Evaluation Report - Selenium Grid for Mobile using Appium

Global Mobile Testing CoE

The purpose of the document is to create a report based on the mobile specific features available in Selenium Grid using Appium

7/07/2015

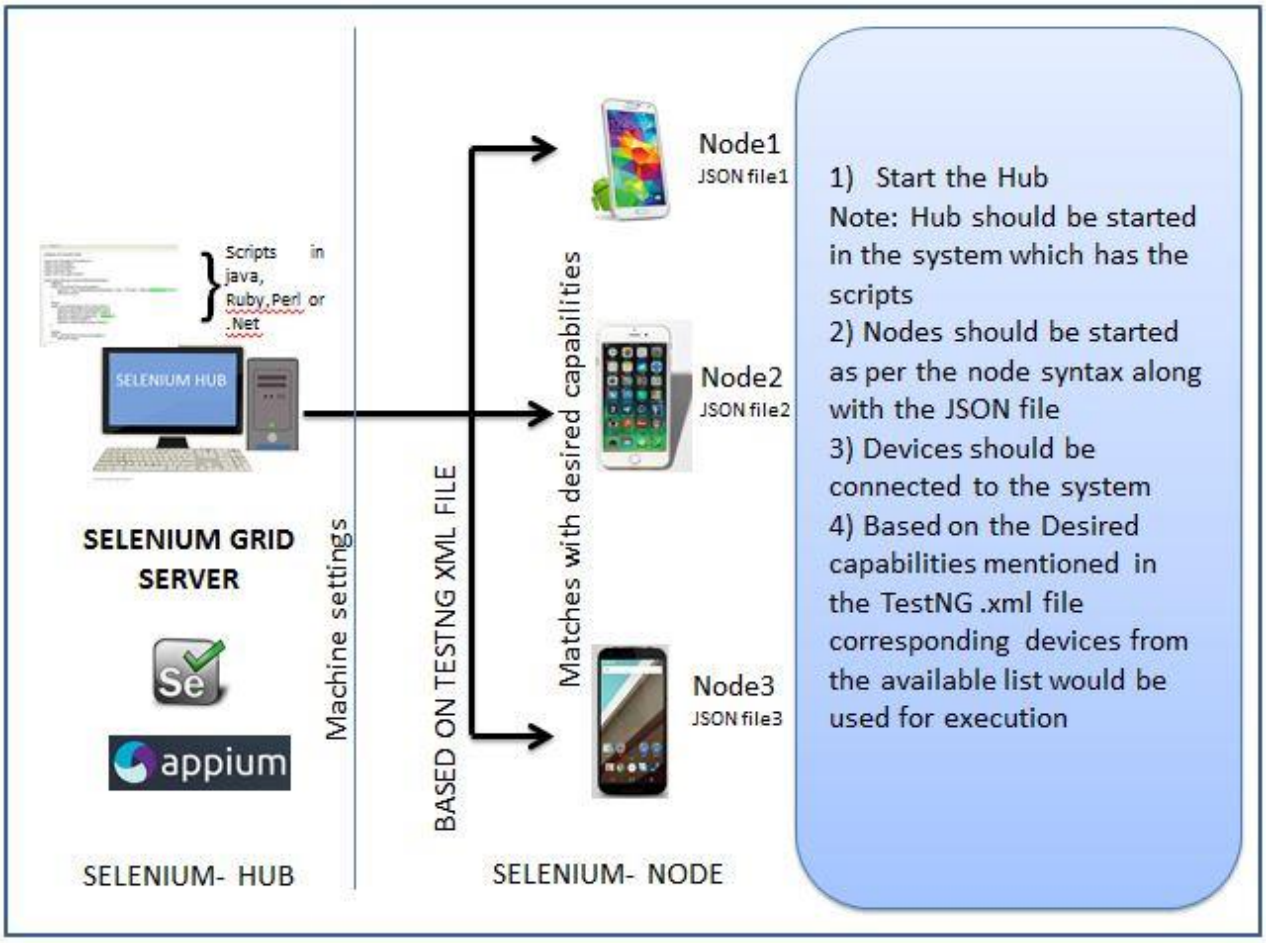
Table of Contents

1. Summarized View

Overview of the tool

Selenium Grid is a part of the Selenium Suite which is used for running multiple tests across mobile devices of different operating system in parallel using Appium. Essentially, Selenium-Grid support distributed test execution. It allows for running your tests in a *distributed test execution* environment [TestNG setup]

2. Architecture for Selenium GRID using Appium in mobile Devices



3. Pre-requisite of GRID with respect to Mobile scripts

To Run mobile scripts parallel we need Selenium + Appium setup.

Pre-requisite for Selenium:
Selenium standalone jar file

Pre-requisite for Appium
In Windows (only Android)

- Java
- Android SDK
- Eclipse
- Android_Home , Environment variables
- Appium.exe

In Mac(for iOS & Android)

- Java
- Android SDK
- Android_Home , Environment variables
- Webkit debug proxy
- Developer device(UDID added to the developer profile)

TestNG setup

4. Parallel Execution in Appium

In general Parallel Execution of mobile script is possible only in the Case of Android [Running more than One Android device at a time]. But in the case of iOS, MAC development environment supports only one mobile device connected at a time . So Parallel execution is not possible between IOS devices

Below are the sets to proceed with Parallel Executions of Mobile Devices

I. Configuring Hub
II. Configuring Nodes
III. Configuring JSON file
IV. Running the scripts in Parallel using TestNG framework

I. Configuring Hub:

The hub is the central point where you load your tests into. There should only be one hub in a grid. The machine containing the hub is where the tests will be run, but you will see the execution being automated on the nodes.

Procedure:

1) Download the Selenium Standalone Server by here.
2) You can place the Selenium Standalone Server .jar file anywhere in your HardDrive.The following steps will launch the hub and the node.
3) Using the command prompt, navigate to the root of Machine where the Selenium Standalone Server placed.
4) On the command prompt, type **java -jar selenium-server-standalone-X.XX.0.jar jar -role hub -hub http:// <MACHINE IP >:4444/grid/register**

The Syntax to configure a Hub

java -jar selenium-server-standalone-X.XX.0.jar jar -role hub -hub http:// <MACHINE IP >:4444/grid/register

Working Example:

java -jar selenium-server-standalone-2.45.0.jar -role hub -hub http:// 192.168.1.113:4444/grid/register

Note:

It's possible to run the script remotely from one system [machine 1] where the Test scripts are present and to the Remote system [machine 2] where the Devices are connected . But

we should start the Hub in the First Machine and should call hub address in the JSON files of the Device under test in the Machine 2

II. Configuring Node settings:

Nodes are the Selenium + Appium instances that will execute the tests that you loaded on the hub. Nodes are the actual Appium instance that drive the device execution based on the script. There can be one or more nodes in a grid. Nodes can be launched on multiple machines with different platforms and browsers

Procedure:

1) After the Hub is started , verify whether the hub is running by using a browser in your machine as below

   localhost:4444 else <MACHINE IP >:4444

We can find a console screen as above , which shows the Hub is launched successfully

2) Open the Terminal /command Prompt and type the below Node syntax as required

Syntax for Node configuration :

For IPHONE :

In Mac :

node /Applications/Appium.app/Contents/Resources/node_modules/appium/bin/appium.js -a <MACHINE IP ADDRESS > -p <APPIUM PORT value > --nodeconfig "<Path for the JSON file >" --udid <UDID Of the device under test > --bootstrap-port<Boot strap port value >

Working Example code :

node /Applications/Appium.app/Contents/Resources/node_modules/appium/bin/appium.js -a 192.168.1.113 -p 4730 --nodeconfig "/Users/MobileTestingCoE/Desktop/KARTHIK/Grid/Iphone4730.json" --udid 7cfxxxxxxxxxxxxxxxxxxxxx74e --bootstrap-port 2257

Note: In case of Iphone , we need to specify Two port values as below

I. Appium port
II. Boot Strap Port value

The Appium port value mentioned in the Syntax should be the same inside the JSON file too . Both Appium port and Bootstrap port values should be unique across all the nodes else we can expect some error through

For Android :

In Mac :

node /Applications/Appium.app/Contents/Resources/node_modules/appium/bin/appium.js -a <MACHINE IP ADDRESS > -p <APPIUM PORT value > --nodeconfig "<Path for the JSON file >" --udid <UDID Of the device under test > --bootstrap-port<Boot strap port value > --chromedriver-port <Chrome port >

Working Example Code :

node /Applications/Appium.app/Contents/Resources/node_modules/appium/bin/appium.js -a 192.168.1.114 -p 4728 –nodeconfig "/Users/MobileTestingCoE/Desktop/KARTHIK/Grid/Nexus4728_Machine2.json" --udid 05cxxxxxx24e --bootstrap-port 2255 --chromedriver-port 4736

In Windows :

node appium.js -a localhost -p 4723 -a <MACHINE IP ADDRESS >--nodeconfig "<Path for the JSON file >" --udid <UDID Of the device under test > --bootstrap-port<Boot strap port value > --chromedriver-port <Chrome port >

Note: In case of Android , we need to specify three port values as below

I. Appium port
II. Boot Strap Port value
III. Chrome Port value

The Appium port value mentioned in the Syntax should be the same inside the JSON file too . All the port values should be unique across all the nodes else we will get error while configuring the node

III. Configuring JSON file:

While configuring the JSON file we should see to that it meets the standard format else , it will throw an error as “not able to load Json file ” . The most comman error which will happen in the config file is the double quotation sign . While editing the " ➔ will often turn into [“] which is an unacceptable format leads to error while launching the node

The JSON file will contain two parts , one is the Capabilities part and another is the Configuration part

I. Capabilities part
II. Configuration part

In the Capabilities part the device specific details should be mentioned and in the configuration part the corresponding Hub & port for which the node has to be mapped related details should be mentioned

Sample JSON file .

{

"capabilities":

```
[
{
"browserName": "Android",
"version":"4.3",
"maxInstances": 3,
"platform":"ANDROID",
"deviceName":"BX903K4MC3"
}
],

"configuration":
{
"nodeTimeout":"120",
"port":"4725",
"hubPort":"4444",
"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
"url":"http://10.251.32.197:4725/wd/hub",
"hub": "10.251.32.197:4444/grid/register",
"hubHost":"10.251.32.197",
"nodePolling":"2000",
"registerCycle":"10000",
 "register": true,
"cleanUpCycle":"2000",
"timeout":"30000",
"maxSession":"1"
}
}
```

We can cross check whether the JSON file created matches to the standard format by pasting it in the below

https://www.jsoneditoronline.org

Note:
"port":"4725", ➔ under configuration , is Appium port

1) Sample JSON files for Android:

For configuring the Android JSON file , we need to provide the android related capabilities as below

```
{
  "capabilities": [
   {
     "browserName":"Chrome",
     "version": "4.3",
     "maxInstances": 3,
     "platform": "ANDROID",
     "applicationName":"BX9xxxxMC3"
   }
 ],
```

Based on the above capabilities it will set the node for the respective device and in return we can find a similar set of capabilities mentioned in our script also. The reason for mentioning the capabilities twice in JSON as well as in script is , mentioning in JSON is to set the node to the corresponding device and in script is to connect to the needed Node which already has the matching capability.

Working sample Android JSON file

Android_JSON file.docx

2) Sample JSON files for Iphone:

For configuring the Iphone JSON file , we need to provide the Iphone related capabilities as below

```
{
  "capabilities": [
   {
     "browserName": "Safari",
     "version": "7.1.1",
     "maxInstances": 3,
     "platform": "MAC",
     "applicationName":"7cf80xxxxxxxxxxxxxxb6ec7879974e"
   }
  ],
```

Working sample Iphone JSON file

IV. Running Two Android devices in Parallel:

We can run two android scripts in parallel using TestNG framework . we need to configure the Test script with the xml file. As a pre-requisite we should install TestNG in the Eclipse .

Configuring TestNG file for running Android Test script parallel :

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="Selenium Grid" parallel="tests" thread-count="2" verbose="1" >

     <test name="SeleniumGrid_Sony" >

     <parameter name="browserName" value="Chrome"/>
     <parameter name="platform" value="Android"/>
     <parameter name="deviceName" value="BX903K4MC3"/>
     <parameter name="version" value="4.3"/>
     <parameter name="sleep" value="1000"/>
     <classes>
      <class name="com.cognizant.grid.SeleniumGrid_mobile_forAndroid"/>

    </classes>
```

```
    </test>

 <test name="SeleniumGrid_Nexus" >

     <parameter name="browserName" value="Chrome"/>
     <parameter name="platform" value="Android"/>
     <parameter name="deviceName" value="5ggdgrh56hgjuj"/>
     <parameter name="version" value="5.0"/>
     <parameter name="sleep" value="1000"/>
     <classes>
      <class name="com.cognizant.grid.SeleniumGrid_mobile_forAndroid"/>

    </classes>

    </test>

</suite>
```

V. Running one Android and one iOS devices in Parallel

We can't run two Iphone scripts in parallel using Appium in TestNG framework .But we run one android and one Iphone in parallel by configuring the Test script with the xml file .

Configuring TestNG file for running One Android One Iphone Test script in parallels:

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="Selenium Grid" parallel="tests" thread-count="2" verbose="1" >

     <test name="SeleniumGrid_Sony" >

     <parameter name="browserName" value="Chrome"/>
     <parameter name="platform" value="Android"/>
     <parameter name="deviceName" value="BX903K4MC3"/>
     <parameter name="version" value="4.3"/>
     <parameter name="sleep" value="1000"/>
     <classes>
      <class name="com.cognizant.grid.SeleniumGrid_mobile_forAndroid"/>

    </classes>

    </test>

 <test name="SeleniumGrid_Iphone" >

     <parameter name="browserName" value="Safari"/>

<parameter name="platform" value="MAC"/>
     <parameter name="deviceName" value="7cfxxxxxxxxxxxxxxxx879974e"/>
     <parameter name="version" value="7.1.1"/>
     <parameter name="sleep" value="15000"/>

    <classes>
      <class name="com.cognizant.grid.SeleniumGrid_mobile_forIphone"/>
     </classes>
 </test>
```

```
</suite>
```

5. Sample TestNG project to handle parallel Execution

I have attached a sample TestNG project designed for parallel Execution in the below link

TestNG_SeleniumGrid
.zip

Note: JAR files are removed from the sample project , while using the above project please add the TestNG.jar , Selenium standalone jar and other Appium related JAR files in the project Build path

Device which are used for the Analysis is as below

#	Device	OS	OS Version	Connection	Device Information
1	Iphone 5	iOS	7.1.1	USB	Non Jail-broken
2	Nexus 5	Android	5.0	USB	Non-Rooted
3	Sony Xperia Z1	Android	4.3	USB	Non-Rooted