**APPIUM**

**What is Appium?**

* It is an automation tool used to automate & test mobile applications (Native app, mobile web app, Built-in app & hybrid app) on both Android and IOS platform.
* It written in Node.js
* It is developed by Sauce Labs company.
* It is open-source mobile automation tool.

**Appium architecture.**



* When user (selenium client) sent a request(consists Desired Capabilities i.e., package name, activity name, platform name & version , application name ) through JSON protocol to Appium server It will create a session id and sent it back as response to client.
* Based on this session id client(Selenium) and server(Appium) will communicate each other.
* Appium server creates service request and send it to UI Automation(IOS) or UI Automator(Android).
* These UI Automator or Automation will communicate with Bootstrap.jar which is running in emulator , simulator & real device and convert request into operations.

**Advantages :**

* It supports all languages
* Platform independent (android, IOS & Windows)
* Supports all application i.e., Built-in apps, native apps, web apps & hybrid apps.
* Open source

**Disadvantages :**

* Server should be up and running before executing script.
* Slow execution on virtual devices
* Supports only android API levels which are greater than 17.
* Parallel execution will not support on single device.
* Can’t handle Toast messages

**Emulator & Simulator:**

* These are the virtual devices which have same functionality of real devices except camera.
* A virtual device(android or ios) on windows os is called emulator and virtual device(android or ios) on IOS os is called simulator.

**Different types of mobile applications :**

1. **Built-in application or System applications.**

Applications which are installed as part of operation system are called built-in applications.

We can’t un-install these applications from mobile, but we can disable or enable it.

Example : Camera, Calculator, Alarm, Calendar etc.

1. **Native application:**

Applications which are developed on base of particular platform (Android & IOS) are called native applications.

Example : WhatsApp, Facebook, Gmail etc.

1. **Mobile web application:**

These are web applications or websites which are opened with the help of any browser in mobile.

Example : Chrome Browser, Firefox Browser etc.

1. **Hybrid application:**

These are combination of Native and Web applications.

Example : Amazon, Flipkart, Facebook, Instagram, Daily hunt etc.

**Different types of mobile Operating Systems:**

1. Android OS by Google
2. IOS OS by Macintosh
3. Windows by Microsoft

**List of Android versions.**

**Installations & Setup:**

1. Android Studio ( Required java pre-installed)
2. Appium Desktop application
3. Jar files :

* Selenium standalone jar files
* Appium client server jar files
* Common lang3

1. Environment variable

System Variables

* JAVA\_HOME
* ANDROID\_HOME (C:\Users\admin\AppData\Local\Android\Sdk)

Edit path & add

* % JAVA\_HOME%/bin
* % ANDROID\_HOME%/platform-tools
* % ANDROID\_HOME%/tools
* % ANDROID\_HOME%/tools/bin
* % ANDROID\_HOME%/build-tools
* C:\Program Files\Android\Android Studio\bin

**What is APK ?**

* APK stands for Android Package Kit and it's the file format that Android uses distribute and install apps.
* APK is an archive file contain all the elements that an app needs to install correctly on android device.

**What is IPA ?**

* IPA stands for iOS App Store Package and it's the file format that IOS uses distribute and install apps.
* IPA is an archive file contain all the elements that an app needs to install correctly on IOS device.

**What is ADB :**

ADB stands for Android debugging bridge.it is a command line tool which is used to communicate with virtual and real devices connected through USB or WI-FI.

**Advantages :**

1. We can get all devices information of virtual and Real devices (connected with USB or WIFI).

**Command line :** adb devices

1. We can connect real device with WIFI

**Command line :** adb tcpip 5555

adb connect <mobile ip>:5555

1. We can get all packages information

**Command line :** adb shell pm list packages

1. We can get specific apk from device

**Command line :** adb shell pm list packages

adb shell pm path <package-name>

adb pull <package-location> <path-on-computer-to-store-APK>

1. We can install APK on device

**Command line :** adb install <apk path>

1. We can kill adb server

**Command line** : adb kill-server

1. We can restart adb server

**Command line** : adb start-server

**What is AVD Manager?**

AVD stands for Android virtual device.

AVD Manager is a platform present in android studio that helps to create and manage android virtual devices.

**What is SDK Manager?**

The SDK manager is a command line tool that allows you to view, install, update, and uninstall packages for the Android SDK

**What Is Appium Inspector ?**

It is a tool which is used to find element on both android and Ios devices.

It present in appium server.

**What is uiautomator & uiautomation?**

**Uiautomator:** Appium uses Uiautomator to execute commands on real devices(Android) and emulators. Uiautomator is Google's test framework for mobile app automation at the UI level.

**Uiautomation:** Appium uses Uiautomation to execute commands on real devices(IOS) and simulators. Uiautomation is ios test framework for mobile app automation at the UI level.

**Install APK in Virtual Device & Real Device using Appium.**

Step1: Start emulator or connect real device with USB or Wifi

Step2. Launch Appium and start server

Step3: Click on start inspector icon

Step4: Under desired Capabilities tab add respective capabilities like platformName, devicename, app, platformversion etc.

Step5: Click on start session.

**Install APK in Virtual Device & Real Device using adb.exe**

Step1: Start emulator or connect real device with USB or Wi-Fi.

Step2. Launch Appium and start server

Step3: open command prompt and execute below line

adb install <apk path>

**Install APK in Virtual Device & Real Device by script.**

**Script:**

DesiredCapabilities ds = **new** DesiredCapabilities();

ds.setCapability(MobileCapabilityType.***AUTOMATION\_NAME***, "Appium");

ds.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, "Android");

ds.setCapability(MobileCapabilityType.***PLATFORM\_VERSION***, "7.1");

ds.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "Android Emulator"); //or

// ds.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "Android"); // Real Device

// ds.setCapability(MobileCapabilityType.APP, "D:\\Selenium\_SS\\ ApiDemos-debug.apk");

or

ds.setCapability(MobileCapabilityType.APP, "https://github.com/appium/appium/raw/master/sample-code/apps/ApiDemos-debug.apk");

ds.setCapability("appPackage", "com.finance.emi.calculate");

ds.setCapability("appActivity", "com.finance.emi.calculate.ui.SplashActivity");

URL url = **new** URL("http://0.0.0.0:4723/wd/hub");

AndroidDriver<WebElement> driver = **new** AndroidDriver<WebElement>(url,ds);

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

Thread.*sleep*(10000);

driver.findElementById("com.finance.emi.calculate:id/btn\_skip").click();

driver.findElementById("com.finance.emi.calculate:id/rootView").click();

driver.findElementByXPath("//android.widget.EditText[@resource-id='com.finance.emi.calculate:id/principal']").sendKeys("1000000");

driver.findElementById("com.finance.emi.calculate:id/interest").sendKeys("10.25");

driver.findElementById("com.finance.emi.calculate:id/loan\_tenure").sendKeys("5");

driver.findElementByXPath("//android.widget.Button[@text='Calculate']").click();

Thread.*sleep*(10000);

driver.quit();

**Different types of locators in Appium?**

Ref: <https://appium.io/docs/en/commands/element/find-elements/index.html#selector-strategies>

|  |  |
| --- | --- |
| **Locator** | **attributes selection** |
| ID | android : resource-id & IOS : name |
| name | Name |
| Class name | Class name |
| Accessibility ID | android : content-desc & IOS : accessibility-id |
| xpath | xpath |
| Android UiAutomator | [Use the UI Automator API, in particular the UiSelector & UiScrollable class to locate elements. In Appium you send the Java code, as a string, to the server, which executes it in the application’s environment, returning the element or elements.](https://developer.android.com/reference/android/support/test/uiautomator/UiSelector.html) |
| Android View Tag | Locate an element by its view tag |
| IOS UiAutomation | [Use the UI Automator API, in particular the UiSelector & UiScrollable class to locate elements. In Appium you send the Java code, as a string, to the server, which executes it in the application’s environment, returning the element or elements.](https://developer.android.com/reference/android/support/test/uiautomator/UiSelector.html) |
| Image | Locate an element by matching it with a base 64 encoded image file |

1. **ID**

Each element has a unique ID which makes easier to identify.

**resource-id** is used as an element identifier for Android and **name** is used for iOS.

**Example:**

driver.findElementById("IntegerA"); // for **iOS**

driver.findElementById("android:id/text1")).click(); //for **Android**

1. **Accessibility ID**

It is a highly preferred locator strategy, especially in the case of automating Android and iOS test cases. Developers can explicitly set the Accessibility ID during development.

For iOS, the default **Accessibility ID** is set to the **name** of the UI element. For Android, the value of Accessibility is same as the value of the attribute “**content-desc**”.

**Example:**

driver.findElementByAccessibilityId("Accessibility").click();

1. **ClassName**

Finding an element using class name is very common, but multiple elements may have the same class name, and this creates a problem in finding one particular element.

So, we needs to use a combination of multiple attributes, for example, combining text with the class name to identify the element.

For iOS, Class Name is represented as the full name of the XCUI element and begins with XCUIElementType.

**For example** – UIAButton, UIARadioButton

In the case of Android, the Class Name is called out as the full name of the UIAutomator2 class. **For example** – android.widget.TextView

**Example** :

List<WebElement> buttons = driver.findElementsByClassName("android.widget.TextView"); for(WebElement button : buttons)

{

System.out.println(button.getText());

if(button.getText().equals("Animation"))

{

button.click();

}

}

1. **Xpath**

We should go xpath only when there is no ID, Name, or accessibility ID assigned to a specific UI element.



**Example:**

driver.findElementByXPath("(//XCUIElementTypeButton)[1]");

driver.findElementByXPath("//android.widget.Button[@text='Calculate']").click();

driver.findElementByXPath("//android.widget.EditText[@resource-id='com.finance.emi.calculate:id/principal']").click();

1. **Android UI Automator**

This locator is Android-specific. we needs to use the UI Automator API, i.e. **UISelector** Class to search for specific elements.

Ref : https://developer.android.com/training/testing/ui-automator#java

**Example:**

String selector = "new UiSelector().text(“Cancel”)).className(“android.widget.Button”))"; MobileElement element = (MobileElement)driver.findElement(MobileBy.AndroidUIAutomator(selector));

**UiSelector class** : Represents a query for one or more target UI elements on a device.

**methods:**

1. text()

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().text(\"Review Tabs\")").click();

1. textContains(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().textContains(\"Tabs\")").click();

1. textStartsWith(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().textStartsWith(\"Review\")").click();

1. textMatches(“.\*app”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().textMatches(\".\*view\")").click();

1. description(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().description(\"android.widget.Btn\"))").click();

1. descriptionContains(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().descriptionContains(\"widget.Btn\"))").click();

1. descriptionStartsWith(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().descriptionStartsWith(\"android.widget \"))").click();

1. descriptionMatches(“.\*app”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().descriptionMatches(\".\*widget.btn\"))").click();

1. resourceId(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().resourceId(\"com.finance.emi.calculate:id/rootView\").index(1)").click();

1. resourceIdMatches(“.\*app”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().resourceIdMatches(\".\*calculate:id/rootView\").index(1)").click();

1. className(“xxxx”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().className(\"android.widget.EditText\").resourceId(\"com.finance.emi.calculate:id/principal\")").sendKeys("10");

1. classNameMatches(“.\*app”)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().classNameMatches(\".\*EditText\").resourceId(\"com.finance.emi.calculate:id/principal\")").sendKeys("10");

1. instance(index)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().className(\"android.widget.EditText\").instance(1)").click();

1. index(x)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().className(\"android.widget.EditText\").index(1)").click();

1. clickable(true)

**Example:**

driver.findElementByAndroidUIAutomator("new UiSelector().resourceId(\"com.finance.emi.calculate:id/extraRepaymentCard\").clickable(true)").click();

1. scrollable(true)) :

**Example:**

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollIntoView(new UiSelector().className(\"android.widget.Button\").index(2))").click();

**scrolling in Appium :**

**UiScrolable** class : Provides support for searching for items in a scrollable UI container.

**methods:**

1. flingForward(performs quick swipe)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).flingForward()"));

} **catch**(InvalidSelectorException e) {

}

1. flingBackward(performs quick swipe)

Example :

**try**

{

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).flingBackward()"));

}

**catch**(InvalidSelectorException e) {

}

1. scrollForward(moves exactly one view)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollForward()"));

} **catch**(InvalidSelectorException e) {

}

1. scrollBackward(moves exactly one view)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollBackward()"));

}**catch** (InvalidSelectorException e) {

}

1. scrollToBeginning(10) moves exactly by one view. 10 scrolls max)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollToBeginning(10)"));

} **catch** (InvalidSelectorException e) {

}

1. flingToBeginning(10) performs quick swipes. 10 swipes max)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).flingToBeginning(10)"));

} **catch** (InvalidSelectorException e) {

}

1. scrollToEnd(10) moves exactly by one view. 10 scrolls max)

**Example** :

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).scrollToEnd(10)"));

} **catch** (InvalidSelectorException e) {

}

1. flingToEnd(10) performs quick swipes. 10 swipes max)

**Example :**

**try** {

driver.findElement(MobileBy.AndroidUIAutomator("new UiScrollable(new UiSelector().scrollable(true)).flingToEnd(10)"));

} **catch** (InvalidSelectorException e) {

}

1. scrollIntoView(new UiSelector().className(“xxxx”))

**Example**:

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector()).scrollIntoView(new UiSelector().description(\"android.widget.Buttn\"))").click();

or

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().resourceId(\"android:id/list\")).scrollIntoView(new UiSelector().text(\"Share Result\").instance(0)").click();

1. getChildByText(new UiSelector().className(“xxxx”),”text”)

**Example**:

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().resourceId(\"android:id/list\")).getChildByText(new UiSelector().className(\"android.widget.Button\"),\"Share Result\")").click();

1. getChildByDescription(new UiSelector().className(“xxxx”),”description”)

**Example**:

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().resourceId(\"android:id/list\")).getChildByDescription(new UiSelector().className(\"android.widget.Button\"),\"Share Result\")").click();

1. setAsHorizontalList()

**Example**:

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().resourceId(\"android:id/list\")).setAsHorizontalList().scrollIntoView(new UiSelector().text(\"Share Result\").instance(0)").click();

or

driver.findElementByAndroidUIAutomator("new UiScrollable(new UiSelector().resourceId(\"android:id/list\")).setAsHorizontalList().getChildByText(new UiSelector().className(\"android.widget.Button\"),\"Share Result\")").click();

1. **Android View Tag**

This is also an Android platform-specific locator. We use this to locate elements using its view tag.

**Example:**

driver.findElementByAndroidViewTag("android.widget.Button").click();

**AndroidDriver class methods:**

1. launchApp(): This method used to launch application.

**Example**:

driver.launchApp();

1. installApp(String appPath) : This method used to Install the given app onto the device.

**Example**:

driver.installApp("D:\\xxxx\\xxx.apk");

1. isAppInstalled(java.lang.String bundleId); This method used to Check whether the specified app is installed on the device. it returns Boolean value.

**Example:**

**boolean** status = driver.isAppInstalled("io.appium.android.apis");

System.***out***.println(status);

1. resetApp() : This method used Reset the currently running app for this session

**Example:**

driver.resetApp();

1. runAppInBackground(Duration); This method Send the currently running app for this session to the background

**Example**:

driver.runAppInBackground(Duration.*ofSeconds*(10));

1. removeApp(java.lang.String bundleId); This method used Remove an app from the device.

**Example:**

driver.removeApp("com.example.AppName");

1. closeApp(): Close an app on device

**Example**:

driver.closeApp();

1. activateApp(String bundleId); Activate the given app onto the device

**Example:**

driver.activateApp('com.apple.Preferences');

driver.activateApp('io.appium.android.apis');

1. terminateApp(String bundleId) : Terminate the given app on the device.

**Example:**

driver.terminateApp(“com.apple.Preferences”);

driver.terminateApp(“io.appium.android.apis”);

1. getPlatformName() : Get the name of the current platform.

**Example**:

String **package** = driver.getCurrentPackage();

1. getCurrentPackage() : Get the name of the current Android package

**Example**:

String **package** = driver.getCurrentPackage();

1. currentActivity() : Get the name of the current Android activity

**Example**:

String activity = driver.currentActivity();

1. getDeviceTime() : Get the time on the device

**Example**:

String time = driver.getDeviceTime();

1. getScreenshotAs() : Take a screenshot of the current viewport/window/page.

**Example**:

File scrFile = driver.getScreenshotAs(OutputType.FILE);

1. rotate(ScreenOrientation):Rotate the device in 3 dimentions.

**Example**:

driver.rotate(org.openqa.selenium.ScreenOrientation.***PORTRAIT***);

1. isBrowser() : Check whether opened app is browser or not

**Example**:

boolean browser = driver.isBrower();

1. isDeviceLocked(): Check whether the device is locked or not

**Example**:

**boolean** isLocked = driver.isDeviceLocked();

1. isKeyboardShown() : Check whether keyboard as displayed or not

**Example**:

**boolean** keyboard = driver.isKeyboardShown();

1. lockDevice(): Lock the device

**Example**:

driver.lockDevice();

1. lockDevice(Duration) : Lock the device for given time.

**Example**:

driver.lockDevice(Duration.ofSeconds(10));

1. unlockDevice() :Unlock the device.

**Example**:

driver.unlockDevice();

1. openNotifications() : open the notifications

**Example**:

driver.openNotifications();

1. getKeyboard() : display keyboard.

**Example**:

driver.getKeyboard();

1. hideKeyboard() : Hide soft keyboard

**Example**:

driver.hideKeyboard();

1. isKeyboardShown() : Whether or not the soft keyboard is shown

**Example**:

boolean isKeyboardShown = driver.isKeyboardShown();

1. longPressKeyCode(AndroidKeyCode) : Press and hold a particular key code on the device

Example:

driver.longPressKeyCode(AndroidKeyCode.HOME);

1. pressKeyCode(AndroidKeyCode) : Press a particular key code on the device

Example:

driver.pressKeyCode(AndroidKeyCode.HOME, AndroidKeyMetastate.META\_SHIFT\_ON);

1. getContext() : Get the current context (application type) in which Appium is running

**Example**:

String context = driver.getContext();

1. setClipboardText(string text) : Set the content of the system clipboard

**Example**:

driver.setClipboardText("happy testing");

1. getClipboardText(string text) : Get the content of the system clipboard

Example:

String clipboardtext = driver.getClipboardText();

1. setPowerAC(PowerACState.***ON***) : For Android emulator. To set the state of the battery charger to connected or not.

**Example**:

driver.setPowerAC(PowerACState.***ON***);

1. setPowerCapacity(int) : For Android emulator. To set the battery percentage.

**Example**:

driver.setPowerCapacity(100);

1. pullFile(path) : Retrieve a file from the device's file system

**Example:**

**byte**[] fileBase64 = driver.pullFile("/path/to/device/xyz.xlx");

1. shake(): Perform a shake action on the device

Example:

driver.shake();

1. pushFile(path) : Retrieve a file from the device's file system

**Example:**

**byte**[] fileBase64 = driver.pushFile("/path/to/device/xyz.xlx", **new** File("/Users/johndoe/files/foo.bar"));

1. makeGsmCall() : Make call (Emulator only)

Example:

driver.makeGsmCall("555-123-4567", GsmCallActions.CALL);

1. sendSMS() : Send sms message (Emulator only)

Example:

driver.sendSMS("555-123-4567",”Hellow”);

1. toggleAirplaneMode() :Toggle airplane mode on device

Example:

driver.toggleAirplaneMode();

1. toggleData() : Switch the state of data service

Example:

driver.toggleData();

1. toggleLocationServices() : Switch the state of the location service

Example:

driver.toggleLocationServices();

1. toggleWifi() : Switch the state of the wifi service

Example:driver.toggleWifi();

1. getPerformanceData() : Returns the information of the system state which is supported to read as like cpu, memory, network traffic, and battery.

Example:

List<List<Object>> performanceData = driver.getPerformanceData("my.app.package","cpuinfo", 5);

1. quit() : End the running session

Example:

driver.quit();

1. getSessionDetails() : Retrieve the capabilities of the specified session

Example:

Map<String, Object> caps = driver.getSessionDetails();

1. location() :Get the current geo location

Example :

Location location = driver.location();

TouchAction Class in appium:

Touchaction class used to perform interactive action on mobile application.

1. lognPress(PointOption) : Perform long press on element.

**Example**:  
TouchActions action = **new** TouchActions(driver);

action.longPress(element).perform();

1. doubleTap() : Perform double tap on element.

**Example**:

TouchActions action = **new** TouchActions(driver);

action.doubleTap(element).perform();

1. singleTap() : Perform single tap on element.

**Example**:

TouchActions action = **new** TouchActions(driver);

action.singleTap(element).perform();

1. waitOption(Duration) : Perform wait between the actions.

**Example**:

TouchActions action = **new** TouchActions(driver);

action.tap(element).waitOption(Duration.ofSeconds(10)).

singleTap(element).perform();

1. press(PointOption) : Preform press action on element.

Example:

TouchActions action = **new** TouchActions(driver);

action.press(element).perform();

1. release() : Preform release action on element.

Example:

TouchActions action = **new** TouchActions(driver);

action.release(element).perform();

1. cancel()
2. moveTo(PointOption)

alerts in appium

Textbox, checkbox, radio button, taggle button & Toast message

swipe in appium

toast message

Date picker, time, and date change.

**How to stop installing the App again and again?**

**a. Setting appium --no-reset Flag to true :** This Flag is used when we don't want to reset app state between sessions (Android: don’t uninstall app before new session). Its default Value is **False**,

We can start your Appium server with "appium --no-reset" command or include server capability like:   
 ds.setCapability("noReset", "true");

or

ds.setCapability(MobileCapabilityType.***NO\_RESET***, **true**);

**b. Make Sure --full-reset Flag is False:** (Android) Reset app state by uninstalling app instead of clearing app data. On Android, this will also remove the app after the session is complete. Default **false**.  
 ds.setCapability("fullReset", "false");

or

ds.setCapability(MobileCapabilityType.***FULL\_RESET***, **false**);

**Run appium server through command line:**

To run appium server from command prompt we need to install below software’s

* Node.js :

Step 1: Download Node.js Installer from browser, navigate to https://nodejs.org/en/download/

Step 2: Install Node.js.

Step 3: Verify Installation by below command in cmd,

**npm -v**

**Note :** npm stands for Node Package Manager (software)

* Appium :

Step 1: Install appium by executing below command in cmd,

**npm install -g appium**

Step 2: Verify Installation by running appium with below command in cmd,

**appium -v**

Run appium execute the below command,

**appium &**

**Run appium server through script:**

* **Run appium with default service**

**Requrired jar files :**

1. **slf4j.jar**
2. **logback.jar**
3. **commons-validator.jar**

AppiumDriverLocalService appiumService = AppiumDriverLocalService.*buildDefaultService*();

appiumService.start();

DesiredCapabilities ds = **new** DesiredCapabilities();

ds.setCapability(MobileCapabilityType.***AUTOMATION\_NAME***, "Appium");

AndroidDriver<WebElement> driver = **new** AndroidDriver<WebElement>(appiumService.getUrl(),ds);

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.quit();

appiumService.stop();

* **Run appium with build service**

AppiumServiceBuilder asb = **new** AppiumServiceBuilder();

asb.withIPAddress("0.0.0.0");

asb.usingAnyFreePort();

// asb.usingPort(4723);

AppiumDriverLocalService appiumService = AppiumDriverLocalService.*buildService*(asb);

appiumService.start();

DesiredCapabilities ds = **new** DesiredCapabilities();

ds.setCapability(MobileCapabilityType.***AUTOMATION\_NAME***, "Appium");

AndroidDriver<WebElement> driver = **new** AndroidDriver<WebElement>(appiumService.getUrl(),ds);

driver.manage().timeouts().implicitlyWait(20, TimeUnit.***SECONDS***);

driver.quit();

appiumService.stop();

**UIAutomator Viewer :**

Graphical user interface, text, application

Description automatically generated

* **Setting up UIAutomatorViewer**

**Prerequisites for setting up UIAutomatorViewer**

1. Install Appium Install Android Studio and SDK Tools
2. Install Java and set up the environment variables
3. Install Eclipse IDE for Java Connect the Android device where the test application is to be tested.
4. The system must have Appium Setup and should be up & running

* **Opening UIAutomator Viewer through SDK tools kit**

This can be done by either of the following methods:

1. entering uiautomatorviewer in the command prompt
2. opening uiautomatorviewer.bat file in the Android installation folder with the following navigation: Android >> Android-SDK >> Tools >> UIAutomatorViewer.bat

* **Understanding different locator strategies**

1. ID
2. Class Name.
3. Xpath
4. Accessibility ID
5. Android UI Automator
6. Android View Tag (Espresso Only)
7. iOS UI Automation

* **Taking the Device screenshot**

This option used to take device screenshot with complete hierarchy

* **Device screenshot with Compressed hierarchy**

This option used to take device screenshot with compressed hierarchy

* **NAF toggle**

NAF stands for Not Accessibility Friendly

We're looking for UI controls that are enabled, clickable but have no **text** nor **content-description**. Such controls configuration indicates an **interactive control** is present in the UI and is most likely not **accessibility** friendly. We refer to such controls here as NAF controls.