<https://www.linkedin.com/pulse/appium-top-50-real-time-interview-questions-evergreen-akhil-reddy>

System requirements:

Android – API version 17 and above, does not support android version lower than 4.4

iOS – OSX 10.7 and XCode 4.6.3 iOS vesion 6.1 and later

Firefox only in simulator

public class ScreenOrientation {

static AndroidDriver driver;

**// Set path of your node.exe file. Set your path.**

**// Progra~1 represents Program Files folder.**

String nodePath = "C:/Progra~1/Appium/node.exe";

**// Set path of your appium.js file. Set your path.**

String appiumJSPath = "C:/Progra~1/Appium/node\_modules/appium/bin/appium.js";

**// This method Is responsible for starting appium server.**

public void appiumStart() throws IOException, InterruptedException {

**// Created object of apache CommandLine class.**

**// It will start command prompt In background.**

CommandLine command = new CommandLine("cmd");

**// Add different arguments In command line which requires to start appium server.**

command.addArgument("/c");

command.addArgument(nodePath);

command.addArgument(appiumJSPath);

**// Set Server address**

command.addArgument("--address");

command.addArgument("127.0.0.1");

**// Set Port**

command.addArgument("--port");

command.addArgument("4723");

command.addArgument("--no-reset");

command.addArgument("--log");

**// Set path to store appium server log file.**

command.addArgument("D://appiumLogs.txt");

**// Execute command line arguments to start appium server.**

DefaultExecuteResultHandler resultHandler = new DefaultExecuteResultHandler();

DefaultExecutor executor = new DefaultExecutor();

executor.setExitValue(1);

executor.execute(command, resultHandler);

**// Wait for 15 minutes so that appium server can start properly before going for test execution.**

**// Increase this time If face any error.**

Thread.sleep(15000);

}

**// This method Is responsible for stopping appium server.**

public static void appiumStop() throws IOException {

**// Add different arguments In command line which requires to stop appium server.**

CommandLine command = new CommandLine("cmd");

command.addArgument("/c");

command.addArgument("taskkill");

command.addArgument("/F");

command.addArgument("/IM");

command.addArgument("node.exe");

**// Execute command line arguments to stop appium server.**

DefaultExecuteResultHandler resultHandler = new DefaultExecuteResultHandler();

DefaultExecutor executor = new DefaultExecutor();

executor.setExitValue(1);

executor.execute(command, resultHandler);

}

@BeforeTest

public void setUp() throws Exception {

**// Start appium server.**

appiumStart();

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability("deviceName", "ZX1B32FFXF");

capabilities.setCapability("browserName", "Android");

capabilities.setCapability("platformVersion", "4.4.2");

capabilities.setCapability("platformName", "Android");

capabilities.setCapability("appPackage", "io.appium.android.apis");

capabilities.setCapability("appActivity", "io.appium.android.apis.ApiDemos");

driver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

}

@Test

public void performOrientation() throws InterruptedException {

**//Get and print current screen orientation.**

System.out.println("\*--\*--\*-- Current screen orientation Is : " + driver.getOrientation());

System.out.println("\*--\*--\*-- Changing screen Orientation to LANDSCAPE.");

**//Changing screen Orientation to LANDSCAPE.**

driver.rotate(org.openqa.selenium.ScreenOrientation.LANDSCAPE);

**//Get and print screen orientation after changing It.**

System.out.println("\*--\*--\*-- Now screen orientation Is : "+ driver.getOrientation());

Thread.sleep(5000);

**// Scroll till element which contains "Views" text If It Is not visible on screen.**

driver.scrollTo("Views");

**// Click on Views.**

driver.findElement(By.name("Views")).click();

System.out.println("\*--\*--\*-- Changing screen Orientation to PORTRAIT.");

**//Changing screen Orientation to PORTRAIT.**

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

**//Get and print screen orientation after changing It.**

System.out.println("\*--\*--\*-- Now screen orientation Is : "+ driver.getOrientation());

Thread.sleep(5000);

}

@AfterTest

public void End() throws IOException {

driver.quit();

**// Stop appium server when test Is ended.**

appiumStop();

}

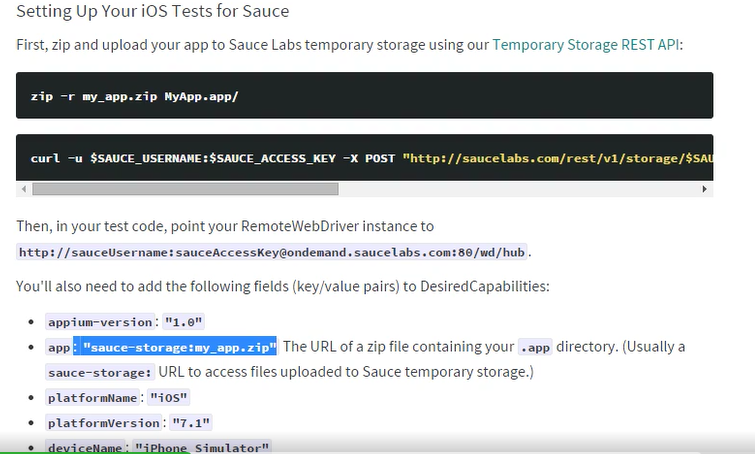
}

## Appium start from cmd prompt

**node appium.js -a 127.0.0.1 -p 1234 -cp 1234 -bp 2345**

* ***Node Appium.js****: To start the instance with node with Appium.js located in the location too provide Appium server arguments*
* *For* ***address*** *we have to pass “****-a****” with the address here address is* ***“127.0.0.1”:  “ -a 127.0.0.1*** *“*
* *For* ***port*** *we have to pass “****-p”*** *with port number here port number is “****1234”****:  “****-p******1234*** *“*
* *For* ***callbackPort*** *we have to pass “****-cp”*** *with port number here port number is “****1234”****:  “****-cp******1234*** *“*
* *For* ***BootstrapPort*** *we have to pass “****-bp”*** *with port number here port number is “****2345”****:  “****-bp******2345*** *“*





## Basic code

File appDir = **new** File("src/apk");

File app = **new** File (appDir,"bookMyShow.apk");

DesiredCapabilities cap = **new** DesiredCapabilities();

cap.setCapability(MobileCapabilityType.***PLATFORM\_NAME***, MobilePlatform.***ANDROID***);

cap.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "Android Emulator");

cap.setCapability(MobileCapabilityType.***NEW\_COMMAND\_TIMEOUT***, "60");

cap.setCapability(MobileCapabilityType.***APP***, app.getAbsolutePath());

AndroidDriver driver = **new** AndroidDriver(**new** URL("http://127.0.0.1:4723/wd/hub"),cap);



Real device

* **Go to Settings -> About Phone -> Build Number -> Tap 7 times**
* **Settings -> USB Debugging** is enabled
* **ADB** lists your devices into the connected devices - adb devices
* Changing the **Desired capability** as per the hardware

<https://github.com/appium/appium/blob/master/docs/en/appium-setup/real-devices.md>

Screen shot

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability("device", "**android**");

 // Take the Screen shot and store it in a file

        org.openqa.selenium.WebDriver augmentedDriver = new Augmenter().augment(webDriver);

**File screenshot = ((TakesScreenshot) augmentedDriver).getScreenshotAs(OutputType.FILE);**

        // the name of the file

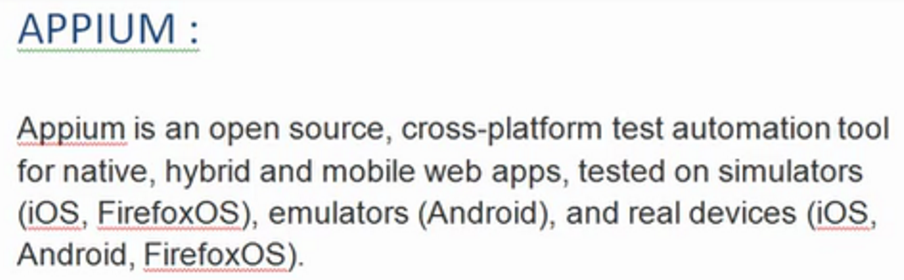
        String filename = "screenshot.png";

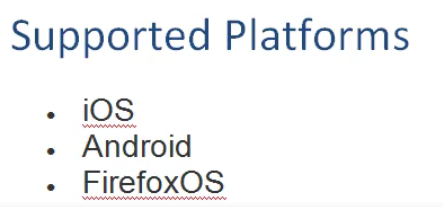
        FileUtils.copyFile(screenshot, new File(filename));

IOS

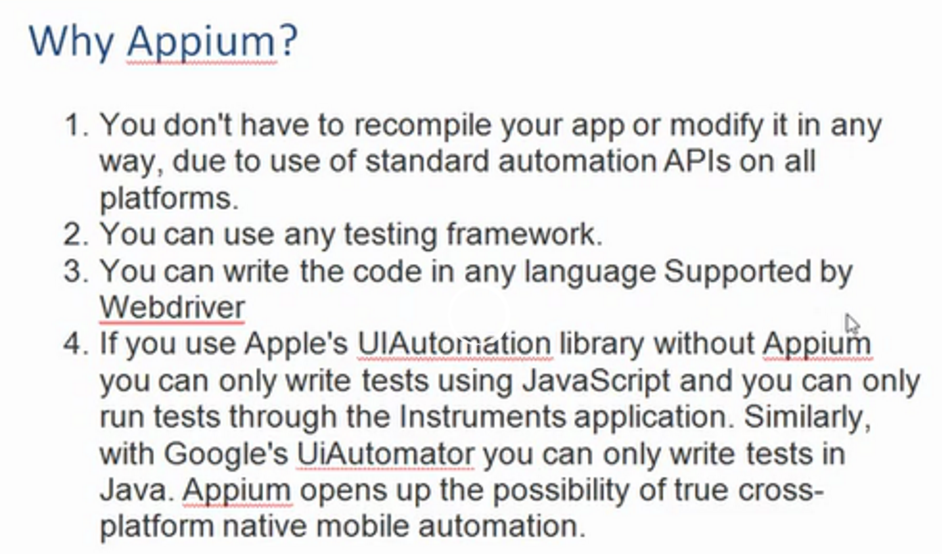
use java Runtime class to execute system commands. You can use it like :

Runtime.getRuntime().exec("idevicescreenshot -u "+udid+" temp.tiff");





## Why Appium?



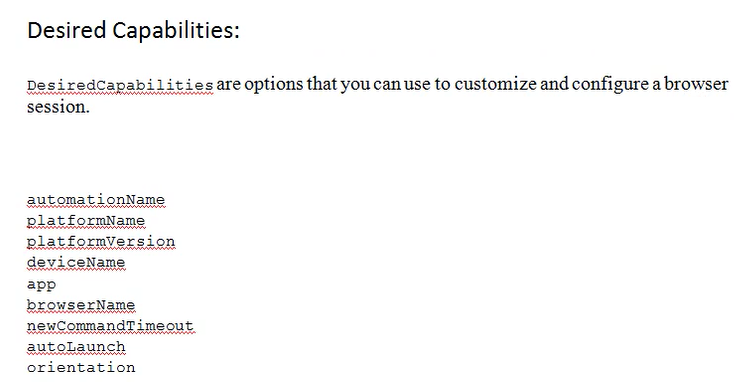
Frameworks bundled in Appium

Android - Android UI Automator

iOS - IOS instrumentation

JSON wire protocol - capabilities in the form desired cap obj and send as session obj

## Desired Capabilities:



### automationName

Which automation engine to use – Appium (default) or Selendroid

### PlatformName

Which mobile OS platform to use - iOS, Android, or FirefoxOS

### platformVersion

Mobile OS version- e.g., 7.1, 4.4

### deviceName

The kind of mobile device or emulator to use

- iPhone Simulator, iPad Simulator, iPhone Retina 4-inch, Android Emulator, Android device, Galaxy S4, etc.... On iOS, this should be one of the valid devices returned by instruments with instruments -s devices. On Android this capability is currently ignored, though it remains required.

### App

The absolute local path or remote http URL to an.ipa or .apk file, or a .zip containing one of these. Appium will attempt to install this app binary on the appropriate device first. Note that this capability is not required for Android if you specifyappPackage and appActivity capabilities (see below). Incompatible with browserName.

/abs/path/to/my.apk or <http://myapp.com/app.ipa>

### browserName

Name of mobile web browser to automate. Should be an empty string if automating an app instead.

'Safari' for iOS and 'Chrome', 'Chromium', or 'Browser' for Android

### newCommandTimeout

How long (in seconds) Appium will wait for a new command from the client before assuming the client quit and ending the session

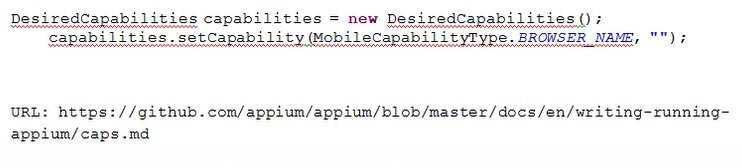
e.g. 60

### orientation

(Sim/Emu-only) start in a certain orientation

### autoLaunch

automationName, PlatformName, platformVersion, deviceName and App are must for native mobile apps.



Desired Capabilities takes inputs, talks to appium server to create session for us.

## Drivers

* Android driver - AndriodDriver driver = new AndroidDriver(new URL(“http://127.0.0.1:4723/wd/hub”),cap);

4723 is the port number of appium server.

* iOS driver

## Steps in invoking the app:

* It will bring the device to initial state i.e., it will close the app of it is open
* If the app is already installed in the device, it will not install the app. Otherwise app will be installed.
* Kills UI Automator if it is open.
* Checks if the screen is unlocked. If it is locked it will unlock.
* It will execute the steps.

## Locating Elements

### Multiple elements with same class

List<WebElement> eles = driver.findElements(By.className(“android.widget.EditText”));

Eles.get(<index>) returns single element

### uiautomatorviewer.bat

* Present in tools folder of sdk
* Take device screenshot
* Various properties like class, id, name will be available

## Context

/\* Code for context to find the View used in application \*/  
Set contextNames = Appiumsetup.driver.getContextHandles();  
for (String contextName : contextNames) {  
System.out.println(contextName);  
if (contextName.contains("WEBVIEW")){  
Appiumsetup.driver.context(contextName);  
}  
}

## Methods:

* Driver.scrollTo(<text>).click(); - this is specific to Appium
* Driver.isLocked();
* Driver.getContext() – When you open an app, it returns if it is opened in a mobile or web view. Eg: NATIVE\_APP
* driver.isAppInstalled(appPath));
* Driver.installApp(appPath);
* Driver.closeApp();
* Driver.currentActivity();

In mobile app each page is defined as an activity. Each page has some functionality called activity.

* Driver.context(name);
* Driver.zoom(element);
* Webelement.getText() – returns the text of the element

### AndroidKeyCodes

* Driver.sendKeyEvent(AndroidKeyCode.<Code>);

Code can be HOME, BACK, BACKSPACE, ENTER

### Touch Action Methods:

TouchAction t = new TouchAction(driver);

.perform() is required after every function of TouchAction class.

* T.tap(WebElement el).perform();
* T.tap(int x, int y).perform();
* T.tap().perform();
* t.longPress(element);

### Mobile Element Methods

To handle actions specific to mobile devices like swipe, tap, zoom we have to cast the element returned by findElement method to MobileElement.

* MobileElement.swipe(SwipeElementDirection direction, int duration\_in\_ms)

If we want to see the elements at bottom, we give SwipeElementDirection as UP. Though swipe and scrollTo do the same job, the difference is that swipe function is treated as touch action and works as if someone is touching the screen and swiping it

Abc.swipe(SwipeElementDirection.UP,2000);

* MobileElement.tap(fingers,duration);
* MobileElement.zoom();

## UiSelector for Android

<https://developer.android.com/reference/android/support/test/uiautomator/UiSelector.html>

It enables to identify the elements of an Android App with various properties of the element like description, text. We don’t have direct methods in WebDriver to handle the elements using these properties though we can use XPath and CSS. We don’t have By.text in WebDriver like we have By.id

Driver.findElementByAndroidUIAutomator(“new UiSelector().text(\”<text>\”)”).Click();

To find a list of elements

List<WebElement> eles= Driver.findElementByAndroidUIAutomator(“new UiSelector().Clickable(true));

Driver.findElementByAccessibilityId()

## To know the devices connected

Open Command Prompt

Go to platform tools folder in sdk folder using cd command

Use the command: adb devices

Use the flwng cmds when the device is not shown in the connected devices list. These cmds will restart the server

Adb kill –server

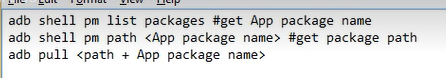
Adb start -server

## Pull App from a real device and install in emulator

Adb devices command can be used to get the list of devices connected.

TO connect a real mobile device, unlock it and connect it to the computer using USB

Following cmds to get the app from real device



Then remove the real device and install the app in emulator:

Adb install <path of app>

## Invoking an App without apk file

We can do it by using package and activity properties. To get these properties:

* Install apk info app and open.
* Click on View Apps
* Select the interested App. It shows the package name and activities list.

To Open an app using these details the app has to be already in the device/ emulator

DesiredCapabilities cap = new DesiredCapabilities();

cap.setCapability(MobileCapabilityType.PLATFORM\_NAME, MobilePlatform.ANDROID);

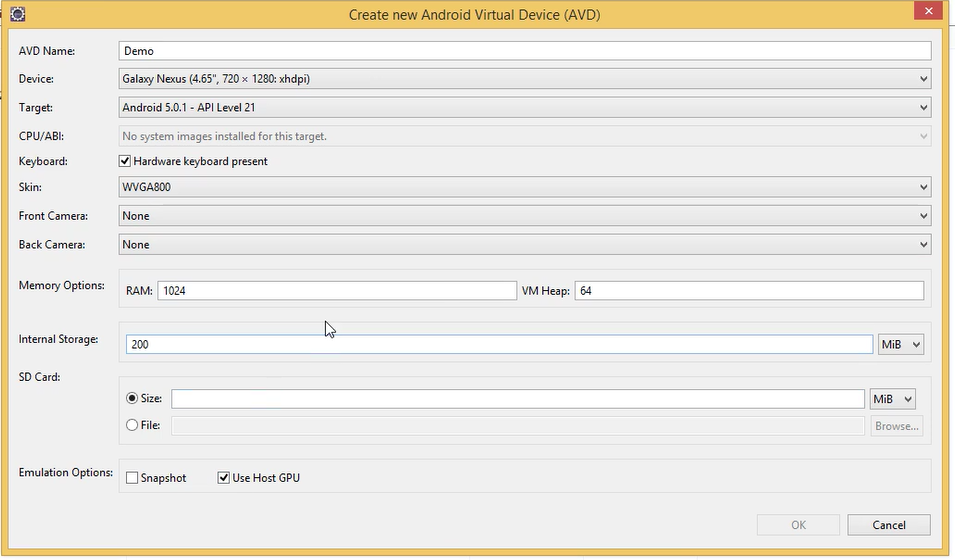
cap.setCapability(MobileCapabilityType.DEVICE\_NAME, "Android Emulator");

cap.setCapability(MobileCapabilityType.NEW\_COMMAND\_TIMEOUT, "60");

cap.setCapability(MobileCapabilityType.APP\_PACKAGE, <Package name>);

cap.setCapability(MobileCapabilityType.APP\_ACTIVITY, <Activity name>);

AndroidDriver driver = new AndroidDriver(new URL("http://127.0.0.1:4723/wd/hub"),cap);

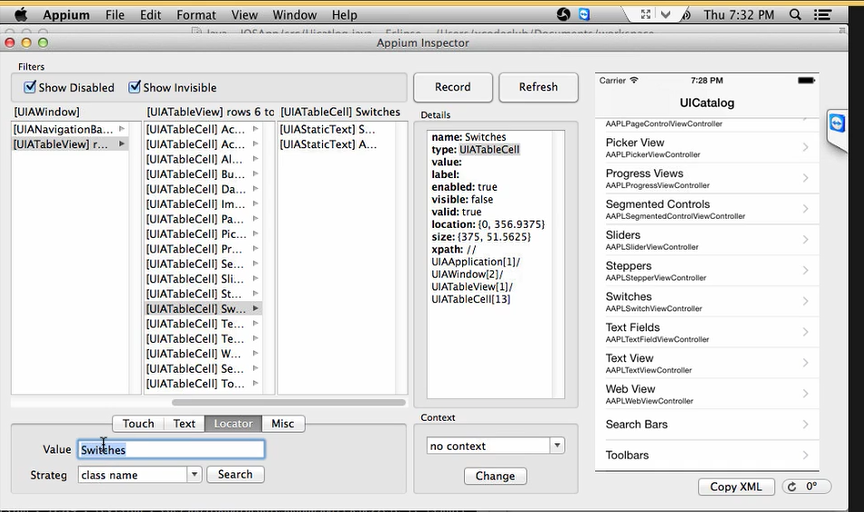


## IOS

XCode – IDE to support apple softwares. To get iOS Simulator go to Xcode -> window -> devices

Use Appium inspector to inspect the objects in iOS

iOS Settings can be used to define the desired capabilities and open the simulator to inspect the elements

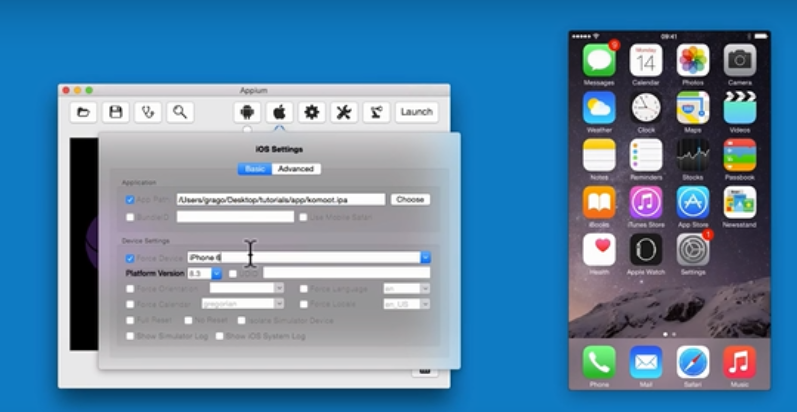


In locators tab at the bottom we can see the number of elements with a particular id or class name or xpath etc.

### Real Device

Settings -> Developer -> Enable UI Automation

Enter App Path, Check Force Device, Enter UDID

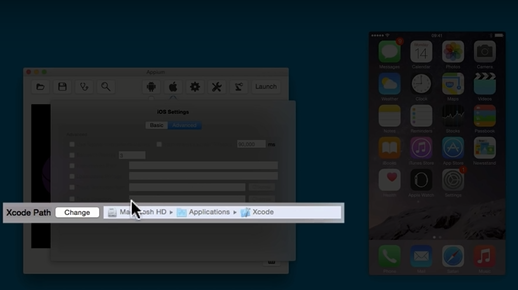


TO get UDID

XCode -> Devices -> Select the device

We will find UDID under identifier

Verify Xcode Path points to XCode installation path



Then click Launch

Once we get Status Code 200, we can start inspector. This will install the app on the phone

Once installed we see the screen shot of the device

## Mobile web

The UI maintained for mobile web and web browser are different.

We have to set only Browser name capability as we don’t need any app to be installed in web app.

Emulator doesn’t have chrome app by default. We have to install t. Use chrome above 33.

Android specific methods can’t be used. Only selenium methods can be used

User agent Switcher add on in firefox can be used to change the

Techpatterns.com/forum/about304.html – download xml user agent list and import it in the User Agent Switcher options window.

Then in the User agent Switcher select Mobile Devices -> OS -> Android -> Android version

## View Switching

Driver.getContext()

Driver.getContextHandles() – contexts supported

Driver.context

## ADB Commands

ADB – Android Debug Bridge

Adb start –server

Adb kill –server

Adb install <apk path>

Adb devices

adb logcat > <local\_path\_to\_text\_file> print the logs data to the screen for the purposes of bug reporting. We can also store these logs into local file

*adb -s <deviceName> <command>*

Direct an adb command a specific emulator/device instance whose deviceName is passed

*adb –d <command>*  
Directs adb command to the only attached ***USB device***.

*adb –e <command>*  
Directs adb command to the only attached ***Emulator***.

Adb commands from Java code:

public class TestCode {

public static void main(String[] args) throws Exception {

Runtime runtime = Runtime.getRuntime();

Process process = runtime.exec(new String[] {"/usr/bin/adb", "devices"});

}

}

In Android content description is Name

## Change Orientation

driver.rotate(ScreenOrientation.PORTRAIT);  
driver.rotate(ScreenOrientation.LANDSCAPE);

To get the current screen orientation value by this code  
ScreenOrientation orientation= driver.getOrientation();  
orientation.value()

# Crash log

### Getting Crash Logs Directly From a Device Without Xcode

Your users can retrieve crash reports from their device and send them to you via email by following these instructions.

(It is not possible to get device console logs directly from a device)

1) Open Settings app

2) Go to Privacy, then Diagnostics & Usage

3) Select Diagnostics & Usage Data

4) Locate the log for the crashed app. The logs will be named in the format: <AppName>\_<DateTime>\_<DeviceName>

5) Select the desired log. Then, using the text selection UI select the entire text of the log. Once the text is selected, tap Copy

6) Paste the copied text to Mail and send to an email address as desired

### Getting Crash Logs and Console Output From a Device Using Xcode

Even though you won't be able to run the app in Xcode's debugger, Xcode can still give you all the information you need to debug the problem.

#### Using Xcode 6

1) Plug in the device and open Xcode

2) Choose Window -> Devices from the menu bar

3) Under the DEVICES section in the left column, choose the device

4) To see the device console, click the up-triangle at the bottom left of the right hand panel

5) Click the down arrow on the bottom right to save the console as a file

6) To see crash logs, select the View Device Logs button under the Device Information section on the right hand panel

7) Find your app in the Process column and select the Crash log to see the contents.

8) To save a crash log, right click the entry on the left column and choose "Export Log"

9) Xcode 6 will also list low memory logs here. These will be shown with a Process name "Unknown" and Type "Unknown". You should examine the contents of these logs to determine whether any of these are caused by your app. For more information about low memory logs, see [Understanding and Analyzing iOS Application Crash Reports](http://developer.apple.com/library/ios/#technotes/tn2151/_index.html).

#### Using Xcode 5

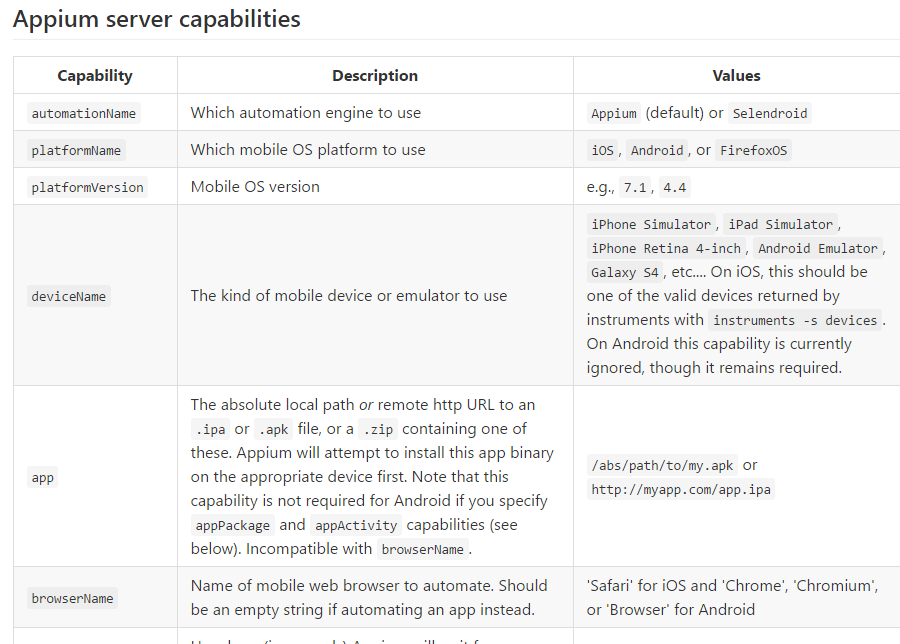
1) Plug in the device and open Xcode

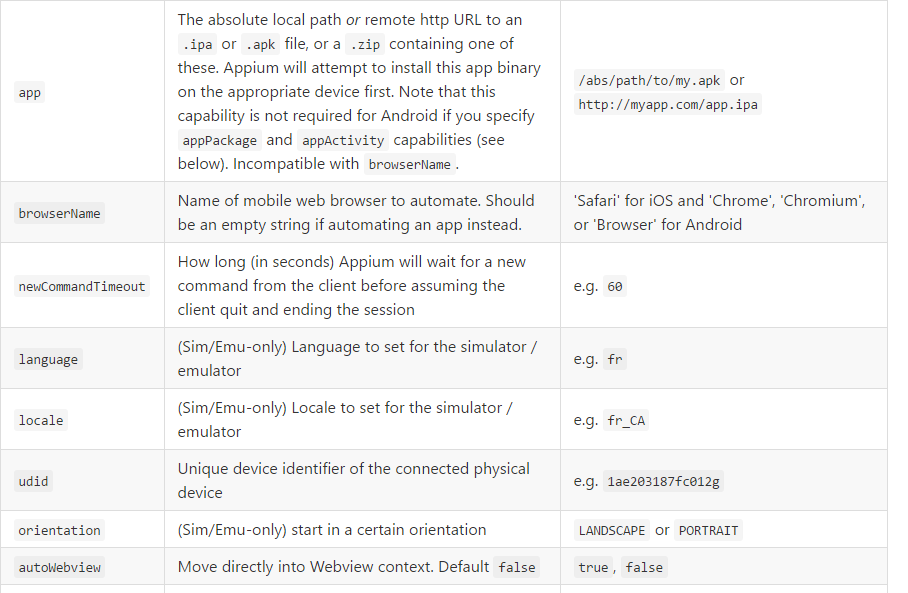
2) Open the Organizer window and select the Devices tab

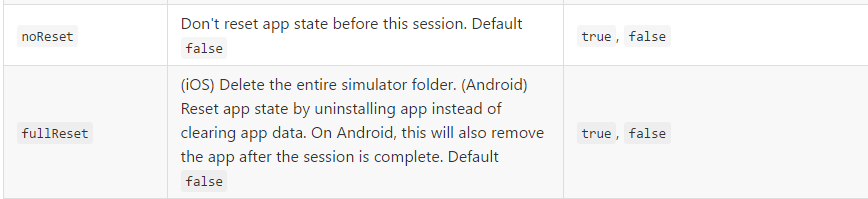
3) Under the DEVICES section in the left column, expand the listing for the device

4) Select Device Logs to see crash logs or select Console to see Console output

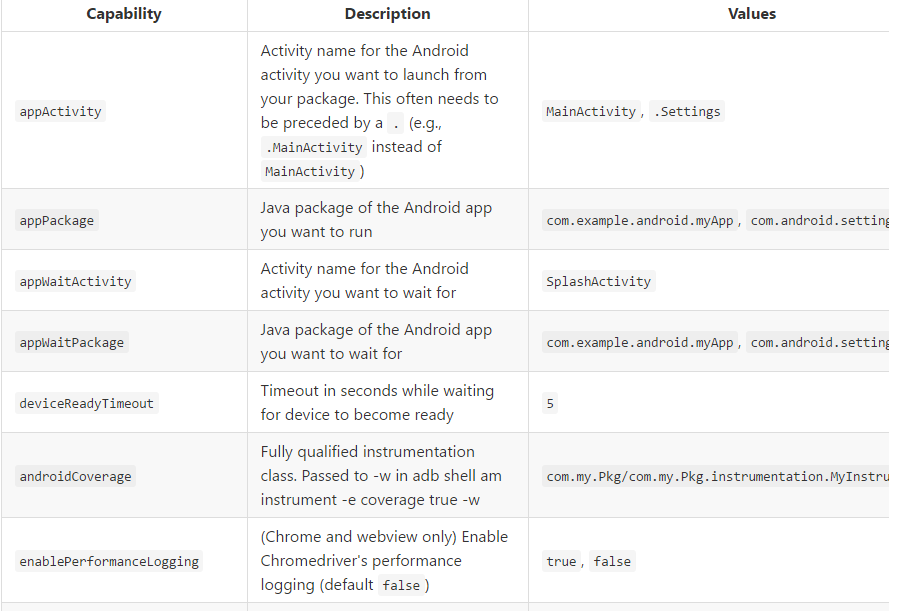
## Cap Cont.

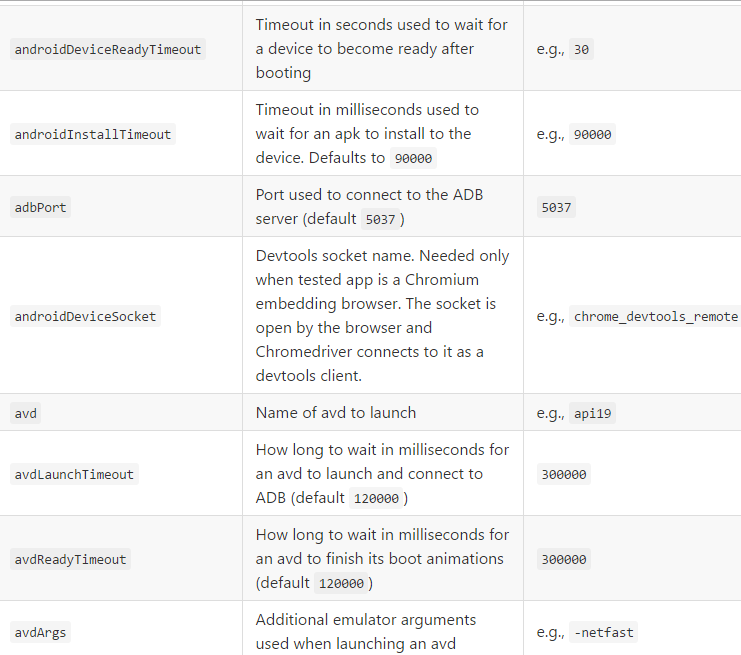


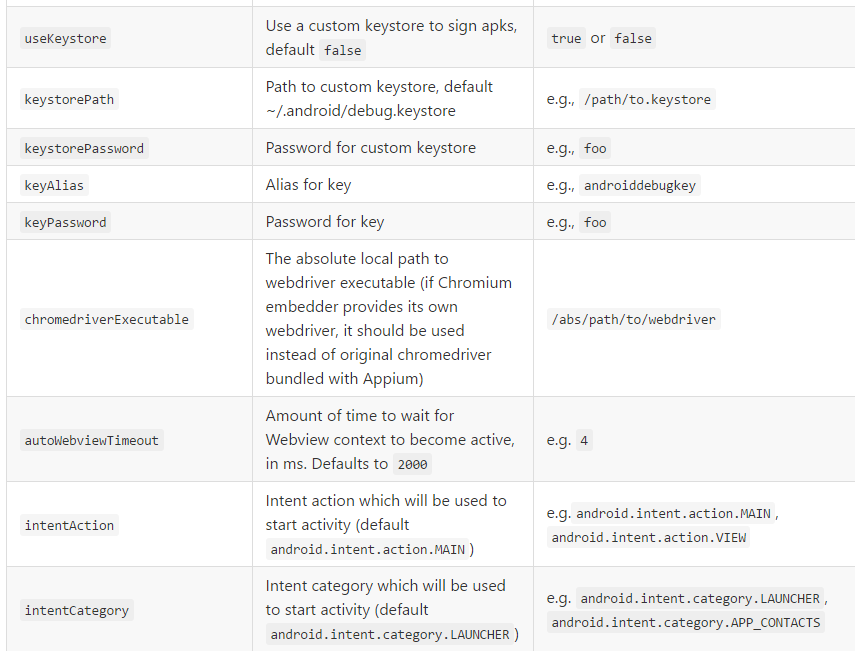




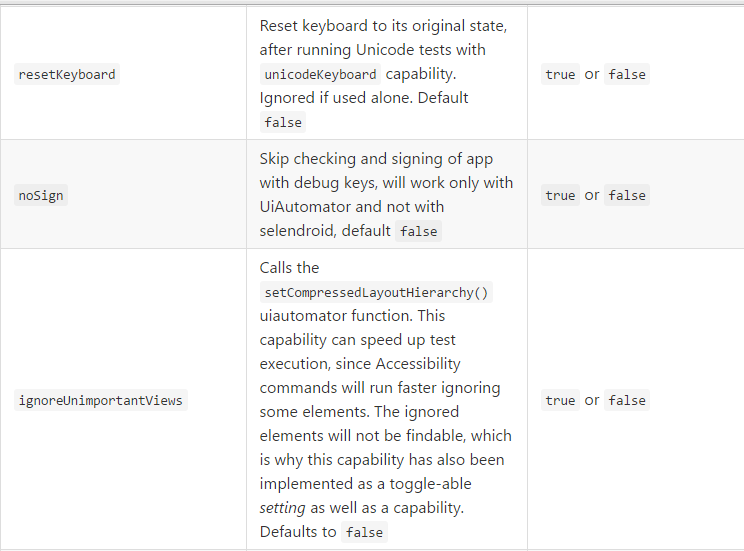
### Android Only Capabilities













### iOS Only

