SK Institute

**Appium**

*by*

Srikanth Pittala Lead Automation Engineer

**Installation and setup**

1. Install Java 8 and setup environment variable.
2. Install Maven and setup environment variable.
3. Install Android studio for android SDK packages.
4. Setup android environment variable.
5. Creating an emulator via android studio.
6. Enabling Developer mode option in android phone.
7. Install PDANet (universal driver) to connect with Android devices.
8. Install Xcode, HomeBrew, Carthage, appium-xcuitest-driver for (Mac only).
9. Install Appium via npm.
10. Install Appium via installer.
11. Install Appium Doctor.

**Install Java 8 and setup environment variable.**

Download Java 8 from <https://www.oracle.com/java/technologies/javase-jdk8-downloads.html> and install it.





Next step is to set environment variables. We will set JAVA\_HOME and the path variable.

**For Mac:**

For Mac OSX 10.5 or later, Apple recommends to set the $JAVA\_HOME variable to /usr/libexec/java\_home

just export $JAVA\_HOME in file *~/. bash\_profile* or *~/.profile.*

Open terminal and type (You must be familiar with vi editor commands).

*$ vim .bash\_profile*

Add below lines( click on i to insert in file)

*export JAVA\_HOME=$(/usr/libexec/java\_home)*  
*export PATH=${JAVA\_HOME}/bin:$PATH*  
  
**Save the file** ( click on ESC+wq+enter)

Then run below command.

*$ source .bash\_profile*

To confirm if it is set correctly type

*$ echo $JAVA\_HOME*

It should return below path

*/Library/Java/JavaVirtualMachines/1.8.xx.jdk/Contents/Home*

Type “*java -version*” to confirm it.

**For Windows:**

1. From windows search bar type advanced system settings.
2. It will open **System Properties** window.
3. Select advance tab and click environment variable.
4. From system variable, click to new button and enter variable name as JAVA\_HOME and value as path to JDK installed.
5. Next we have to edit the system path variable.
6. click on New button and add this *%JAVA\_HOME%bin. Click OK*
7. From command prompt type *java -version*. The output show java version example:

Text, letter

Description automatically generated

**Install HomeBrew**

Run below command to install homebrew.

*/usr/bin/ruby -e "$(curl -fsSL*[*https://raw.githubusercontent.com/Homebrew/install/master/install*](https://raw.githubusercontent.com/Homebrew/install/master/install)*)"*

**Install Maven and setup environment variable.**

**Method 1: Via homebrew for mac only.**

To install maven

*brew install maven*

**Method 2: Manual install**

1.         Download maven [apache-maven-x.x.x-bin.zip](http://mirrors.estointernet.in/apache/maven/maven-3/3.6.0/binaries/apache-maven-3.6.0-bin.zip) from path <https://maven.apache.org/download.cgi> and extract to your local directory.



**Setting up Maven environment variable.**

**For Mac:**

1. The variable should be updated to bash\_profile or ~/.profile.
2. Open terminal and run below command.
   * 1. *$ vim .bash\_profile*
     2. *export MAVEN\_HOME=/Applications/apache-maven-x.x.x*
     3. *export PATH=$MAVEN\_HOME/bin:$PATH*
     4. *$ source .bash\_profile*
     5. *$ echo $MAVEN\_HOME*
     6. */Applications/apache-maven-x.x.x*
3. After setting environment variable run ***mvn -version***to confirm if maven is set properly.



**For Windows:**

1. From windows search bar type advanced system settings.
2. It will open **System Properties** window.
3. Select advance tab and click environment variable.
4. From system variable, click to new button and enter variable name as **MAVEN\_HOME** and value as path to the Maven folder present.
5. Next, we must edit the system path variable.
6. click on new button and add this *%MAVEN\_HOME%/bin*. Click OK.
7. From command prompt type *mvn -version*. The output show maven

**Install Android studio for android SDK packages.**

**Android Studio Installation and download SDK packages. (Common for both Mac and Windows)**

1.         Download Android studio from <https://developer.android.com/studio> as per your OS and install.

Graphical user interface, text, application, email

Description automatically generated

2.         Launch android studio.

3.         Create a dummy project, after Android studio is launched.

4.         Go to Tools  >> SDK manager. In the latest version of Android Studio, the 'SDK Manager' is an icon on the menu. This opens up the SDK manager dialog box.

Graphical user interface, text, application

Description automatically generated

5.         Select the 'Android SDK' link under 'System Settings' on the left pane, and choose 'SDK Tools' and check the box “Intel x86 Emulator Accelerator” if not checked.

1. Uncheck "Hide Obsolete Packages"  option.
2. Tick mark "Android SDK Platform -Tools (obsolete)" .
3. Click on Apply -> click on OK Button. It will start downloading android sdk to your system.
4. After installation cross check the path where you are able to see all sdk components.
5. Note down the Android SDK location path from sdk manager and copy to a notepad.
6. For ex: /Users/**username**/Library/Android/sdk
7. Click OK to download start, it will take some time to download all the components.
8. Close SDK manager.

Table

Description automatically generated

**Setup android environment variable.**

**For Mac:**

**Just like we set JAVA\_HOME, set the Android environment variable.**

Open terminal and run below commands

*$ vim .bash\_profile*

*export ANDROID\_HOME=/Users/username/Library/Android/sdk*  
*export PATH=$ANDROID\_HOME/platform-tools:$PATH*  
*export PATH=$ANDROID\_HOME/tools:$PATH*

Save the file.

*$ source .bash\_profile*

Open terminal and type *adb version* to verify the environment is set properly.

Text, letter

Description automatically generated

**For Windows:**

Steps for adding environment in windows are same as above for JAVA\_HOME and MAVEN\_HOME.

Add ANDROID\_HOME and value to sdk location under system variable.

Variable Name: **ANDROID\_HOME**

**Variable Value:***C:Usersxxxxx/AppData/Local/Android/Sdk*

**Then add***%ANDROID\_HOME%/platform-tools and %ANDROID\_HOME%/tools***in path variable.**

**For Mac only.**

So if we add all environment variables then the file will look like this. Yes. there are some extra variables added which will come to know as you start automating.

*export JAVA\_HOME=$(/usr/libexec/java\_home)*  
*export PATH=${JAVA\_HOME}/bin:$PATH*  
*export ANDROID\_HOME=/Users/$(whoami)/Library/Android/sdk*  
*export ANDROID\_SDK\_ROOT=$ANDROID\_HOME*  
*export ANDROID\_AVD\_HOME=/Users/$(whoami)/.android/avd*  
*export PATH=$ANDROID\_HOME/tools:$ANDROID\_HOME/platform-tools:$ANDROID\_SDK\_ROOT/emulator:$PATH*  
*export M2\_HOME=/Users/$(whoami)/Downloads/Softwares/Maven*  
*export PATH=$PATH:$M2\_HOME/bin*

**Creating an emulator via android studio**

1. Open Android studio.
2. Go to Tools >> Android >> AVD Manager.
3. From Virtual configuration window select >> Create a new device.
4. Select the device you want to download. **Ex. Pixel**
5. Select OS version as per your requirement. **Ex. Oreo Android 8.0 with Google APIs.**
6. Accept the licence agreement.
7. It will start downloading the image. (Takes time depending on network speed and size).
8. After download start the emulator from AVD manager device list.

**Enabling Developer mode option in android phone/Emulator.**

1. Go to settings and scroll down to **About phone** option.
2. Scroll down to Build number and tap multiple times. You can see the toast message “You are now x steps away to being a developer”.
3. After tapping 5-7 times when you see “No you are a developer” , which confirms that you have enabled developer mode.
4. To confirm go back to the previous page and select System >> Advanced >> Developer option.
5. From Developer option scroll down and enable **USB debugging**, **Verify apps over USB** option.

**Install PDANet (universal driver) to connect with Android devices.**

To interact with devices and system we need a USB driver, usb driver is different with respective to android devices. But we can install one driver which will work across all the android devices.

1. Go to <http://pdanet.cov/install/>.
2. Download the installer as per your operating system and install it.Graphical user interface, text, application

   Description automatically generated

**Install Xcode (For Mac only)**

Open App store and search for xcode then install it.

Graphical user interface, text, application, email

Description automatically generated

**Install HomeBrew(For Mac Only)**

1. Open terminal.And type

*/bin/bash -c "$(curl -fsSL*<https://raw.githubusercontent.com/Homebrew/install/master/install.sh>)"

1. This will install **homebrew** in your mac machine.
2. For more information go through <https://brew.sh/>
3. There is also a dependency, made necessary by Facebook's [WebDriverAgent](https://github.com/facebook/WebDriverAgent), for the [Carthage](https://github.com/Carthage/Carthage) dependency manager. If you do not have Carthage on your system, it can also be installed with [Homebrew](http://brew.sh/)

*brew install carthage*

1. To install webdriver-agent

npm i appium-webdriveragent

-- After installing open below project with XCode

/Applications/Appium.app/Contents/Resources/app/node\_modules/appium/node\_modules/appium-xcuitest-driver/WebDriverAgent/WebDriverAgent.xcodeproj

--Build the project on simulator device.

1. For real devices we can use [xcpretty](https://github.com/supermarin/xcpretty) to make XCode output more reasonable. This can be installed by gem install xcpretty
2. More information at:

<https://www.npmjs.com/package/appium-xcuitest-driver>

1. For appium-xcuitest-driver installation refer below link.

<https://github.com/appium/appium-xcuitest-driver/blob/HEAD/docs/real-device-config.md>

**Appium Setup:**

There are two ways to install such as via npm and installer.

**Installing via npm** which requires node.js installation.

**Install Node.js**

1.         Download node.js as per your operating system from <https://nodejs.org/en/download/>.





2.         After successful installation we have to set the environment variable for Windows OS but for mac it is not needed.

Open command prompt or terminal and type “*npm-v*” it returns the version number which confirms that node.js is configured properly.

**Appium Installation:**

1.         Open command prompt or terminal in **administrator mode** and run below command.

2.         Run *npm install -g appium*  to install globally, if you want to install a particular version then run ***npm install -g appium@1.16.0***  This will trigger the installation and it will take 5-10 mins depending on your network and RAM size.

3.         After successful installation run **appium** command in terminal/command prompt.



1. If this returns below result, then Appium is successfully installed to your system.

**Install via Appium installer.**

You can download appium installer from appium official website.

Go to <http://appium.io/downloads.html> and click to download.

Text

Description automatically generated

It will take you to [https://github.com/appium/appium-desktop/releases/tag/vx.](https://github.com/appium/appium-desktop/releases/tag/v1.15.1)xx.x

Select the installer as per your operating system and download.

[Appium-windows-x.xx.x.exe](https://github.com/appium/appium-desktop/releases/download/v1.15.1/Appium-windows-1.15.1.exe) file is for windows and [Appium-mac-x.xx.x.dmg](https://github.com/appium/appium-desktop/releases/download/v1.15.1/Appium-mac-1.15.1.dmg) file for mac.





Install it. Then run it will open one window , hit the start button  appium server console.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

**Install appium doctor**

This tool is used to diagnose and fix common Node, iOS and Android configuration issues before starting Appium.

To install run below command in terminal

*npm install appium-doctor -g*



To execute, run  ***appium-doctor*** command.

This will give you a report as below, if any configuration is missing please fix them and proceed.

Text, letter

Description automatically generated

**Getting familiar with ADB**

**What is ADB?**

* Adb stands for Android Debug Bridge. It is a client server program which allows us to communicate with an Android device or an Emulator using command line.
* ***A client****, which runs on our machine. We can invoke a client by issuing an adb command.*
* ***A daemon****, which runs as a background process on each emulator or device instance. It by default runs on port number 5037*
* ***A server****, which runs as a background process on our machine. It manages communication between the client and the adb daemon running on an emulator or device.*Where is it located?
* After successful download of android sdk, the adb tools comes with SDK inside the platform-tools folder. You can get the path using “which adb” command.

**Pre-requisites:**

Before running adb commands, ensure that USB debugging is enabled, and the device is connected to the system using USB.

**List of ADB commands and usage:**

1. **adb devices :** To find out list of connected devices.
2. ***Offline :****The device is not connected or not responding*
3. ***device :****The device is connected to adb server*
4. ***no device :****There is no device connected to adb server*
5. **adb usb →**To switch to USB mode.
6. **adb tcpip [port no] →** To start TCP IP Connection.
7. **adb connect [device IP address] →** To connect devices over wifi.
8. **adb version →**To check ADB version.
9. **adb logcat →** To display logs of connected devices.
10. **adb logcat > path of the filelog.txt →** To store adb logs in text file.
11. **adb kill-server →**To kill adb server.
12. **adb start-server →** To start adb server.
13. **adb shell →** To run the terminal of host Android device. Below command can be used with adb shell command.
14. ***getprop ro.product.model →***This command get’s the attached device model no
15. ***getprop ro.build.version.release →***This command get’s the android build release version i.e. 4.4.2/4.4.4
16. ***getprop ro.build.version.sdk →***This command get’s the device api version i.e. 19,20,21,22
17. ***pm list packages –f →****To get app package of installed apps.*

**How to install apk using abd command?**

1. Get the apk file from dev or any other source.
2. Paste in your system location.
3. Connect your device and your device should have USB debugging enabled.
4. Open command prompt.
5. Type below command (enter your apk file name without angular bracket).  
     
   **Note:**Make sure when you execute you have .apk extension mentioned.  
    *adb install <YourApkName>.apk*  
   If multiple devices connected then how to install into a particular device.

**Solution:**

1. Follow 1-4 steps from solution 1.
2. Type <adb devices> in command prompt.
3. This will list out the connected device.
4. Note the device id and enter below command  
   *adb -s <YourDeviceId> install <YourApkName>.apk*  
     
   If Apk Installation failed due app is already exists,  
   Enter command prompt and enter below command.  
     
   *adb install -r <YourApkName>.apk*  
     
   

**adb uninstall [package name or apk’s path] →**To uninstall apk of package.  
  
**adb install -r [apk] →** To command to update the app.  
  
**Adb command for file sharing:**  
*adb pull <source folder> <destination folder>*  
  
**Command to send files from computer to phone:**  
*adb push <source folder> <destination folder>*

**Understanding Desired Capabilities:** *DesiredCapabilities is a JSON object in the form of a key and value pair sent from client to server. It has basic set of information to create automation sessions. Let's look out some essential desired capabilities.*  
cap.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "0797818a87d4f5cb");  
It shows the kind of mobile device or emulator used for automation. Such as “Samsung S10” , “Google Pixel”, “Android Emulator”, “iOS Simulator”.  
  
*cap.setCapability(MobileCapabilityType.****VERSION****, "9.0");*  
The device operating system such as 7.0, 9.0, 10, 11 etc  
  
*cap.setCapability(MobileCapabilityType.****PLATFORM\_NAME****, "Android");*  
cap.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, "iOS");  
The device OS platform such as iOS or Android.  
  
*cap.setCapability(MobileCapabilityType.****AUTOMATION\_NAME****, "UiAutomator2");*  
The automation engine, for android mention UIAutomator2 and for iOS it is XCUITest  
  
*cap.setCapability(MobileCapabilityType.****UDID****, "E5EB4A6D-32AC-4B2A-9AA8-3BC25D8FC0BF");*  
Unique device identifier of the connected physical device. We will later learn how to get this for android and iOS.  
  
*cap.setCapability(MobileCapabilityType.****APP****, “//Path to build file”));*  
The absolute or local path for build. For android it is .apk and for iOS .ipa and .app  
  
*cap.setCapability("noReset","true");*  
Don't reset the app before this session. It could be true or false.  
  
*cap.setCapability("fullReset","false");*  
It performs a complete reset, it could be true or false.  
  
*cap.setCapability(AndroidMobileCapabilityType.****APP\_PACKAGE****, "io.appium.android.apis");*  
*cap.setCapability(AndroidMobileCapabilityType.****APP\_ACTIVITY****, ".ApiDemos");*  
  
The above capabilities are mainly used to invoke installed android app.

App package is the java package of an app on which you want to run.

App activity is the android activity you want to launch from your package.

Generally, apps start from .splashActivity. We will see how we can get it later.

In the case of iOS if you want to invoke installed app then use budleID with app desired capabilities.  
  
**Example:**

cap.setCapability(MobileCapabilityType.***APP***,"com.example.apple-samplecode.UICatalog");  
We will discuss how to get this later.  
  
  
**How to get the app package and activity of an app?**

1. Connect your device and make sure it’s detected by adb
2. Then open command prompt type **adb shell**
3. Now type

dumpsys window windows | grep -E 'mcurrentFocus|mFocusedApp'

1. **"Do not press enter now".**Launch the app in your device and at the same time execute the above command it will list out the package name and the current activity.

Sample output:  
  
C:UsersSiddharth>adb shell  
dreamlte:/ *$ dumpsys window windows | grep -E*'mcurrentFocus|mFocusedApp'mFocusedApp=AppWindowToken{6ec47ae token=Token{e3aa829 ActivityRecord{59aa1b0d0 u0 **io.appium.android.apis**/**.ApiDemos** t2017}}}  
dreamlte:/ $  
  
**From above output :**  
"**io.appium.android.apis**" is your app package.  
"**.ApiDemos**" in your app activity.  
  
  
App package and app activity to be used in desired capabilities as  
  
*DesiredCapabilities cap = new DesiredCapabilities(); cap.setCapability(AndroidMobileCapabilityType.APP\_PACKAGE, "io.appium.android.apis"); cap.setCapability(AndroidMobileCapabilityType.APP\_ACTIVITY, ".ApiDemos");*  
  
**Official link:** <http://appium.io/docs/en/writing-running-appium/caps/>How to get bundleId (for iOS only)  
  
Appium provides bundleID to identify the iOS application. Generally in organizations we can get it the same from developers. If the developer is not unknown and you are working on public application, then refer below steps.   
  
  
**A. Using Command prompt:**

* 1. Get the file (.ipa or .app) file to your local. Here we will take example of UICatalog app.
  2. Run below command

osascript -e 'id of app "/path to the app/UIKitCatalog.app"'

* 1. It will will give you the output as

com.example.apple-samplecode.UICatalog  


**B. From the apple store app store.**

* 1. Take the Udemy app.
  2. App store link for Udemy app is : <https://apps.apple.com/us/app/udemy-online-video-courses/id562413829>
  3. Note down the id of the app **562413829**
  4. Create the link to get the bundle id as <https://itunes.apple.com/lookup?id=562413829>
  5. Open the above link in a browser that will download a text file. Open the file and search for bundleID bundle id is com.udemy

**C. From Console.**

* 1. Launch the IOS simulator from XCode or connect your iPhone to mac.
  2. From spotlight search, search for Console and launch.
  3. Select your service from the left side list then search for the app as shown below.Graphical user interface, text, application

     Description automatically generated

**How to get the UDID of a device?**  
A UUID is a string of letters and digits that forms a unique pattern. Your Mac, iPhone, and iPad each have one UUID, and no other device shares it. It’s like a serial number in that regard, but whereas serial numbers are used to identify your device by Apple and your cellular carrier, developers usually use the UUID instead.

**For Android**:

* + 1. Make sure the device has USB debugging enabled and connected via USB.
    2. From terminal enter command adb devices. This will give you the UDID.

**For iOS devices using XCode:**

* + 1. Launch simulator from XCode or connect an iPhone to mac.
    2. Click to menu windows, select devices and simulators.
    3. Click to the tab Devices or Simulators as per your device/Simulator.
    4. Select the device and on the right side you will get the UDID as an identifier.
    5. Refer below screenshot.

Graphical user interface, text, application

Description automatically generated

* **CLU tool for iOS as like ADB for Android:**  
  CLU is nothing but the Command Line Utility tool. To save the space for the menu link name, I’ve used the short-term CLU. You know very well ADB is nothing, but the Android Debug Bridge used to connect and control the Android emulators or devices in the command prompt. I would like to share some of the important XCRUN commands in this post, which will be useful for you to control the iOS simulator.  
  XCRUN is going to play a major role for iOS platform as like ADB in Android platform. XCRUN is a dedicated one for iOS and ADB is a dedicated one for Android, which cannot be used vice-versa.
* **What is XCRUN?**  
  XCRUN is a binary file available in the XCODE tool in which simulators installed in the device can be controlled. If you compare with the Android platform, it is almost equivalent to ADB (Android Debug Bridge). ADB can be used to control the Android Emulators and the real devices running in the Android platform. XCODE is an IDE with the iOS SDK’s which is used to develop the iOS applications in Mac machine. And the same XCODE should be used for iOS Automation to build the code, install the application, invoke the application etc.
* **What is SIMCTL?**  
  simctl will work with the XCRUN binaries in which it is nothing but Command line utility to control the Simulator, which is again developed by using the Ruby. All the simulator-controlled commands will work based on the combination of XCRUN and SIMCTL.

**SIMCTL – Simulator Control:**  
**Run the following the commands in the terminal window and check the expected results for your reference:**  
  
**How to list the device types, runtimes and devices;**  
*$ xcrun simctl list*  
  
**How to list the devices;**  
*$ xcrun simctl list devices*  
  
**How to list the available device types;**  
*$ xcrun simctl list device types*  
  
**How to list the runtimes available in the Mac;**  
*$ xcrun simctl list runtimes*  
  
**How to list the Known Devices;**  
*$ instruments -s devices*  
  
**How to get the real device UDID; Tried with iPhone 7 real device**  
*$ idevice\_id -l*

**How to install the .APP file in the simulator under the command line:**  
*$ xcrun simctl install<space><device id><space><path of the .app build file>*

Example:  
*$ xcrun simctl install 37CEC824-C20E-4D77-8A69-59DD778928AF*/Users/gopikannan/Library/Developer/Xcode/DerivedData/UICatalog-alaibrcewphnhpbkacmggbnkdyim/Build/Products/Debug-iphonesimulator/UICatalog.app

**How to Uninstall the .APP file in the simulator under the command prompt:**  
*$ xcrun simctl uninstall<space><device id><space><application identifier>*

Example:  
*$ xcrun simctl uninstall 37CEC824-C20E-4D77-8A69-59DD778928AF com.example.apple-samplecode.UICatalog*

**How to take screenshot:**  
*$xcrun simctl io<space><device id><space>screenshot<space><filename.png>*  
*$ xcrun simctl io 37CEC824-C20E-4D77-8A69-59DD778928AF screenshot testscreen.png*

**How to record the screen; Ctrl+C to stop in the command prompt:**  
*$xcrun simctl io<space><device id><space>recordVideo<space><filename.mov>*  
*$ xcrun simctl io 37CEC824-C20E-4D77-8A69-59DD778928AF recordVideo testfile.mov*

**How to create a simulator in command prompt:**  
*$ xcrun simctl create “iPhone 6” “iPhone 6” 9.2*

**How to launch the simulator in command prompt:**  
*$xcrun simctl launch<space><device id><space><application identifier>*  
*$ xcrun simctl launch 37CEC824-C20E-4D77-8A69-59DD766628AF com.example.apple-samplecode.UICatalog*

**How to get help information about xcrun simctl? You could get all the command information:**  
*$ xcrun simctl help*  
  
Text

Description automatically generated with medium confidence