**What is Appium and why Appium need?**

1. Appium is mobile web, native and hybrid software application test automation tool.
2. It is open-source software automation tool which is useful to automate android and IOS platform apps.
3. Most important thing is: Appium is “cross-platform” automation tool so you can write software automation tests against iOS and Android (multiple platforms) using same API.
4. “cross-platform” enables you large amount of code reuse between iOS and Android test suites.
5. Appium support software test automation on emulator or simulator and physical mobile devices.
6. Appium is developed on few key philosophy points: 1. should not have to recompile your app to automate it. 2. Should not be locked into a specific language or framework. 3. Should be open source and 4. Should not reinvent the wheel when it comes to automation APIs

If you are mobile app software test engineer, Appium can makes your mobile app regression testing task easy. Specially for large mobile apps where continues apps are updating with new features and functionalities. Another main benefit of using appium for mobile app automation is it supports bellow given multiple platforms and languages. Also you can use any testing framework.

**Multiple Platforms Support:**

Appium support bellow given different platforms.

1. Android
2. IOS

**Multiple Languages Support:**

Appium supports bellow given languages with the Selenium WebDriver API and language-specific client libraries

1. Java
2. Objective C
3. JavaScript
4. PHP
5. Python
6. Ruby
7. C#
8. Perl

Also there are other advantages like no source code is needed to test app as you can test it directly, also you can engage built in apps like camera, calendar, contacts, dialer etc in your test script if required.

**Limitations of Appium:**

1. For Android, No Support for Android API level below 17
2. Gestures support is limited.
3. No support for Toast messages so need to use third party engines

**Appium iOS Automation:**

**Prerequisites:**

1. Mac machine
2. Device iOS version should be above 12 if you are using xcode version 10.1

**Follow below steps to configure Appium in Mac machine:**

1. **Install Xcode:**

Download and install xcode from app stores and version should be 10.1 for ios verision 12 and above

1. **Install brew package manager:**

/usr/bin/ruby -e “$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)”

Note 1: if you face access write issues while trying to install brew run the following: sudo chown -R $USER:admin /usr/local

Note 2: will ask you to install xcode command line tools if you haven’t got any version of them installed

1. **Install Node.js :**

brew install node

npm install npm@latest -g

npm init ( tap enter to all )

1. **Install Java :**

brew install caskroom/cask/brew-cask

brew cask install java

Note: Java8.0 is mandate to install

1. **The following are needed to be installed for iOS devices.**

brew install carthage

brew install libimobiledevice –HEAD

Note: From MAC Book, Open AppStore -> Search “Xcode” -> Observed “Xcode Developer tools” in app store -> Select Get -> Install App

Xcode installation should be successful before going to further steps (installation takes more than 30 minutes based on internet speed)

npm install -g ios-deploy

brew install ideviceinstaller

1. **Install Appium**

npm install -g appium

npm install wd

npm install appium-doctor –g

**Follow the below steps for Xcode configuration:**

Then run the following line in the terminal to install the command line tools (below steps to install Xcode command line tools which can’t do with AppStore)

xcode-select –install

Check if xcode is installed:

xcode-select –p

**Appium setup for iOS devices:**

To starting using Appium and the Appium inspector for iOS devices you must first install the webdriveragent app to your iOS device.

Open a terminal and go to the folder below.

cd /usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent

The above directory will be generated after installing Appium in the MAC

Now create the following directory:

mkdir -p /usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest- driver/WebDriverAgent/Resources/WebDriverAgent.bundle

Now run the following script

bash Scripts/bootstrap.sh –d

Note: After all the above steps, we can use device to test using Xcode and Appium

**Follow the below steps in MAC for Xcode Setup**

Open the path in Finder: /usr/local/lib/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent

Open WebDriverAgent.xcodeproj in Xcode.

Open Xcode -> Preferences -> Login with developer id -> Manage Certificates -> Download iOS development certificate

Select your development team for both the WebDriverAgentLib and WebDriverAgentRunner targets. This should also auto select Signing Certificate. The outcome should look as shown below.

For signing the WebDriverRunner you need to change the BundleIdentfier to a unique identifier by going to WebDriverAgentRunner>Build Settings>Packaging>Product Bundle Identifier

Build WebDriverAgentLib and WebDriverAgentRunner on your device once to verify all above steps worked by selecting your device

Install integration app finally

Download and install appium from the link: http://appium.io/downloads.html

**For Locating Elements in iOS Appium:**

Launch Appium and give host no: 127.0.0.1 and port no: 4723 and click on start server

Then click on appium inspector which is located at right top

Select Automation server tab> Add below required desired capabilities

* platformName: iOS
* udid: device udid
* automationName: XCUITest
* app: com.wipronec.test1
* deviceName: device udid
* xcodeSigningId: iPhone Developer
* xcodeOrgId: apple account id
* clearSystemFiles: true
* networkConnectionsEnabled: true
* wdaLocalPort: 8100
* usePrebuiltWDA: true
* startIWDP: false
* shouldUseSingletonTestManager: false
* noReset: true

Save it and click on Start session

Then Inspector window will be populated and click on any of the element you will able to be view element automation ids

**Appium Android Automation:**

**Prerequisites:**

1. Download and install Java in window

JDK is java development kit. If you are working with java software programming language then you must need to install it as it is prerequisite to code and run java software programs, use junit/testng framework and Steps are given below for installing java on windows and setting JAVA\_HOME path .

Steps:

1. Goto the link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and download jdk and install.
2. **Set JAVA\_HOME Variable In Windows**

After installation of jdk, you need to set JAVA\_HOME variable in your windows and steps to set JAVA\_HOME are as bellow

1. Locate JDK Installation folder path and copy i.e. Go to C:\Program Files\Java\jdk1.8.0\_45
2. Right click on My Computer icon which is located on your desktop or win start and select properties>Advanced system settings>Environmental variables.
3. Under user variables, Click on New button. It will open New System Variable dialog then set variable name = JAVA\_HOME and set variable value = C:\Program Files\Java\jdk1.8.0\_45

Note: It depends on your own java installation path and installed version.

1. Under system variables select path variable and click on edit
2. At the end of Path variable value string, put semicolon and then put JDK folder's bin folder path e.g. C:\Program Files\Java\jdk1.8.0\_71\bin. Then click on ok
3. To verify java is installed properly or not, open command prompt and run command: java –version. It will show java current version if path is set properly.
4. **Steps To Download And Install Android Studio In Windows**
5. To Download Android Studio, Go to the link : <https://developer.android.com/studio/index.html>
6. Launch Android Studio>Tools>Android>SDK Manager>Then Click on Launch Standalone SDK manager>Tools Options>provide proxy and port
7. In order to create android emulators for testing, you need to download and install few packages. You can do it using Android SDK Manager as described in bellow given steps.

Android SDK Tools package will be installed by default. You can select your required package from list of different packages and then click on Install packages button. After installation of all packages, you need to install Intel Hardware Accelerated Execution Manager to avoid error during android virtual device creation. For to install, select and install Intel x86 Emulator Accelerator (HAXM installer) from Android SDK Manager Dialog.

1. **Set ANDROID\_HOME and Path Environment Variables For SDK In Windows**

After installation of different packages, you need to set ANDROID\_HOME and path environment variables. Follow the below steps to set ANDROID\_HOME path.

1. Open "Environment Variables" dialog from win start menu -> Right click on My Computer -> Select properties -> Advanced system settings -> Environment Variables button from Advanced tab
2. To Set User variable path: Click on New button under User Variable table. It will open New User Variable dialog.
3. Set Variable Name = ANDROID\_HOME and Variable value = C:\SDK (Path of SDK folder). Path can be different for you as per your SDK folder location. And click on OK button.
4. To set system variable path:
5. Open SDK Folder from C: drive.
6. You will find "tools" and "platform-tools" folders inside SDK folder.
7. Copy path of both these folders. e.g. E:\SDK\tools and E:\SDK\platform-tools\
8. Open "Environment Variables" dialog as described above.
9. Locate Path variable line under System Variables table.
10. Edit Path variable by clicking on Edit button. It will open Edit System Variables dialog.
11. Append "tools" and "platform-tools" folder's full path at the end of line as shown in bellow image. e.g. ;C:\SDK\tools;C:\SDK\platform-tools\;
12. To verify Android Is Installed And Configured Properly
13. Open command prompt.
14. Type command android. It will open Android SDK Manager Dialog.
15. That means Android is configured properly in the system.

Now ANDROID\_HOME and Path Environment Variables are set for android SDK in windows environment. So android environment is configured and ready to use with Appium to execute automation tests.

1. **Download And Install Appium In Windows**
2. Go to the link : <http://appium.io/>
3. Download and extract the zip file. Open AppiumForWindows folder. Double click on "appium-installer.exe" and install.
4. Launch the installed appium server.
5. **Connect Android Device With PC In USB Debugging Mode To Run Appium Test**
6. You need to connect real android device with PC in USB debugging mode in order to run android app automation tests in real android device using appium. Follow the below steps to connect
7. Enable Developer option in Android device by clicking 3 times on the build number option.
8. Connect device with PC and enable USB debugging from developer option.
9. To verify device connected properly with pc: Open Command prompt and run command adb devices, it will show the list of devices connected. That means your device is connected properly.

**Locating Android App Elements Using UI Automator Viewer:**

UI Automator Viewer is a tool provided by android SDK. UI Automator Viewer will provide you an user interface to scan and analyse UI components of android software application. Using UI Automator Viewer, We can inspect the android app component hierarchy, Inspect properties of android app components and then we can use those element's properties to create xpath of element and use them in automation test scripts. You will find uiautomatorviewer.bat file in tools folder of SDK. e.g. C:\SDK\tools.

1. **Inspecting App's UI Element Using UI Automator Viewer**

We will use default Camera app of android phone to learn how to inspect its UI elements using UI Automator Viewer tool. Follow the steps given bellow.

1. Connect your android device with PC (USB debugging mode should be enabled).
2. Run command "adb devices" in command prompt to verify device is connected properly with PC.

Run uiautomatorviewer.bat file from C:\SDK\tools folder. It will open UI Automator Viewer tool's UI as bellow.

1. For ex Launch camera app in the device.
2. In UI Automator Viewer software tool, Click on Device Screenshot image button at the left top corner. It will capture the screenshot of device screen
3. After capturing screenshot, It will show your android phone's Camera UI in UI Automatorviewer
4. On left site it will show you Camera app's screenshot which is open in android device.
5. Right side top part will show Camera app's UI element's hierarchy view. It will display node structure to explain how elements are arranged.
6. Right side bottom part will show property detail of selected element.

**Appium - Run iOS Automation Test**

1. Gather required desired capabilities and update in DesriredCapability.properties under Project path/Data/TestData accordingly

* platformName: iOS
* udid: device udid
* automationName: XCUITest
* app: app bundleIdentifier
* deviceName: device udid
* clearSystemFiles: true
* networkConnectionsEnabled: true
* wdaLocalPort: 8100
* usePrebuiltWDA: true
* startIWDP: false
* shouldUseSingletonTestManager: false
* noReset: true
* Url: 127.0.0.1:4723:w

Start Appium Node Server from Terminal:

1. Launch Terminal in mac
2. Run below command

appium –address 127.0.0.1 –port 4723

Build maven package:

1. Launch terminal in mac
2. Go to project folder in terminal
3. Run below command

mvn clean package

1. After building maven build successful, fat jar and Data folder zip files will be generated under Target folder and unzip Data

Start Test from terminal:

1. Open one more terminal
2. Run below command

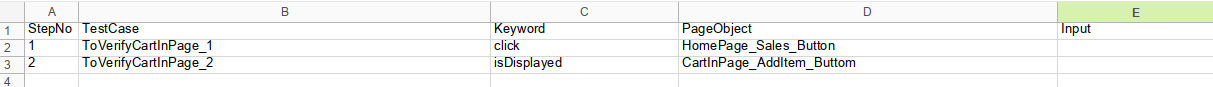
java –jar /path of fat jar file under Target folder/

1. Test cases will starting executing

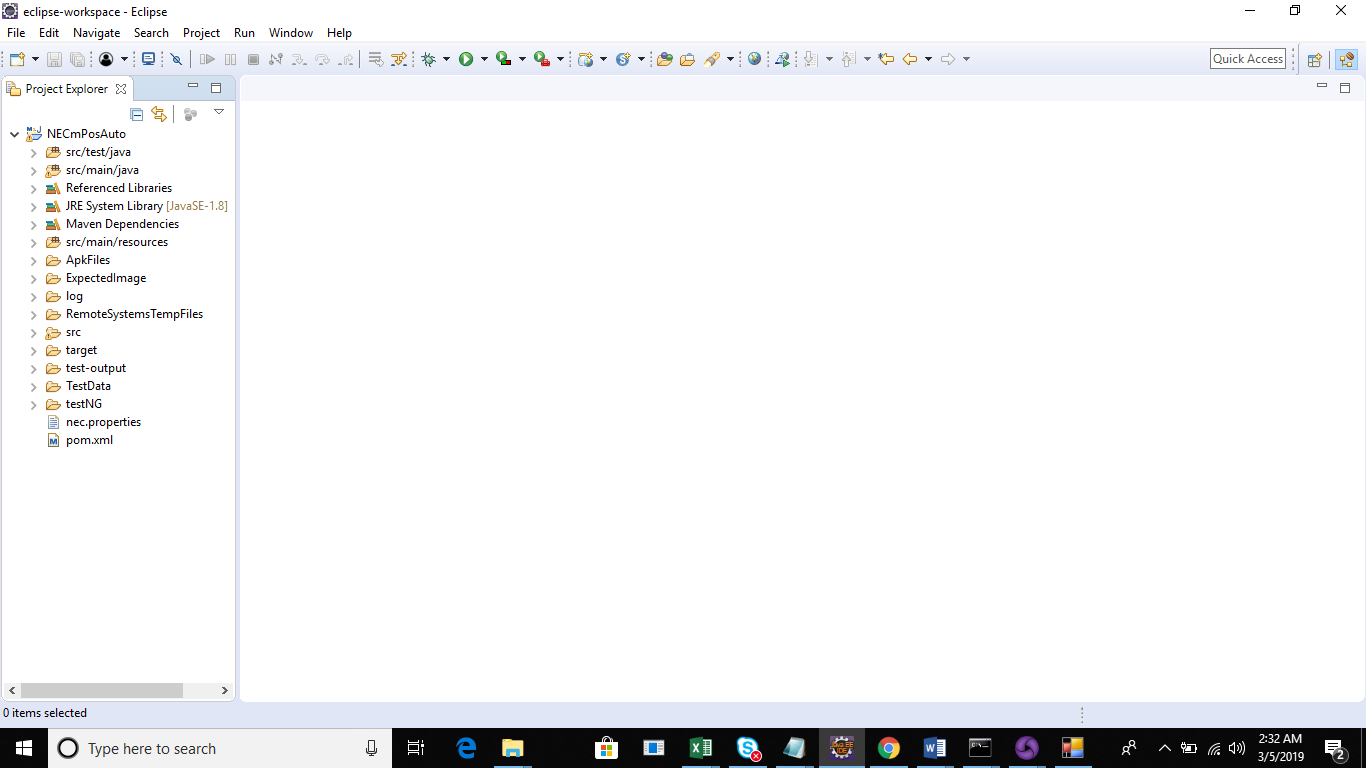
**Project structure for NEC Automation:**

**Adding Element’s locators in object repository and Test steps in TestData excel:**

Preconditions:

1. For example : Test steps should be added in TestData\_CartInPage which is in property file CartInPage.properties under Data/TestData/ as show in below screen
2. C:\Users\DI20064320\Downloads\Screen Shot 2019-03-05 at 7.09.19 PM.png
3. TestData\_CartInpage.xls excel should be in the format shown in below screen
4. 
5. Object repository excel—TestDataPOM.xls excel should be in the format shown in below screen
6. 

The project that has been created is a Maven Project whose project structure is shown as below:



All the required dependencies required for project has been added to pom.xml

The execution starts from Capabilities class which has been created under src/main/java folder as IOS/AndroidCapabilities.java where we define following details:

For Android:

* deviceName - Name of device which is connected with PC.
* platformName - platform name of your android device. ex-Android
* udid - same as device name
* app - path of .apk file

For iOS:

* platformName: iOS
* udid: device udid
* automationName: XCUITest
* app: app bundleIdentifier
* deviceName: device udid
* xclearSystemFiles: true
* networkConnectionsEnabled: true
* wdaLocalPort: 8100
* usePrebuiltWDA: true
* startIWDP: false
* shouldUseSingletonTestManager: false
* noReset: true

And all the desired capabilities like deviceName, platformName, udid, apk path, bundleIdentifier etc details have been fetching from properties file which is created under Data/TestData folder called DesiredCapability.properties.

The above capabilities we add under @BeforeTest method in capabilities class.

Now, actual implementation of code starts with @Test method which has been created in src/main/java as SampleTest.java class

In this class, we define method which is fetching testcase name and properties file details.

Properties file is a simple file which is created under TestData folder with .properties like “SampleTest.properties”.

There are two excel sheets which has been created-one for Object repository and other for Test Steps. And both these excel sheets will be reading from properties file.