Here's the setup Guide for Appium on MacOS

Author: Ratna Madda

Dt: 05/22/2017

<u>Requirements</u>:

- Mac OS X 10.10 or higher.
- XCode Version >= 8
- JAVA Development Kit(JDK)
- Appium Desktop Server
- Apple Developer Tools (command line tools, iPhone simulator)
- HomeBrew – Helps installing Software packages and symlinks on MacOS

  These are the pre-requisites required for setting up Appium on MacOS

Setup Instructions:

1. Download APPIUM Desktop for MacOS from Appium Website following the below provided Link
   https://github.com/appium/appium-desktop/releases/tag/v1.0.2-beta.2

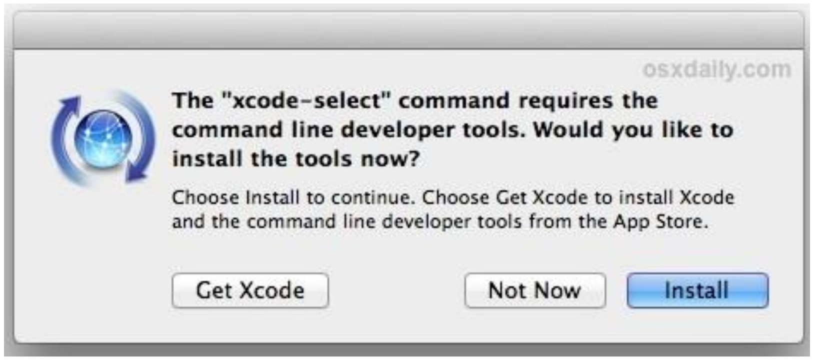
2. Install Homebrew using the Following command that will let you install the Packages for Mac
   ```
   /usr/bin/ruby -e "$(curl –fsSL
   https://raw.githubusercontent.com/Homebrew/install/master/install)"
   ```

3. Install XCode from AppStore using your Apple's developer account.

4. Download the Latest JAVA Development Kit(JDK) from the following Link and Setup Path for JDK in Environment Variables.
   http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html.

5. Install apple CommandLineTools using the following command
   ```
   xcode-select –install
   ```

6. You will get a pop-up which asks you to install command line tools,



The "xcode-select" command requires the command line developer tools. Would you like to install the tools now?

Choose Install to continue. Choose Get Xcode to install Xcode and the command line developer tools from the App Store.

Click Install and you'll see command line tools getting installed.

7. Here are some installations to be done on your MacOS before launching Appium

```
brew install node
brew install npm
brew install idevice installer
npm install wd
brew install ios-webkit-debug-proxy
brew install –HEAD libimobiledevice
npm install –g ios-deploy
```

8. Authorize the need of iOS Simulator by running the following command through npm.

```
npm install –g authorize-ios
```

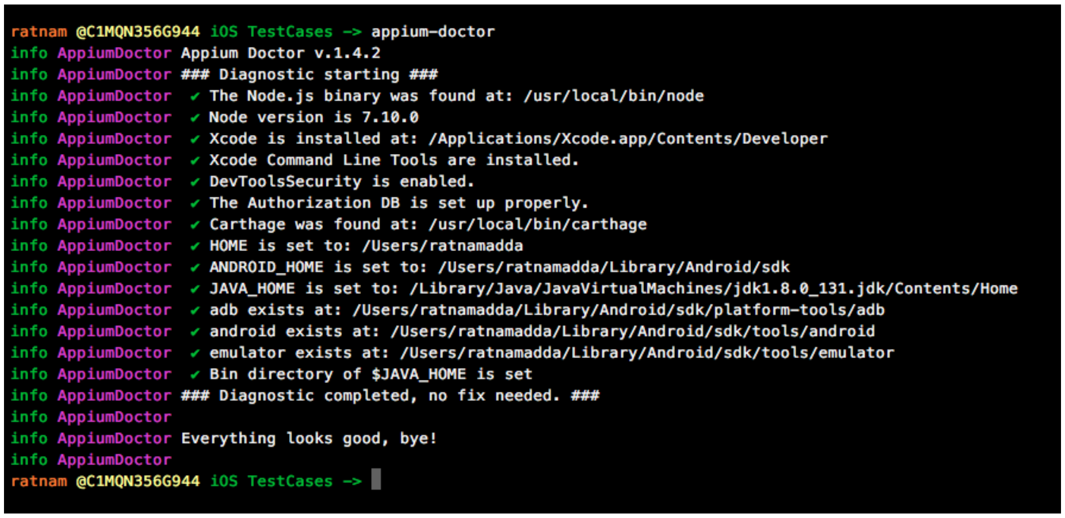
9. In order to automate iOS devices with Xcode 8 (which includes all testing of iOS 10+), you need to install the Carthage dependency manager, using the following command

```
brew install carthage
```

10. Setup environment Variables for Node, NPM, and JAVA, ANDROID SDK (if needed). Use JAVA_HOME, as the JDK Path in system path and environment variables.

11. Verify if all the specified pre-requisites are installed or not using appium-doctor command.

12. Once you see everything is installed as in the below image, you are ready to run the test-scripts.



```
ratnam @C1MQN356G944 iOS TestCases -> appium-doctor
info AppiumDoctor Appium Doctor v.1.4.2
info AppiumDoctor ### Diagnostic starting ###
info AppiumDoctor  ✔ The Node.js binary was found at: /usr/local/bin/node
info AppiumDoctor  ✔ Node version is 7.10.0
info AppiumDoctor  ✔ Xcode is installed at: /Applications/Xcode.app/Contents/Developer
info AppiumDoctor  ✔ Xcode Command Line Tools are installed.
info AppiumDoctor  ✔ DevToolsSecurity is enabled.
info AppiumDoctor  ✔ The Authorization DB is set up properly.
info AppiumDoctor  ✔ Carthage was found at: /usr/local/bin/carthage
info AppiumDoctor  ✔ HOME is set to: /Users/ratnamadda
info AppiumDoctor  ✔ ANDROID_HOME is set to: /Users/ratnamadda/Library/Android/sdk
info AppiumDoctor  ✔ JAVA_HOME is set to: /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home
info AppiumDoctor  ✔ adb exists at: /Users/ratnamadda/Library/Android/sdk/platform-tools/adb
info AppiumDoctor  ✔ android exists at: /Users/ratnamadda/Library/Android/sdk/tools/android
info AppiumDoctor  ✔ emulator exists at: /Users/ratnamadda/Library/Android/sdk/tools/emulator
info AppiumDoctor  ✔ Bin directory of $JAVA_HOME is set
info AppiumDoctor ### Diagnostic completed, no fix needed. ###
info AppiumDoctor
info AppiumDoctor Everything looks good, bye!
info AppiumDoctor
ratnam @C1MQN356G944 iOS TestCases -> 
```

Apple changed the way it makes testing available, and the new way requires the installation of a helper application onto the device, through which the application under test is automated While this is simple in theory, the hoops of code signing and provisioning applications for development and testing can make this a bit of a headache. The application that Appium installs is called WebDriverAgent-Runner

13. Now we have to configure the WebDriverAgent in XCode.
14. The Easiest way to get up and running is to use the Automatic Configuration Strategy.

Use the 'xcodeOrgId' and 'xcodeSigningId' desired capabilities:

```
{
  "xcodeOrgId": "<Team ID>",
  "xcodeSigningId": "iPhone Developer"
 }
```

If automatic configuration doesn't work, then it's time for manual configuration to setup, This usually has to do with code signing and the configuration of the project to be able to be run on the real device under test.

15. Alternatively, the provisioning profile can be manually associated with the project (keep in mind that this will have to be done each time the WebDriverAgent is updated, and is *not* recommended):
16. Find out where your Appium installation is:

```
$ which appium
/path/where/installed/bin/appium
```

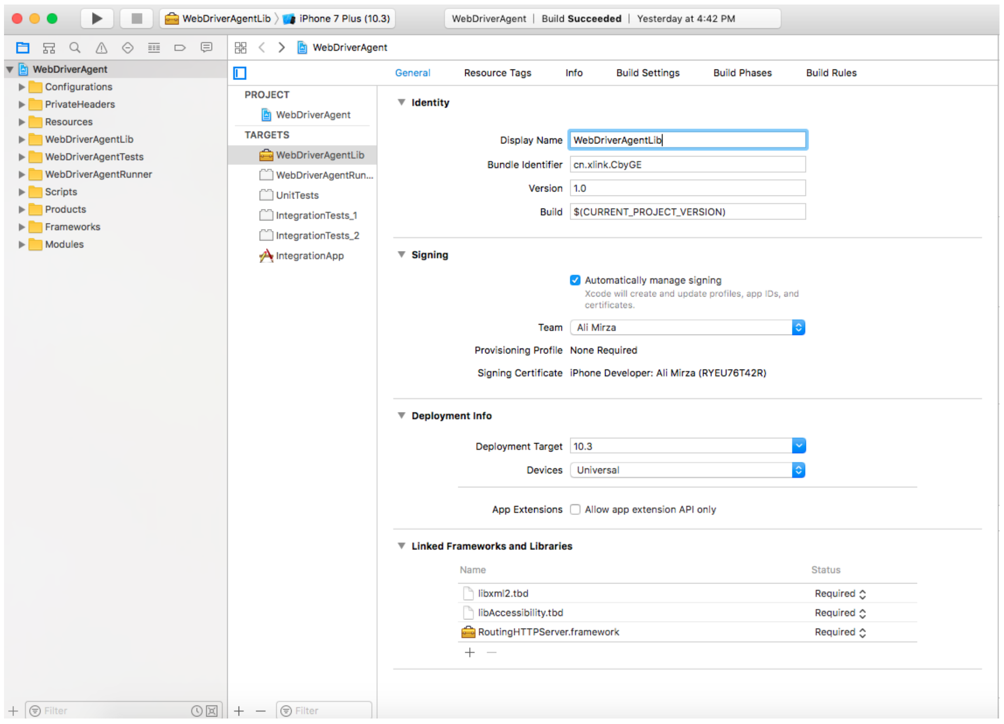
17. Given this installation location, /path/where/installed/bin/appium, WebDriverAgent will be found in /path/where/installed/lib/node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent. Open a terminal and go to that location, then run the following in order to set the project up:

```
mkdir -p Resources/WebDriverAgent.bundle

./Scripts/bootstrap.sh -d
```

18. Go to the WebDriverAgent in /path/where/installed/lib/node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent and ‘ls’ for the list of files in that directory and open the WebDriverAgent.proj using ‘open’ command and it’ll open in Xcode
19. For **both** the WebDriverAgentLib and WebDriverAgentRunner targets, select "Automatically manage signing" in the "General" tab, and then select

your Development Team. This should also auto select Signing Certificate.



Identity

Display Name	WebDriverAgentLib
Bundle Identifier	cn.xlink.CbyGE
Version	1.0
Build	$(CURRENT_PROJECT_VERSION)

Signing

✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team	Ali Mirza
Provisioning Profile	None Required
Signing Certificate	iPhone Developer: Ali Mirza (RYEU76T42R)

Deployment Info

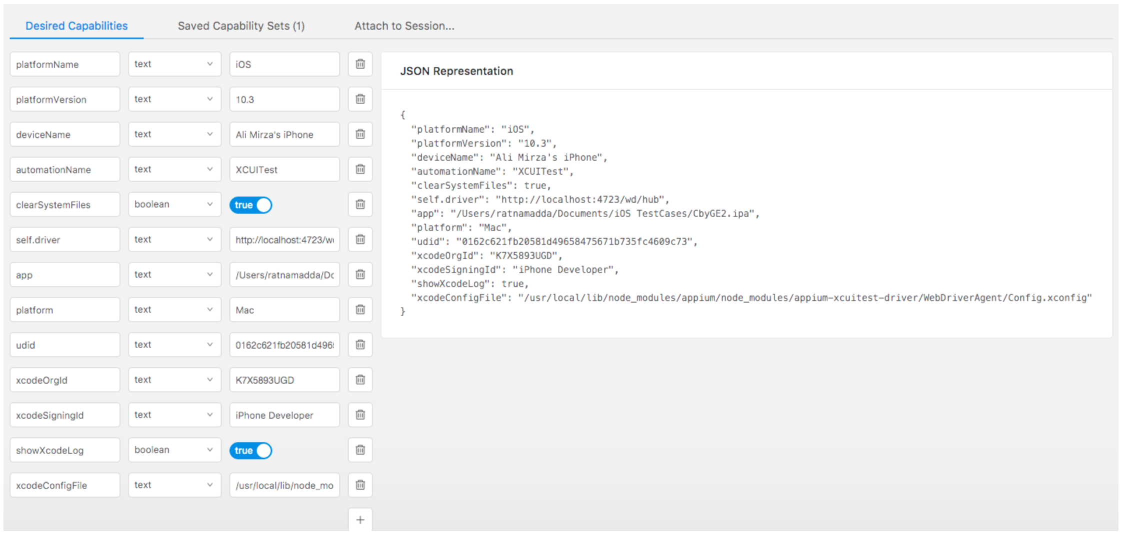
Deployment Target	10.3
Devices	Universal
App Extensions	☐ Allow app extension API only

Linked Frameworks and Libraries

Name	Status
libxml2.tbd	Required
libAccessibility.tbd	Required
RoutingHTTPServer.framework	Required

20. You have to do the same by enabling Automatic Signing and Select Signing Certificate and Provisioning profile for WebDriverAgentRunner too as you did for WebDriverAgentLib.
21. Sometimes Xcode fails to Create a Provisioning Profile, then this demands the necessity of Manually changing bundle id for the target by going into build settings tab.
22. Go to WebDriverAgent Build Settings tab and change product bundle identifier to cn.xlink.CbyGE.
23. Once you change this you can be able to see the Provisioning Profile and Signing certificate will be created.
24. Create a .xcconfig file somewhere on your file system and add the following to it.
```
DEVELOPMENT_TEAM = <Team ID>
CODE_SIGN_IDENTITY = iPhone Developer.
```
25. Once everything is clear, go to product > clean the build and build again, you'll see the build is successful. Now we have successfully configured the WebDriverAgent.
26. Go to Applications > Appium and Open Appium app and Start Server.

27. Once the Server gets started, click start new session, and you'll see appium inspector will be opened.
28. Enable desired capabilities by selecting either automatic (if the port is running on 4723 which is default for Appium) or custom server.
29. Once you fill all the Desired Capabilities you will see the auto-fill in JSON representation to the right.

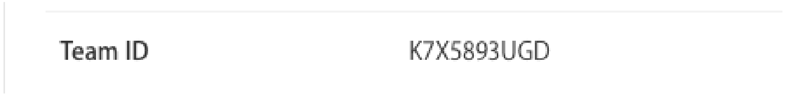


platformName	text	iOS
platformVersion	text	10.3
deviceName	text	Ali Mirza's iPhone
automationName	text	XCUITest
clearSystemFiles	boolean	true
self.driver	text	http://localhost:4723/wı
app	text	/Users/ratnamadda/Dc
platform	text	Mac
udid	text	0162c621fb20581d496!
xcodeOrgId	text	K7X5893UGD
xcodeSigningId	text	iPhone Developer
showXcodeLog	boolean	true
xcodeConfigFile	text	/usr/local/lib/node_mo

JSON Representation

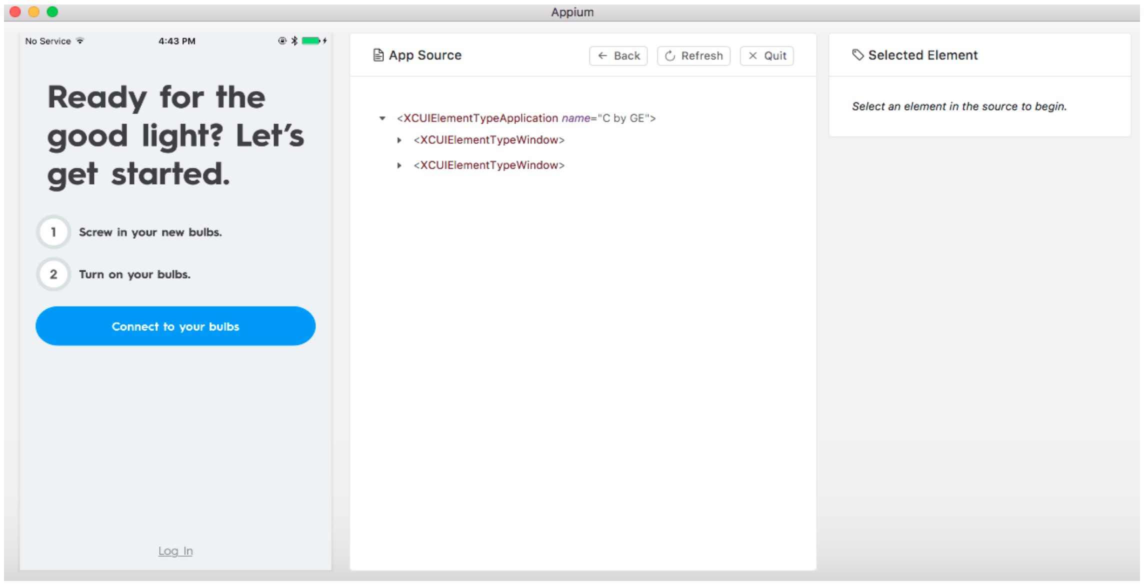
```
{
  "platformName": "iOS",
  "platformVersion": "10.3",
  "deviceName": "Ali Mirza's iPhone",
  "automationName": "XCUITest",
  "clearSystemFiles": true,
  "self.driver": "http://localhost:4723/wd/hub",
  "app": "/Users/ratnamadda/Documents/iOS TestCases/CbyGE2.ipa",
  "platform": "Mac",
  "udid": "0162c621fb20581d49658475671b735fc4609c73",
  "xcodeOrgId": "K7X5893UGD",
  "xcodeSigningId": "iPhone Developer",
  "showXcodeLog": true,
  "xcodeConfigFile": "/usr/local/lib/node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent/Config.xconfig"
}
```

30. Make sure your device capabilities are set correct, the app capability value is the path for your .ipa file and most importantly, the Xcode org id should be the developer Team ID, you'll find in the membership section of your developer account.



Team ID	K7X5893UGD

31. When you are using. ipa file as the build to launch the app, use physical device instead of iPhone Simulator for testing. Apple has made the process of testing difficult where ipa files are not meant for Simulator.
32. If the build is made for physical device, it will not work with simulator.
33. if you still want to use a Simulator, compress the .ipa file and open with Archive Utility and then you'll see the Payload Folder and inside that you'll see the file with .app extension which works with simulator.
34. Once you click Start Session, you will see appium inspector being opened along with the app on your device.



```
<XCUIElementTypeApplication name="C by GE">
  <XCUIElementTypeWindow>
  <XCUIElementTypeWindow>
```

35.Open your terminal and run your test scripts.