**Context**

AquaMark platform provides enterprise users access to Mozark device cloud, which can be used for executing automated functional tests. These results can be viewed in highly customizable dashboards which would help users navigate functional test results by device make-model, OS version and app version.

AquaMark support automation tests of Android Native tests in Appium Java TestNG framework.This section describes how to configure, package, and upload your Appium Java TestNG tests for Android to our Device Farm using the AquaMark portal. Appium is an open source tool for automating native and mobile web applications. For more information, see [Introduction to Appium](http://appium.io/docs/en/about-appium/intro/) on the Appium website.

For a sample app and test package, please see our Github link

**Version Information**

* Currently we support Java 8 for running Appium Tests
* Mozark device farm is integrated with Appium server version x.x

**Steps for scripting**

* Step 1: Declare necessary dependencies in your Maven project
* Step 2: Write test script and verify on real device locally
* Step 3: Modify global variable values for AquaMark platform

**Step 1: Declare necessary dependencies in your Maven project**

Create a Maven project. Add following dependencies in pom.xml.

Dependency for TestNG jars:

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.0.0</version>

<scope>test</scope>

</dependency>

Dependency for Selenium jars:

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

Dependency for Appium jars:

<dependency>

<groupId>io.appium</groupId>

<artifactId>java-client</artifactId>

<version>7.0.0</version>

</dependency>

<dependency>

<groupId>net.lightbody.bmp</groupId>

<artifactId>browsermob-core-littleproxy</artifactId>

<version>2.1.0-beta-3</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

</dependencies>

**Step 2: Write test script and verify on real device locally**

In the Eclipse file explorer, expand “src/test/java” and expand the package.Main Test class should include Test in its name. (eg. AndroidTest.java)

Write the test cases in main test. Include the following codes to make sure the output is in the desired format

|  |  |  |
| --- | --- | --- |
| Sl # | Purpose | Code |
| 1 | Declare Environment variable to connect to devices, store images, assign ports | IMAGE\_PATH :- (For Screenshots to store in Server)  TEST\_ID :- (For different tests different ID)  DEVICE\_SERIAL :- (Device Name)  APPIUM\_PORT :- (Appium port for particular Device)  SYSTEM\_PORT :- (System port for particular Device)  For executing the test on local device, hardcode the following values:  private static int APPIUM\_PORT = 4723;  private static int SYSTEM\_PORT = 8200;  private static String DEVICE\_SERIAL = "ce10171ac0a0cb2b04";  // change the device serial here to run in local system |
| 2 | Start Appium server through URL | http://127.0.0.1:" + APPIUM\_PORT + "/wd/hub |
| 3 | Store APK in Resources folder and Declare APK name in the script with version in given format (required for Dashboarding) | private static String APK\_PATH = "src/test/resources/Test\_v1.1.2.apk";  private static String APK\_NAME = "Test\_v1.1.2.apk"; |
| 4 | To log the statements in the Appium output log, create "log4j.properties”in src/test/resources and paste the given code | #Set level  log4j.rootCategory= debug, console, file  # Appender which writes to console  log4j.appender.console=org.apache.log4j.ConsoleAppender  log4j.appender.console.layout=org.apache.log4j.PatternLayout  log4j.appender.console.layout.ConversionPattern=MOZARK\_METRICS - DATE:%d{yyyy-MM-dd HH:mm:ss.SSS} - %m%n  # Appender which writes to a file  log4j.appender.file=org.apache.log4j.RollingFileAppender  log4j.appender.file.File=application.log  # Defining maximum size of a log file  log4j.appender.file.MaxFileSize=10mb  log4j.appender.file.MaxBackupIndex=10  log4j.appender.file.layout=org.apache.log4j.PatternLayout  log4j.appender.file.layout.ConversionPattern=MOZARK\_METRICS - DATE:%d{yyyy-MM-dd HH:mm:ss.SSS} - %m%n  log4j.appender.file.Append=false |
| 5 | Print logger for Test case and Sub test cases in this format | MOZARK\_METRICS - DATE:2019-10-31 15:39:04.812 - TestCase:Test Case Name-SubTestCase:Sub Test Case Name-SubCaseResult:PASSED/FAILED  MOZARK\_METRICS - DATE:2019-10-31 15:39:16.884 - TestCase:Test Case Name-SubTestResult:PASSED/FAILED |
| 6 | Capture screenshot before/after executing SubTestCases image name and path should be | Format: TestCaseName\_SubTestCaseName, IMAGE\_PATH  Example:  screenshot.captureScreenshot("Click on Login Text\_Before Click on Login", IMAGE\_PATH); |

Once the script is complete execute it locally and verify if the output is in the desired format. For a sample script and output log, please refer our Github link

**Step 3: Modify the script for Mozark platform**

Once the script has been verified through local execution, modify the below global environment variables to make is executable in the Aquamark platform

// the hardcoded values for the following variables with the following

//private static int APPIUM\_PORT = 4723;

//private static int SYSTEM\_PORT = 8200;

//private static String DEVICE\_SERIAL = "ce10171ac0a0cb2b04"; // change the device serial here to run in local system

//replace with following values

private static String DEVICE\_SERIAL = System.getenv("DEVICE\_SERIAL");

private static String APPIUM\_PORT = System.getenv("APPIUM\_PORT");

private static String SYSTEM\_PORT = System.getenv("SYSTEM\_PORT");