted**

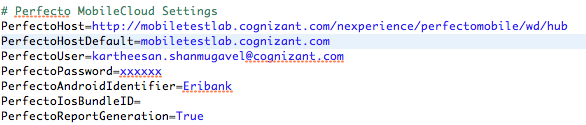
As Craft implements dynamic polymorphism and we use CraftDriver on Top of all the WebDriver implementation. We would some time required to pass original driver object for few of the implementation.

* To achieve this, use ***driver.getAppiumDriver()*** instead of driver.

# **5. Perfecto Automation Settings**

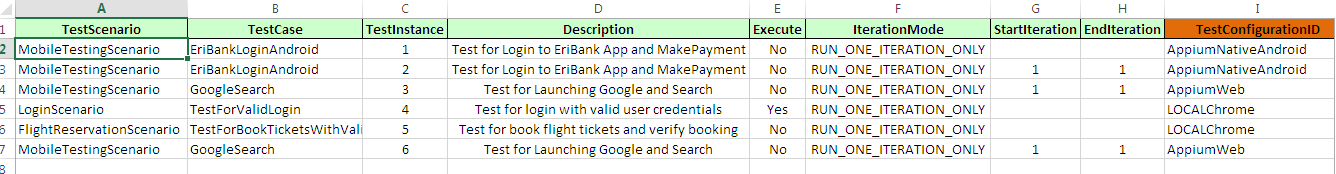
## **5.1. Working with Mobile Automation Settings**

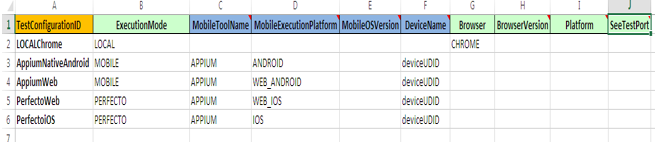
* **PerfectoHost** – Mention the Host details of Perfecto Mobile Cloud as to access the Cloud devices
* **PerfectoUser, PerfectoPassword** – Mention the authentication details of Perfecto Mobile Cloud
* **PerfectoAndroidIdentifier** – Mention the Android application’s name or identifier to specify the Native or Hybrid Application under test
* **PerfectoIosBundleID** – Mention the iOS Bundle Identifier to specify the iOS Native or Hybrid Application which is under test
* **PerfectoReportGeneration** – Mention True or false to enable or disable the report generation in Craft local Reports folder



***Note****: If User want to Use Bundle ID or Package Main Activity, kindly use other settings to pass the details.*

## **5.2. Working with Run Manager**

* Configure the Run manager with all required details such as Test Scenario, TestCase, Test Instance, Iteration, and TestConfigurationID.
* For Appium automation, do the following
  + **ExecutionMode** – PERFECTO
  + **MobileToolName** – APPIUM/REMOTE\_WEBDRIVER
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



# **6. SeeTest Automation Settings**

Listed below are the changes needs to be done at setting level while working with SeeTest Automation with CRAFT

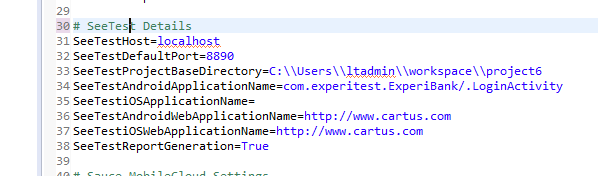
## **6.1. Mobile Automation Settings Properties**

* **SeeTestHost**: This property needs to be set “localhost”.
* **SeeTestDefaultPort**: By default SeeTest comes with two ports “8890”,”8889”.

Any one port number can be used in SeeTestDefaultPort.

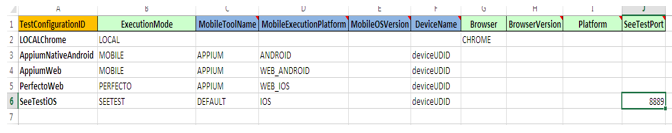
* **SeeTestProjectBaseDirectory**: This directory is used by SeeTest Automation for storing image objects. We need to give full path with directory name.
* **SeeTestAndroidApplicationName**: If you are testing Native or Hybrid Android application then application name has to be given in this property.
* **SeeTestiOSApplicationName**: If you are testing Native or Hybrid iOS application then application name has to be given in this property.
* **SeeTestAndroidWebApplicationName**: If you are testing a web application on Android device then application name has to be given in this property.
* **SeeTestiOSWebApplicationName**: If you are testing a web application on iOS device then application name has to be given in this property.
* **SeeTestReportGeneration**: If you want to generate SeeTest Automation reports along with C.R.A.F.T. reports then this property should be true else it will be false.

Sample properties are given in below image.

****

## **6.2. Run Manager Settings**

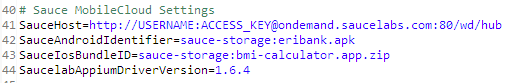
* Listed below are the options needs to be chosen in Run Manager for SeeTest Automation in C.R.A.F.T
  + **ExecutionMode**: From dropdown you need to select “SEETEST”.
  + **MobileToolName**: It will be “DEFAULT”.
  + **MobileExecutionPlatform**: It depends on what type of application you are testing
    - **ANDROID**: if Native or Hybrid Android application
    - **IOS**: if Native or Hybrid iOS application
    - **WEB**\_ANDROID: if web android application
    - **WEB\_IOS**: if web iOS application.
  + **DeviceName**: this field contains device name or UDID of the mobile device on which you are testing application.
  + **SeeTestPort**: Default SeeTest ports like “8889” or “8890” any one of these can be chosen.



# **7. Sauce Lab Automation Settings**

## **7.1. Mobile Automation Settings**

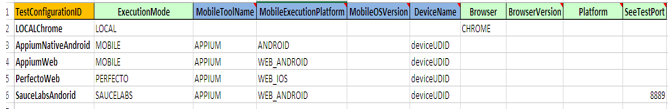
* Mention the sauce lab host.
* For Android, mention the Android bundle Identifier.
* For iOS, mention the IOS bundle Identifier & AppiumVersion to be used.



***Note: Saucelabs will support only selenium standalone server 2.53.1 & below***

## **7.2. Run Manager**

* For Sauce Lab automation, do the following in TestConfigurations Sheet
  + **ExecutionMode** - SAUCELABS
  + **MobileToolName** -APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** - Provide the respective device serial name or UDID



# **8. Browser Stack Automation Settings**

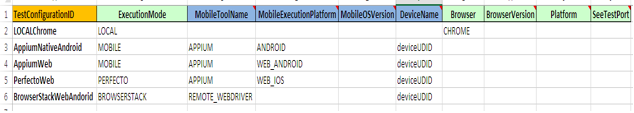
## **8.1. Mobile Automation Settings**

* Mention the BrowserStackHost.



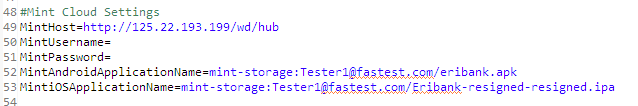
## **8.2. Run Manager**

* For Sauce Lab automation, do the following in TestConfigurations Sheet
  + **ExecutionMode** – BROWSERSTACK
  + **MobileToolName** –REMOTE\_WEBDRIVER
  + **MobileExecutionPlatform** - WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



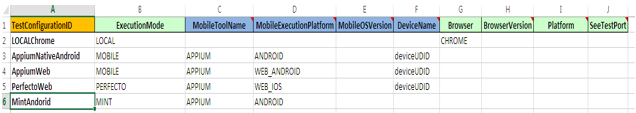
# **9. Mint Automation Settings**

## **9.1. Mobile Automation Settings**

* Mention the Mint host, Username & Password.
* Upload an APK/ IPA file in mint cloud and provide the respective application path.

## **9.2. Run Manager**

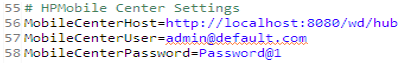
* For Sauce Lab automation, do the following in TestConfigurations Sheet
  + **ExecutionMode** – MINT
  + **MobileToolName** -APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



# **10. HP Mobile Center Automation Settings**

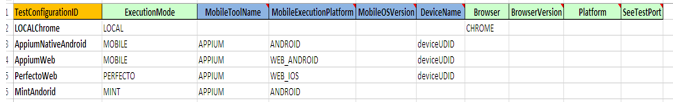
## **10.1. Mobile Automation Settings**

* Mention the **HP Mobile Center Host, User & Password**
* **Application\_Package\_Name**: Provide the Package Name of Android Application
* **Application\_MainActivity\_Name:** Provide the Main Activity Name of Android Application.
* **iPhoneBundleID**: Provide the iOS application bundle ID.



## **10.2. Run Manager**

* For Sauce Lab automation, do the following in TestConfigurations Sheet
  + **ExecutionMode** – MOBILECENTER
  + **MobileToolName** -APPIUM
  + **MobileExecutionPlatform** - ANDROID/IOS/WEB\_ANDROID/WEB\_IOS
  + **DeviceName** – Provide the respective device serial name or UDID



# **11. Integration of Amazon Device Farm**

## **11.1. Configure ADF settings in Framework**

Please download the additional files (POM.xml & Zip.xml files) from <http://acoetoolkit.cognizant.com/setup/CRAFTVersions/V3/Addons/ADF.zip>

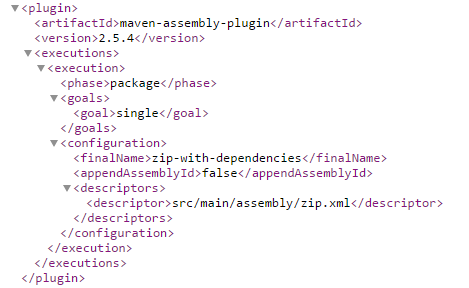
* Replace the old POM.xml file with new POM in CRAFT Maven.
* The configuration that are given in the zip.xml file instructs Maven to build a .zip file containing everything in the root of the build output directory and the dependency-jars directory

***NOTE:*** *Paste the* ***zip.xml*** *file under* ***src/main/assembly*** *(Create an ‘assembly’ folder under src/main package)*

* Configure the test cases in testng.xml file (NOTE: Make sure that you have named the file as testng.xml only)
* Place the testng.xml file under src/test/resources package
* Specify the xml file path in POM under Maven Surefire plugin.
* ADF will automatically perform TearDown activities after the execution, so no need to mention teardown method in testscripts package. Comment the tearDownTestRunner(testParameters, driverScript) method.

## **11.2. Execute CRAFT Maven in ADF**

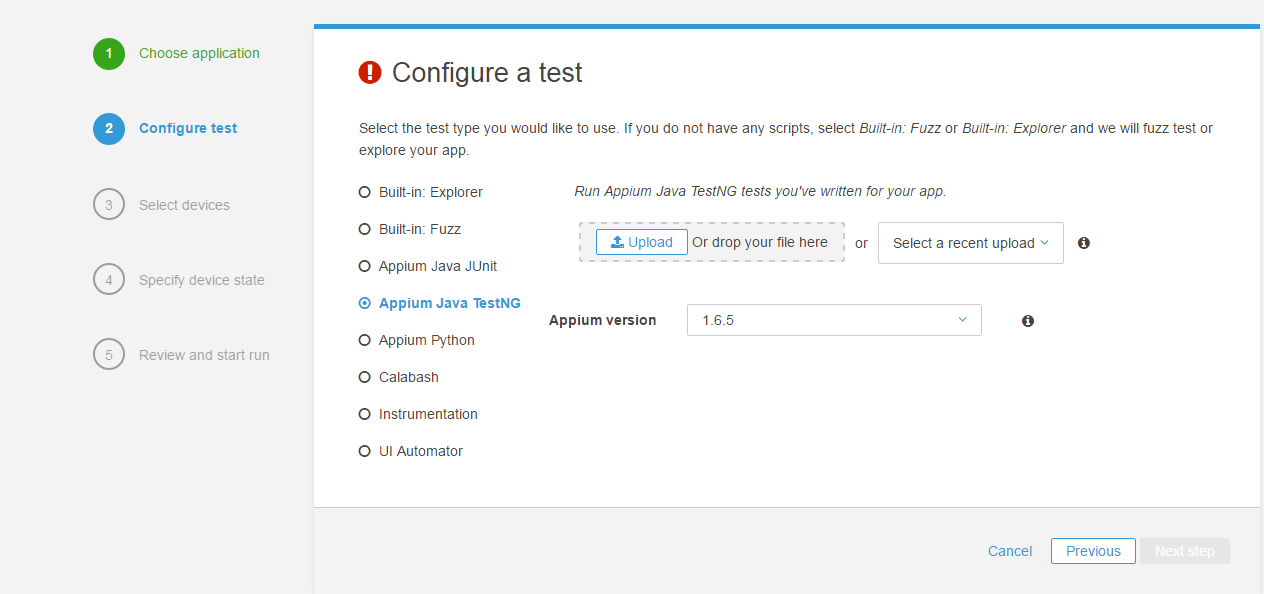
* Maven assembly plugin is required to package a project into zip format,



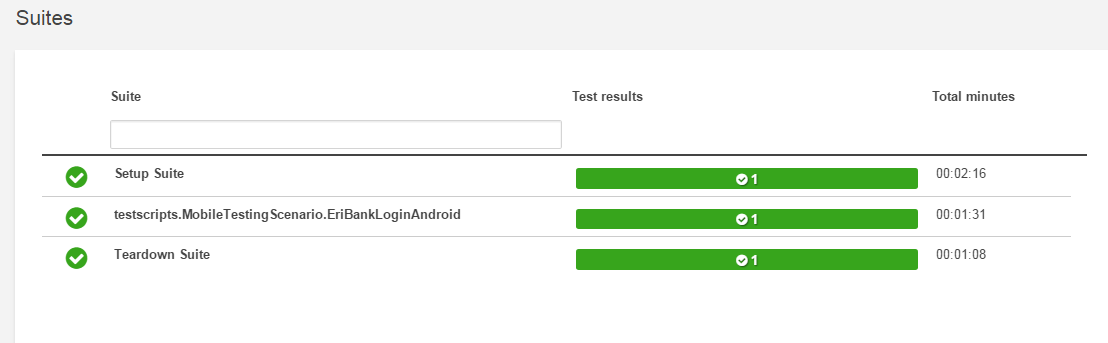
* Execute the POM using maven goal,

**mvn -P ProfileName package –DskipTests=true**

* Zip-with-dependencies.zip file will get generated under target/ folder.
* Create a new Run & upload the application if required (apk/ipa)
* Choose Appium Java TestNG option and configure the test,



* Upload the zip-with-dependencies.zip file to run the test.
* Select the devices and start the run.
* Results will be generated in Device Farm itself,



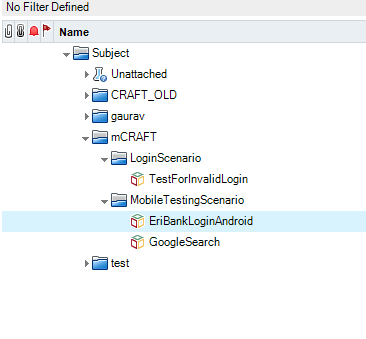
* Reports will contain Test execution video, Logs, Performance details and other Appium TestNG files.
* **NOTE**: Configuring Amazon Device Farm is out of box for this doc. Please refer <http://docs.aws.amazon.com/devicefarm/latest/developerguide/setting-up.html>

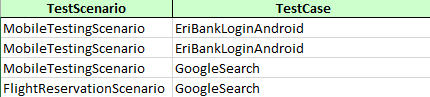
# **12. C.R.A.F.T QC INTEGRATION**

C.R.A.F.T. has capability to be integrated with QC. Listed below are the steps for the integration.

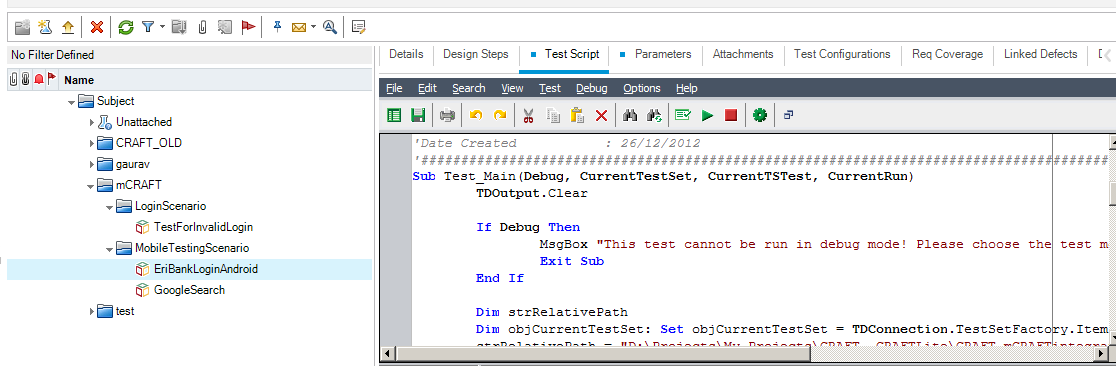
## **12.1.** **C.R.A.F.T and QC setup**

* Open HP QC by providing valid user credentials.
* Click on Testing Menu present on Left hand side panel.
* Under Testing menu click on Test Plan
* Create a Folder under root folder under Subject. For better readability you can name as Eclipse Project Name example : mCRAFT
* Now create a sub-folder under mCRAFT folder and make sure the name of the
  + Sub-folder is referred to TestScenario Name in Run Manager.xls e.g. “Scanerio1”, “MobileTestingScenario” etc.

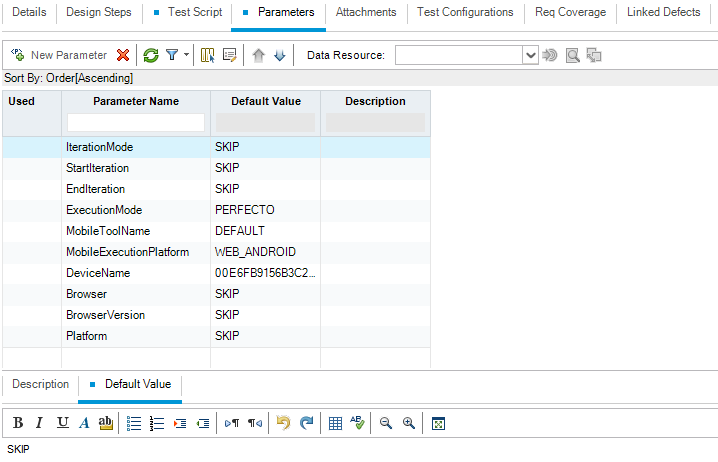




* Create a New VAPI – XP - Test , click on Next, Next and finish to create the script, name it as same name referred as “Test Case “ in (Run Manager.xls) CRAFT framework and newly created VAPI – XP Test should be present under “Scenario1”/ “MobileTestingScenario” subfolder
* Navigate to Eclipse (CRAFT) project and look up for package called “Miscellaneous” and copy paste the content of “VAPI-XP Script.vbs” and paste it under same test case created in above step in Test Script Tab



* Now create a new set of parameters as mentioned below under Parameters Tab present in VAPI-XP Test “TC1 “, make sure the order is as below



|  |  |  |
| --- | --- | --- |
| ***Parameter Name*** | ***Valid Values (sample values)*** | ***(Default value)*** |
| **IterationMode** | |  | | --- | | RUN\_ALL\_ITERATIONS | | RUN\_ONE\_ITERATION\_ONLY | | RUN\_RANGE\_OF\_ITERATIONS | | SKIP |
| **StartIteration** | 1 | SKIP |
| **EndIteration** | 1 | SKIP |
| **ExecutionMode** | |  | | --- | | LOCAL | | REMOTE | | LOCAL\_EMULATED\_DEVICE | | REMOTE\_EMULATED\_DEVICE | | GRID | | PERFECTO | | MOBILE | | SEETEST | | MOBILECENTER  MINT  SAUCELABS  BROWERSTACK | | SKIP |
| **MobileToolName** | |  | | --- | | DEFAULT | | APPIUM | | REMOTE\_WEBDRIVER | |  |
| **MobileExecutionPlatform** | |  | | --- | | ANDROID | | IOS | | WEB\_ANDROID | | WEB\_IOS | |  |
| **DeviceName** |  | SKIP |
| **Browser** | |  | | --- | | CHROME | | FIREFOX | | GHOST\_DRIVER | | HTML\_UNIT | | INTERNET\_EXPLORER | | OPERA | | SAFARI | | SKIP |
| **BrowserVersion** | SKIP | SKIP |
| **Platform** | WINDOWS | SKIP |

Here if Default Value “SKIP” means, it would take up all default values as per CRAFT Framework, it will refer Global Setting.Properties and Mobile Automation Setting.Properties

* Navigate to Eclipse (CRAFT) project and look up for RunQC.bat under Miscellaneous Folder. Edit accordingly.

*C -Drive:*

SET JAVALIBS=C:\External\_Jars

C:

cd "C:\Projects\My Projects\CRAFT\target\test-classes\"

java -cp ".;%JAVALIBS%\\*" runners.QcTestRunner %\*

*D -Drive:*

SET JAVALIBS=C:\External\_Jars

D:

cd "D:\Projects\My Projects\CRAFT\target\test-classes"

java -cp ".;%JAVALIBS%\\*" runners.QcTestRunner %\*

**Note:** Since its MAVEN, all jars referred will be from MAVEN repo, it would be difficult to choose CRAFT specific jars & also bad idea to choose all jars to refer in classpath. Please have all required jars in separate folder and mention in RunQC.bat file as above. If Framework specific jars are required kindly download it from <http://acoetoolkit.cognizant.com/setup/CRAFTVersions/V3/Addons/External_Jars.zip>

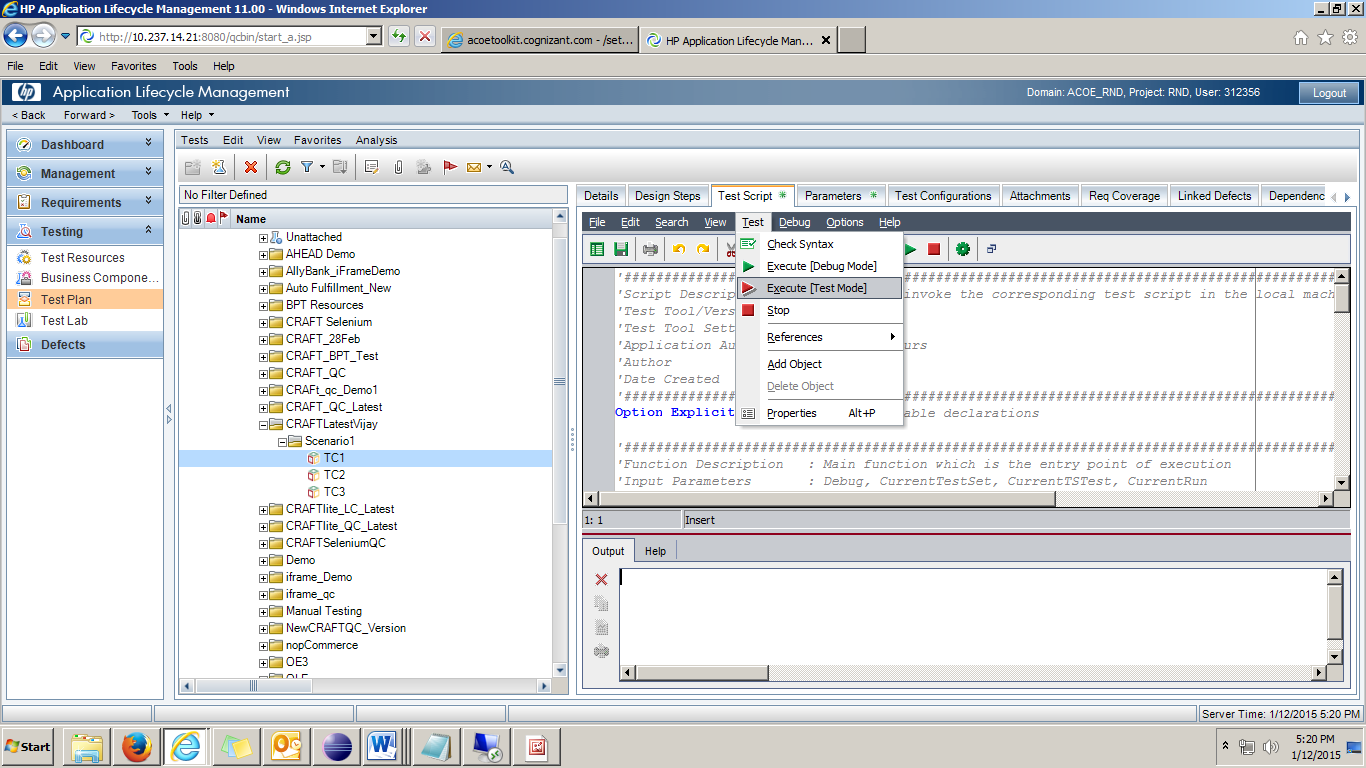
* Traverse to **Test Lab** Menu and create a New Folder under **Root** Folder. For better readability you can name as Eclipse Project Name example : mCRAFT
* Now create a new **Test Set** and make sure the name of the Test Set should be referred to Run Configuration value present in Global settings. properties file CRAFT Framework for example : Test Set Name : **Regression**
* Provide your Craft Framework Eclipse workspace path into Framework Path field present in Test Set
* Or Please provide the Eclipse project path under VAPI XP test script,

**strRelativePath = "D:\Projects\My Projects\CRAFT "** 'Customize this line as required

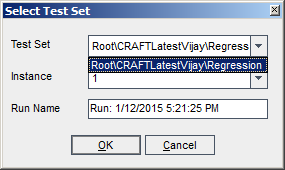
* Now move on to **Execution Grid Tab** in **Test Set** and try importing “TC1” Test by selecting Select Test button
* After importing Click on Run button present at the top of Execution Grid menu Tab
* Click on **Run Tests Locally**
* You can able to view Automatic Runner and click on Run or Run All and you can able to view execution window opening up which will invoke QcDriverScript.vbs present under supportlibaries package in CRAFT framework
* Now you can able to view the execution over browser/device instance
* After the execution “Test Status” will be updated in status field present in Execution Grid Tab
* Results will be attached for each **Test Run** : Double click on Test instance and you can able to view Test Instance in detail and navigate to Attachments Tab you can view CRAFT Framework results attached to it

## **12.2. Execution in Test Plan**

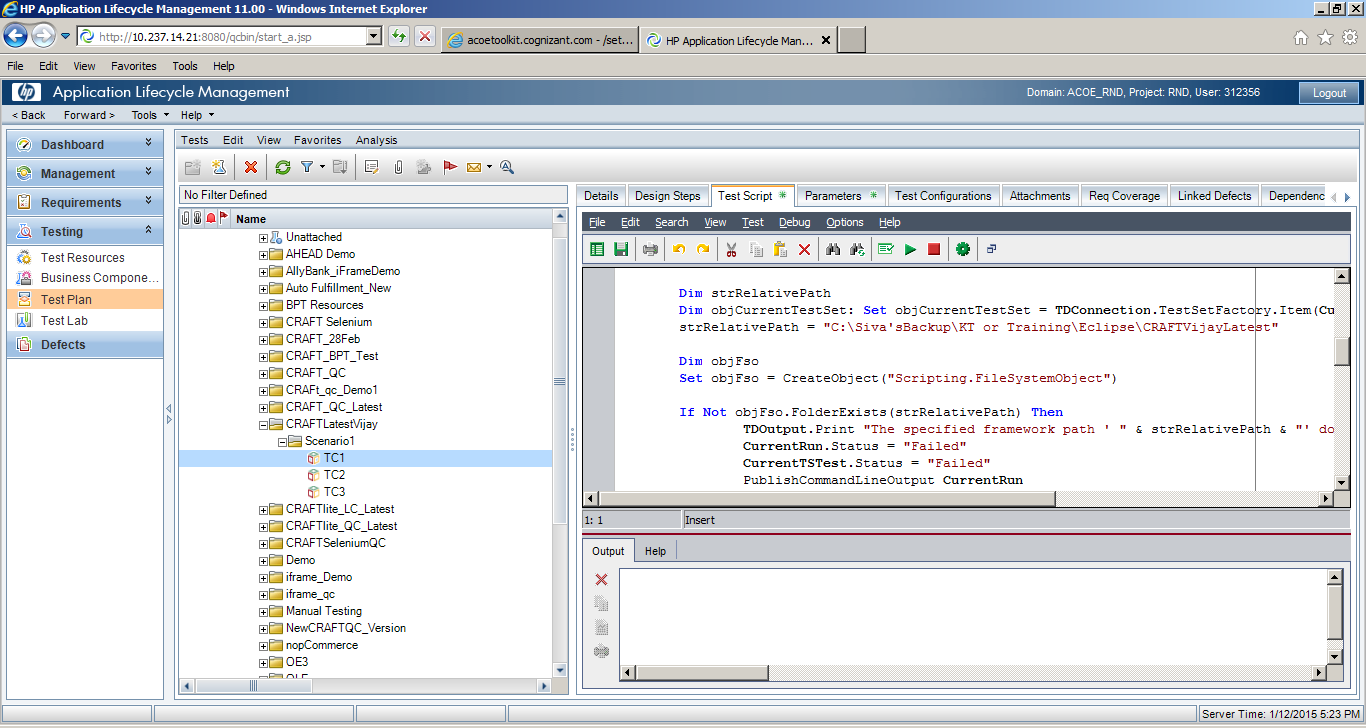
* Please Navigate to Test Plan, go the respective VAPIXI Script (ex: TC1), click on Test Script Tab.
* Now click on Test in Menu Bar, click on Execute [Test Mode].



* Select the appropriate Test Set from Test Lab as below and click on OK.( Make sure that before you execute the Test Script should be mapped in any of Test Set in Test Lab)



* Now the execution starts. If it fails, please check the error from Output Memo field.



## **12.3.** **Executing QC from local without opening QC**

We can also trigger the execution from Local without Opening the QC. Please download the additional files from <http://acoetoolkit.cognizant.com/setup/CRAFTVersions/V3/Addons/QCIntegrationMaven.zip>

Below are the steps to execute from Local,

* Please extract and place entire supportlibraries folder under src/test/resource of CRAFT Framework.
* Also place Allocator\_QC.vbs & InitScript\_QC.vbs under OtherRunners package of Framework.
* Please follow the 11.1 Section for all basic integration steps.
* Once the basic integration is working, we have to modify/ create the Sheet in Run Manager.xlsm very similar to QC Test Set and also it should have same test cases as per the execution Grid available in QC/ALM.
* Since the Control is from local, we should do a replica of Test Set into Run manager there by giving control which test case to be executed from Local Itself.

Ex: QC has Regression Test Set and have 5 test cases, Run manager should also have

* However The Test parameters will be taken from respective VAPI-XP test script parameters.
* Test Parameters can also be controlled but need Customization, please reach us if you require this feature.
* Open InitScript.vbs, provides the values for Test Set Path and Test Set name.
* Provide all the details QCURL, username, password, project and domain.
* Double Click the InitScript.vbs, it should start executing test cases from QC and upload result to QC/ALM.
* This can be triggered from Jenkins too, please refer to 12.5 Section.

# **13.** **Jenkins Integration**

## **13.1.** **Pre-Requisites for Jenkins CRAFT Integration**

* Please download JAVA jdk from <https://java.com/en/download/>.
* Please download Apache Maven https://maven.apache.org/download.cgi and configure accordingly.
* Please download the Jenkins and configure accordingly.

**Note:** The pre-requisites for above is out of scope for this doc.

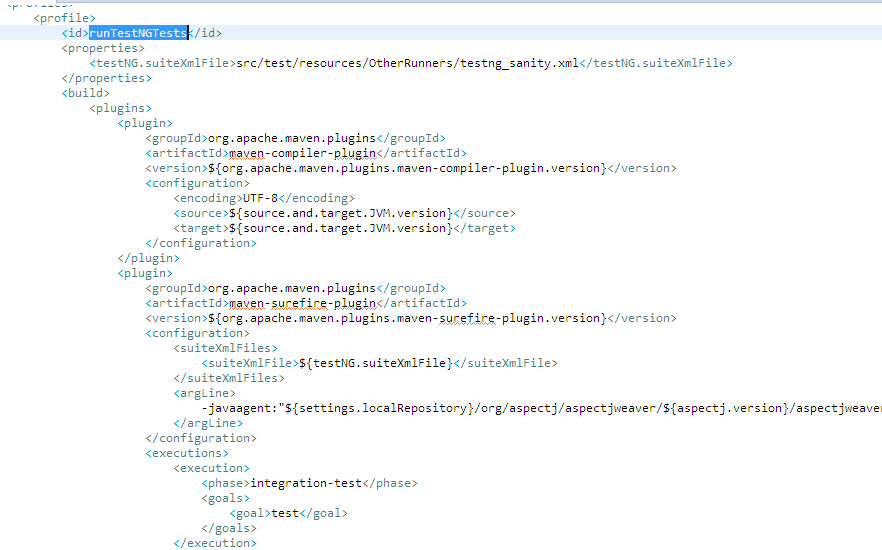
## **13.2.** **Executing Allocator from POM file**

* We can create separate profile for running Allocator.java.
* We would need mojo plugin as mentioned below to trigger the java file.
* We can run below as **mvn clean test –P runAllocator** from Jenkins, command prompt.



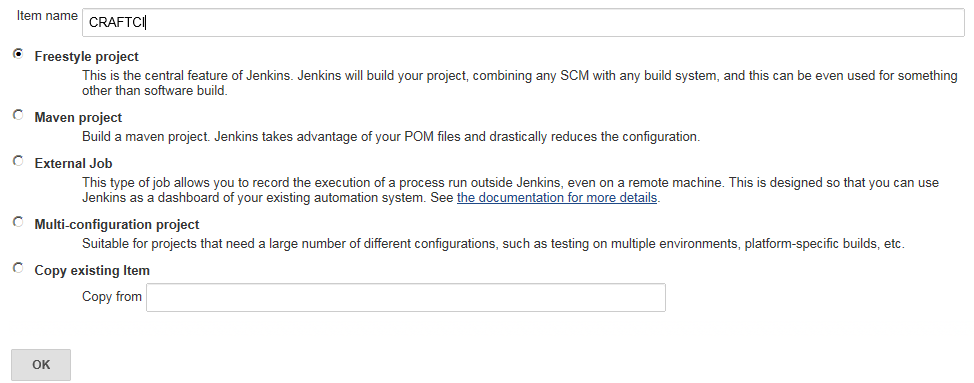
## **13.3. Executing TestNG from POM file**

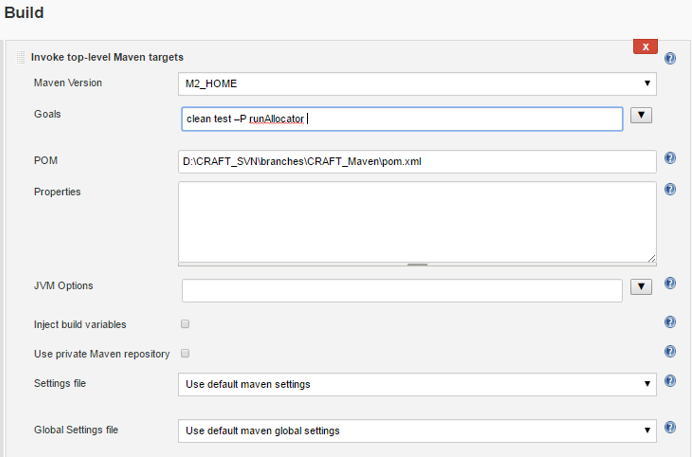
* We can create separate profile for running TestNG Suite files.
* We would need surefire plugin as mentioned below to trigger the TestNG files.
* We can run below as **mvn clean test –P runAllocator** from Jenkins, command prompt.



**Note:** Other configuration & tags of MAVEN explanation is out of box for this doc. Please refer <http://maven.apache.org/guides/>

## **13.4.** **Invoking from Jenkins**

* All POM file configuration can now be executed from Jenkins
* Now add a new job and call it **CRAFTCI**, select **Build a free-style software project**.  
  
* After the job is created, you will be on the job configuration page.
* Add a new **Build** step and select **Invoke top-level Maven targets**

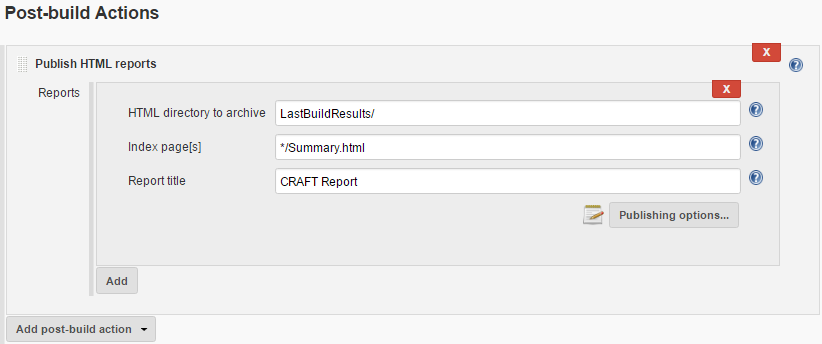


* Select the Maven version which we have already configured it on the Configure System page.
* Click on Advanced, you can give the path of pom.xml
* Click on Build Now, this will trigger pom file and trigger the respective profile for Execution and generate reports

## **13.5.** **Report update to Jenkins from CRAFT**

There is also an option to upload the latest Summary report back to Jenkins. Below are the steps to achieve this.

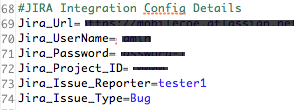
* Download the HTML Plugin for Jenkins from
* Once Download Restart Jenkins as required.
* From CRAFT, please change **UploadSummaryToJenkins=True** in Global Settings.properties.
* From Jenkins, under Post build actions please browser the Workspace/Project path as below,



# **14.** **JIRA Integration**

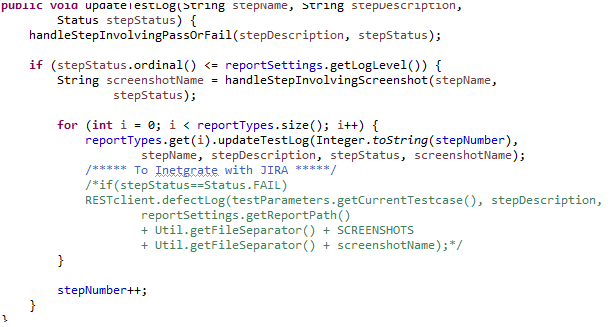
## **14.1. Global Settings**

* Mention the Jira URL, username, password and other required details.



## **14.2. Updating the JIRA API’s**

* Under *com.cognizant.framework* package, open Report.java and uncomment the lines as mentioned in the below screenshot. Execute the script and check the bug in Jira portal.

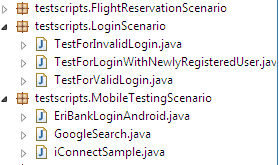


# **15. C.R.A.F.T – TestNG Implementation**

We can execute the CRAFT through TestNG also. Below are the major factors while executing the CRAFT through TestNG framework.

## **15.1.** **One-to-one Mapping for TestNG classes**

* We should have one-to-one mapping for every test case defined in Scenario Sheet.
* The template is available under miscellaneous folder, please copy and replace name with Test case name.





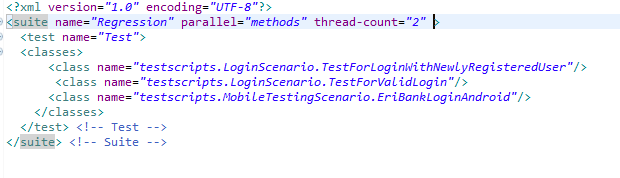
* Please note that , Run Manager will not be considered while working with TestNG way fo CRAFT as TestNG drives the execution.
* All the Test parameters require to execute is mentioned in respective Data Provides of each TestNG Class as below,



* We can now Run/Debug individual Test case by right cliking on TestNG created class and **Run As -> TestNG Test**.

## **15.2.** **One-to-one Mapping for TestNG classes**

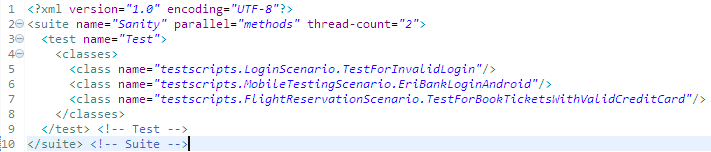
* We have TestNG XML suite file in CRAFT Framework, please call the test classes to be executed as below,



* To achieve parallel execution while executing through TestNG, we can increment the thread-count.

## **15.3.** **Batch Execution with TestNG**

* Batch execution within CRAFT can also be achieved by TestNG XML Suite file.
* We have TestNG suite files under root folder.
* Here is basic sample of TestNG Suite file

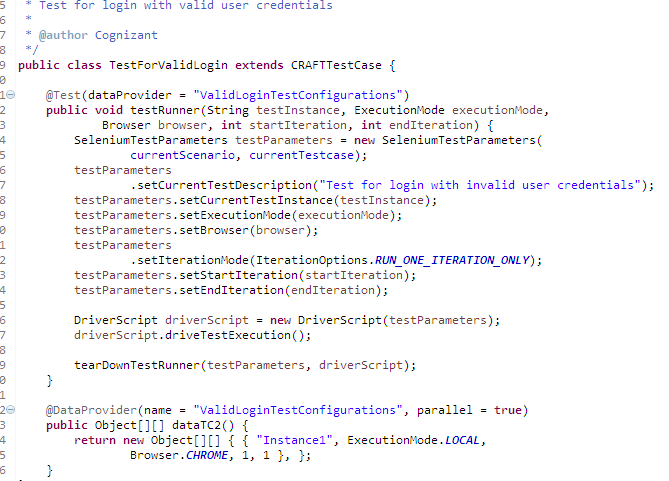


* The above example will execute all three classes mentioned. For Parallel execution thread-count can be increased.

## **15.4.** **Parameterization of Test Parameters**

* Within CRAFT we need Test parameters for execution to happen. These parameters are nothing but ExecutionMode, MobileExecutionPlatform etc., Since here TestNG is driving the execution Run manager will be utilized there by we would need to pass these parameters to TestNG class. We can pass it in twos ways

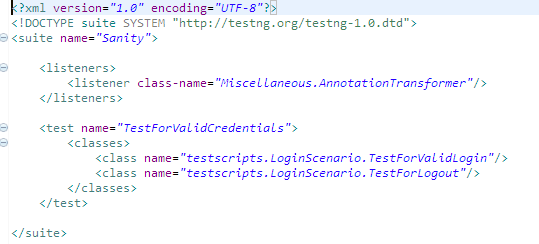
1. *From Data Providers:*
   1. Data providers are used to externalize or pass params to TestNG class. In CRAFT every TestNG Class will be associated with a Data provider and within Data provider we will have Test parameters defined. This Parameters will in turn being passed to TestNG class for driving the execution. Below is the sample example,



1. *From TestNG Suite file as Parameters:*
   1. We can also pass Parameters from TestNG xml file by defining parameters and value’s to it.
   2. Each Test tag can have Parameters being passed to respective TestNG methods.
   3. Below is the Sample,

### **15.5.1.** **Rerunning the Failed Test Case at that moment**

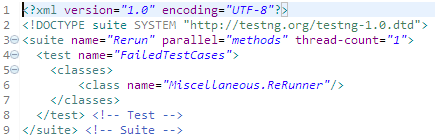
* This method will rerun the failed test case once respective test fails.
* Two Java files are involved for this method of rerun.
* Retryanalyser.java, where you define how many times it has to rerun.
* The other AnnotationTransformer.java which implements IAnnotationTransformer whose major task is to create annotations at runtime.
* In TestNG xml file AnnotationTransformer should be called as listener, below is sample



* Sample is also available under **Miscellaneous** folder.

### **15.5.2. Rerunning the Failed Test case at the end of execution**

* This method will rerun the failed test case once after entire execution is done.
* One java file is required here to achieve it, Rerunner.java.
* failed-testng.xml, is a default TestNG suite file where all failed cases are updated within it.
* To achieve this method, we just have to invoke above Rerunner file after the execution completes.



* We can use Jenkins to rerun failed test cases at the end automatically, the failed test cases are automatically maintained in an XML file which gets generated after running your build. The xml that contains list of failed test cases can be run through Post Build action of Jenkins.
* For running TestNG Suite from Jenkins, please refer to 12.4 Section.

# **16.** **Modular Driven Approach within CRAFT**

Modular Driven Approach here is formally known as CRAFTLite. CRAFTLite has been bundled within CRAFT. By changing flag we can drive the execution using Keyword Driven Approach (CRAFT) or Modular Driver Approach (CRAFTLite)

## **16.1.** **Download Modular Driven Samples**

* Since the Demand for Modular Driven Approach is less the Datatables & testscripts by default defined in CRAFT format. Please download Datatables & testscripts for Modular Driven Approach from

## **16.2.** **Configuration for Modular Driven Approach**

* Please change the Approach=ModularDriven in Global Settings.properties



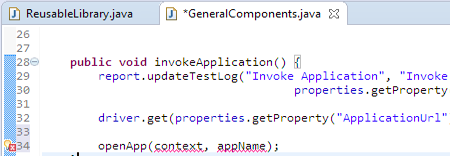
* Replace all downloaded DataTables & Test Scripts within the Framework.
* All Details of Modular Approach is mentioned here

# **17.** **Leverage the Reusable functions with Framework**

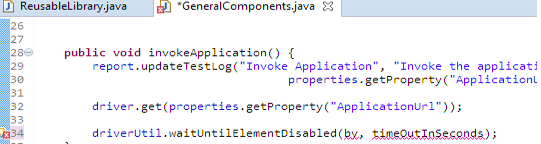
Framework provides reusable functions which can be called directly into your Business components. All Reusable functions are available in ReusableLibarary.java.

## **17.1.** **Working with Selenium, Appium & Perfecto Reusable Functions**

* One can directly access resuable functions from Business Component, all details like which ExectionMode, Paramerts has been mentioned for ease of use.



* We have also class called WebDriverUtil.java, this has commonly used selenium resusbale functions, this can be directly accessed by calling driverUtil. In any Business component java file.



## **17.2.** **Working with additional Data Access Functions**

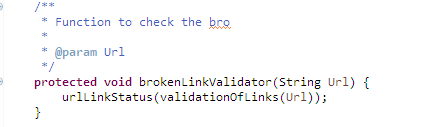
These are few additional Data Access functions designed for any specific requirements.

* **dataTable.putData(“sheetName”,” fieldName”):** To fetch the value from specified Sheet name & Column Name.
* **dataTable.getExpectedResult(“fieldName**”): To get the expected result from parameterize checkpoints sheet.
* **dataTable.getDataWithSubIteration(“sheetName”,”fieldName”,”subIteration”):** To get value from specified subiteration row.
* **dataTable.getData(“sheetName”,”keys”):** To get the values of all columns mentioned on keys (keys are array of column names. This will return list of all values of column
* **datatTable.getDataWithSubIterationTDID(“testcase”,”sheetName”,”fieldName”,”subIetration”):** To get the value of some other test case with specified subiteration.

## **17.3.** **Working with Broken Links**

Framework provides reusable functions for get the broken link response.

* One can directly access broken link reusable function from Business Component.



* Call above funcion within Business Component