Purpose

The purpose of this document to provide insight on

1. ALM Driven test execution
2. Integration with framework
3. Utilities created for auditing of ALM test case mapping with Framework test scripts

Workflow

The intent of this development is to provide User with the feature to setup the Test Lab for the planned automated test cases to be executed with the single Jenkins Job.

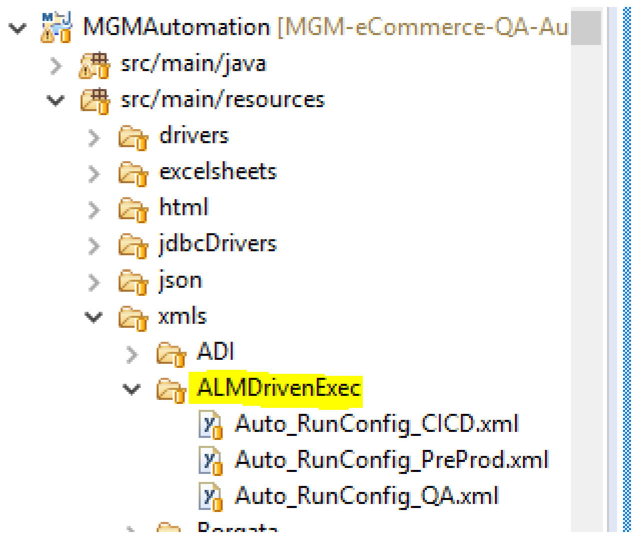
- User sets up the ALM Test Lab by pulling test cases from WhatsAutomated Folder planned for automated regression test execution.
- User need to setup Sanity as one of the Test Lab Folder and Test Set for it in the Test Lab.
- User triggers the MultiJob from Jenkins, which in turn:
    - Pulls the code from GIT Repository
    - Build the code
    - Triggers Child Jobs in below phases:
        - Phase 1 - Generate TestNG XMLs
            - This job performs following:
                - Pulls the list of Test Cases setup in Test Lab
                - Captures the mapped Test Script Names to these Test Cases
                - Generates TestNG XMLs for the Test Scripts captured
                - For each Test Set in Test Lab, one TestNG XML is generated
                - Once new TestNG XMLs are generated, Job commits these new XMLs back to Git
                - For Sanity, the TestNG xml name will be suffixed with **_00** (i.e. TestNGxml_00.xml)
                - For Regression Test Sets, the TestNG xml names will be in sequence as (TestNGxml_01.xml, TestNGxml_02.xml, TestNGxml_03.xml and so on depending on number of Test Sets setup)
        - Phase 2 – Execute Sanity Test
            - This job is setup for TestNG xml generated for Sanity Test (TestNGxml_00.xml) and performs following:
                - Pull the code from GitHub
                - Triggers the Job
                - Update results to ALM Test Lab in respective Test Set
                - Post Test Results to Grafana
        - Phase 3 – Execute Regression Test
            - This phase is associated to child jobs associated to regression TestNG xmls generated in Phase 1 (TestNGxml_01.xml, TestNGxml_02.xml, TestNGxml_03.xml and so on). During this Phase, all child jobs

associated to regression test sets are executed and following actions are performed:

- Pull the code from GitHub
- Triggers the Job
- Update results to ALM Test Lab in respective Test Set
- Post Test Results to Grafana

Framework Configurations

User will be required to configure Auto_RunConfig_<Environment>.xml file under /src/java/resources/xmls/AlmDrivenExec



```
MGMAutomation [MGM-eCommerce-QA-Au
  src/main/java
  src/main/resources
    drivers
    excelsheets
    html
    jdbcDrivers
    json
    xmls
      ADI
      ALMDrivenExec
        Auto_RunConfig_CICD.xml
        Auto_RunConfig_PreProd.xml
        Auto_RunConfig_QA.xml
```

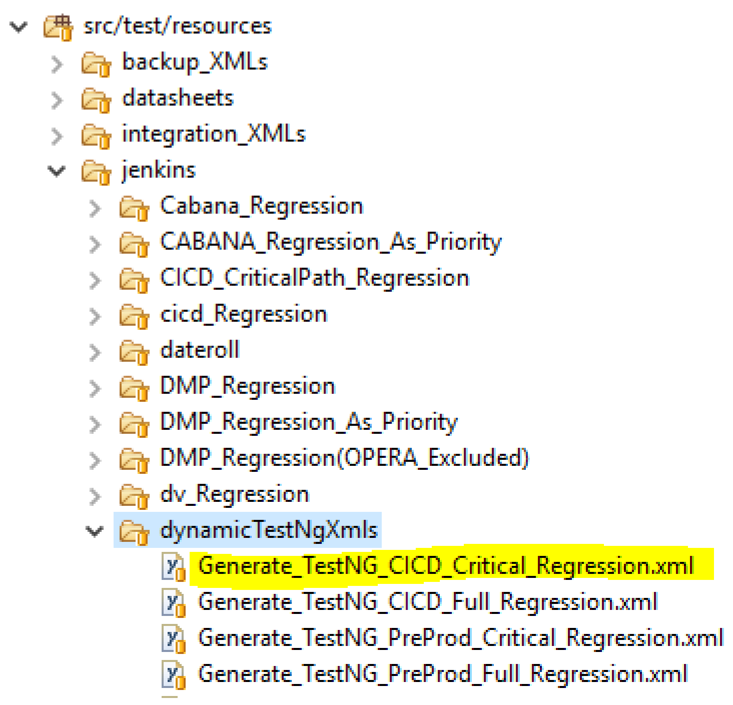
XML Structure: User needs to update XML for below Tags highlighted in Green below:

1. Release
2. TestNGParameters
3. ALM
4. Grafana

```
<?xml version="1.0" encoding="UTF-8"?>
<AutoRunConfig>
	<Release>SR_21</Release>
	<Cycle>Cycle-1</Cycle>
	<TestNGParameters environment="QA"
		runLocation="gridOnCloud"
		browserUnderTest="chrome"
		browserVersion=""
		operatingSystem="windows 7"
		logLevel="INFO"
```

```
            reportToMustard="false"
            testPlanFolder=""
            testLabFolder=""/>
      <ALM RunTestFromALM="true"
            Host="vmalmapp01p"
            Port="8080"
            UserName="user"
            Password="password"
            Domain="MGMIT"
            Project="MGMIT_MAIN"
            TestLabFolder="QA/Regression/CriticalPath/Automated"
            TestSetName=""/>
      <Slack></Slack>
      <Jira pushJiraIssuesToGrafana="true"
            jiraUrl = "https://mgmridev.atlassian.net"
            userName = "gsoni@mgmresorts.com"
            password = "lElhpLvxP93xCLSay9FG9255"
            project = "QM"
            influxDbEndPoint="http://10.191.112.198:8086"
            influxDbDatabase = "Jira"
            influxDbUser = "jmeter"
            influxDbPassword = ""/>
      <Grafana PushResultsToGrafana="true"
            EndPoint="http://10.191.112.198:8086"
            Database="Selenium"
            UserName="jmeter"
            Password=""/>
</AutoRunConfig>
```

Note: Though the Environment and ALM → Test Lab Folder is given in this Configuration, it is taken from TestNG XML that triggers the code to generate TestNG XMLs.



```
src/test/resources
  backup_XMLs
  datasheets
  integration_XMLs
  jenkins
    Cabana_Regression
    CABANA_Regression_As_Priority
    CICD_CriticalPath_Regression
    cicd_Regression
    dateroll
    DMP_Regression
    DMP_Regression_As_Priority
    DMP_Regression(OPERA_Excluded)
    dv_Regression
    dynamicTestNgXmls
      Generate_TestNG_CICD_Critical_Regression.xml
      Generate_TestNG_CICD_Full_Regression.xml
      Generate_TestNG_PreProd_Critical_Regression.xml
      Generate_TestNG_PreProd_Full_Regression.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Opera EOD Job Report Check" parallel="methods" thread-count="1"
      data-provider-thread-count="1" verbose="1">
      <!-- <listeners>
            <listener class-name="com.orasi.utils.listeners.TestListener" />
            <listener class-
name="com.orasi.utils.listeners.TestNGCustomEmailReportSummary" />
            <listener class-
name="com.orasi.utils.listeners.TestNGCustomEmailReportDetail" />
            <listener
                  class-
name="com.orasi.utils.listeners.TestNGCustomEmailReportDetailTrace" />
      </listeners> -->
      <parameter name="browserUnderTest" value="api" />
      <parameter name="environment" value="QA" />
      <parameter name="runLocation" value="grid1" />
      <parameter name="reportToMustard" value="false" />
      <parameter name="browserVersion" value="" />
      <parameter name="operatingSystem" value="windows 7" />
      <parameter name="application" value="OPERA" />
      <parameter name="logLevel" value="INFO" />
      <parameter name="testLabFolder"
value="QA/Regression/CriticalPath/Automated" />

      <test name="TestDynamicTestNGxmlCreation">
            <classes>
                  <class
```

```
        name="com.mgm.generateTestNGxml.DynamicTestNGxmlGenerator" />
            </classes>
        </test>
</suite> <!-- Suite -->
```

Based on the *environment* parameter, code will pickup the respective Auto_RunConfig_< *environment* >.xml file for rest of the parameter. And based on *testLabFolder* parameter, code will access ALM for the Test Lab Folder and captures all the Test Instances to be executed and generates TestNG XMLs accordingly.

Jenkins Job Setup

Multiple MultiJobs are setup in Jenkins based on Environment and Regression type combinations:

Environments:

- CICD
- QA
- PreProd

Regression Types:

- Full
- Critical

Refer below embedded Excel file for details of Jobs setup.

DynamicJobSetupDetails.xlsx

<u>Note:</u>

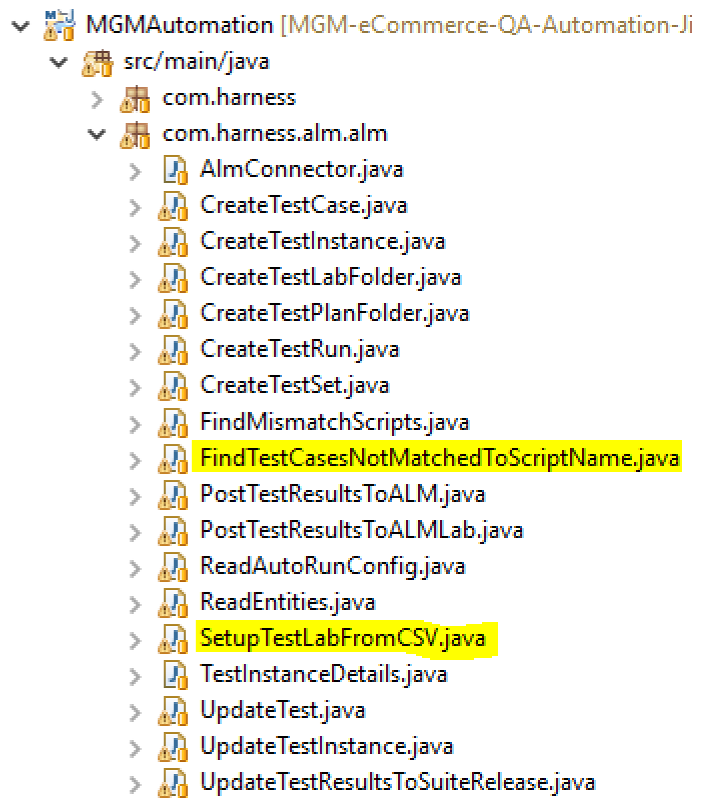
The Job setup for generating TestNG XMLs based on ALM Test Lab setup requires pushing newly generated TestNG xmls back to Git Repository as part of Post Steps in Job. Below is the command line code for it:

```
cd ${WORKSPACE}
git config --global user.name "GitUser"
git config --global user.email "GitUser@email.com"
git status
git add "src/test/resources/jenkins/dynamicTestNgXmls/TestNGxml_*.xml"
git add "src/test/resources/testInstances.csv"
git commit -m "Added file with automated Jenkins job"
git push https://<GitUser>:<PersonalAccessToken>@github.com/MGMResorts/MGM-eCommerce-QA-Automation.git HEAD:<Git Branch Name>
```

Utilities

Following Utilities are created to speed up the setup of ALM Test Lab:

Sr #	Utility	Purpose	Package / Class Name
	Setup ALM Test Lab	This utility sets up Test Lab from the inputs provided from CSV file. Sample CSV file:	com.harness.alm.alm.SetupTestLabFromCSV.java
	ALM Audit	Finds ALM Test Cases not having correct Script Names and also finds Test Scripts not mapped to any Test Case in ALM	com.harness.alm.alm.FindTestCasesNotMatchedToScriptName.java



```
MGMAutomation [MGM-eCommerce-QA-Automation-Ji
  src/main/java
    com.harness
    com.harness.alm.alm
      AlmConnector.java
      CreateTestCase.java
      CreateTestInstance.java
      CreateTestLabFolder.java
      CreateTestPlanFolder.java
      CreateTestRun.java
      CreateTestSet.java
      FindMismatchScripts.java
      FindTestCasesNotMatchedToScriptName.java
      PostTestResultsToALM.java
      PostTestResultsToALMLab.java
      ReadAutoRunConfig.java
      ReadEntities.java
      SetupTestLabFromCSV.java
      TestInstanceDetails.java
      UpdateTest.java
      UpdateTestInstance.java
      UpdateTestResultsToSuiteRelease.java
```

Note: All ALM Integration related code is under below packages:

1. com.harness.alm.alm
2. com.harness.alm.infrastructure