**Mobile Automation Testing with Appium and Android Emulator**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.No | Version | Author | Date |
| 1 | 2.0 | Jayesh Meethaleveetil | Jun 16, 2016 |

**Contents**

* **Introduction**
* **Appium installation**
* **Android sdk & emulator configuration**
* **Starting an Android Emulator**
* **Automation test execution with Appium**
* **Appium Test Automation in Emulator Browser**

**Introduction**

Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS and Android platforms. Native apps are those written using the iOS or Android SDKs.Mobile web apps are web apps accessed using a mobile browser.

**Appium installation**

* Appium server - <http://appium.io/>
* Appium client libraries - <http://appium.io/downloads.html>

Appium has various languages associated with it like java, JavaScript, ruby,C# etc.

Here we go for java and download the *java-client.jar*.

* NodeJs - <https://nodejs.org/en/>

Kick up an Appium server, and then run a test written in your favorite [WebDriver](https://code.google.com/p/selenium/wiki/JsonWireProtocol)-compatible language! You can run an Appium server using node.js.

* APK file - appium server expects an apk file of the application to be tested

**Android sdk & emulator configuration**

1. Java SDK (minimum 1.6) must be installed and JAVA\_HOME configured.
2. Latest [Android-Sdk](http://developer.android.com/sdk/index.html) must be installed and ANDROID\_HOME set.

**System Requirements -Windows**

• Microsoft® Windows® 7/8/10 (32-bit or 64-bit)

• 2 GB RAM minimum

• 2 GB of available disk space minimum, 4 GB Recommended

• 1280 x 800 minimum screen resolutions

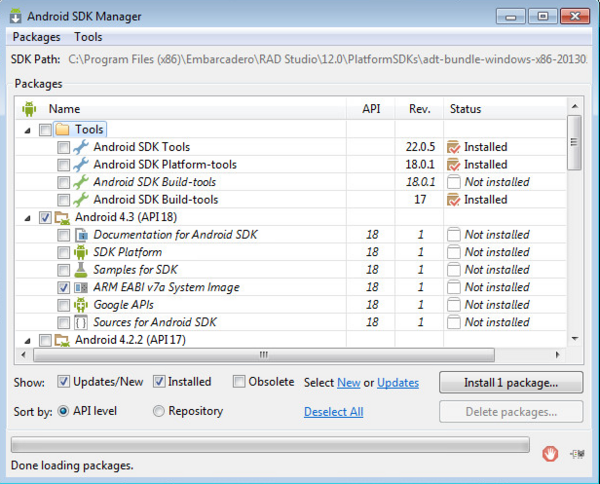
• Java Development Kit (JDK)

• For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VT-x

**Once the android sdk is installed follow the below steps**

1. Start the Android SDK
2. In the packages tree, locate and check the Android node in the list.

For example: Android 4.2.2 (API 17) or Android 4.3 (API 18)



1. Click Install 1 package. (Clear any checkboxes that were auto-selected.)
2. In the Choose Packages to install page, click Accept License, and then click Install to install the system image.

**Creating an Android Emulator Using the Android Virtual Device Manager**

**Emulator configuration**

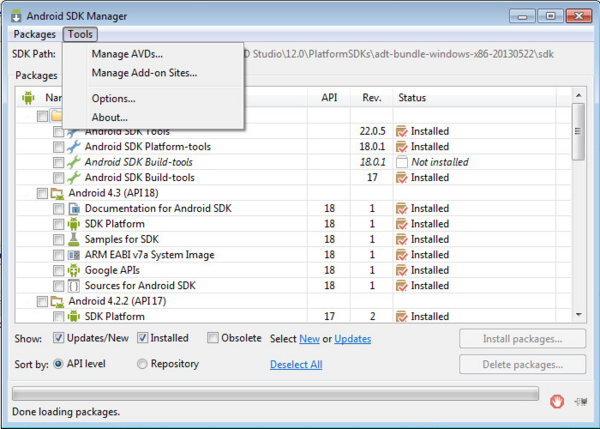
Appium cannot create new emulators; they must be created manually by the tester. After an emulator has been created, we recommend the first start be done manually in order to be sure everything works as expected.

When creating avds, please read the following configuration recommendations:

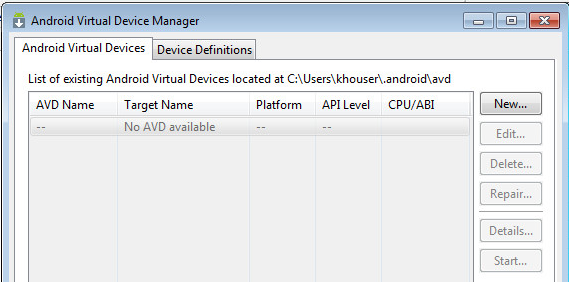
* Whenever possible, use the Intel x86 ABI
* Install [Intel x86 Emulator Accelerator](http://software.intel.com/en-us/android/articles/installation-instructions-for-intel-hardware-accelerated-execution-manager-windows) on Mac and Windows and use KVM on Linux to enable hardware acceleration to massively speed up the Emulator
* Use at least 1024MB of RAM per emulator
* Use at least 32MB of VM Heap per emulator
* Configure the hardware keyboard to be used: hw.keyboard=yes
* Use the Host GPU. If only a black screen is displayed, please deactivate this option.
* The number of emulators you can run in parallel per machine depends heavily on the hardware of the machine you use.

To create an Android emulator on your system, follow these steps:

1. Start the Android SDK Manager (select Start | All Programs | Embarcadero RAD Studio | Android Tools).
2. In the Android SDK Manager, select Tools | Manage AVDs.



1. In the Android Virtual Device Manager, click the new button to create a new virtual device.



1. In the Create new Android Virtual Device (AVD) dialog box, select an Android device to emulate, and enter the details describing the Android device you want to emulate:
   1. In Target, select an Android SDK with an API level that is 17 or higher. The drop-down list contains your installed versions of the Android SDK.
   2. Under Emulation Options, check Use Host GPU.
   3. In Device, select the Android device to emulate.

Tip: Emulating an older Android device like the Nexus S might be faster than emulating a newer, larger device like the Nexus 10.

1. Click OK to create your new Android emulator.
2. You can now view your emulator in the Android Virtual Device Manager.

## AVDwithEmulator.png

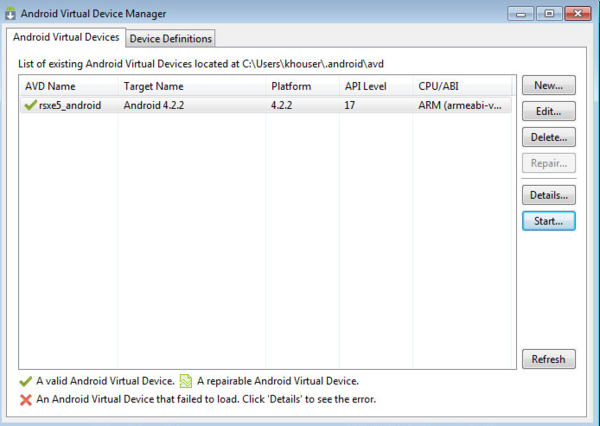
**Starting an Android Emulator**

When the emulator is the current target platform, you do not have to start the emulator before running your Android app on the emulator. However, we recommend that you start your Android emulator separately, perhaps before you start RAD Studio. Emulators are notoriously slow to start and to run, so starting the emulator beforehand means that your app can get to the emulator more quickly.

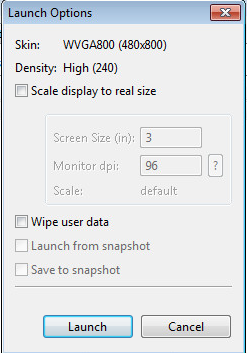
The recommended way to start an Android emulator is by using the Android Virtual Device Manager, which can be started from the Android SDK Manager.

To start an Android emulator such as the default emulator installed in RAD Studio:

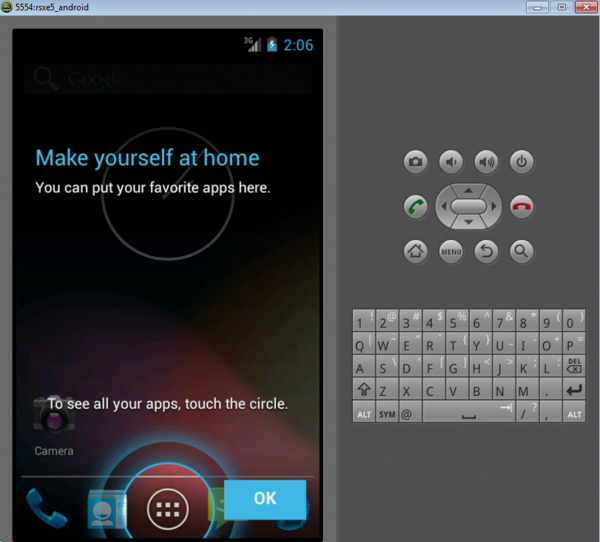
1. In the Android Virtual Device Manager, select the emulator and click Start.



1. Then click Launch on the Launch Options dialog box.

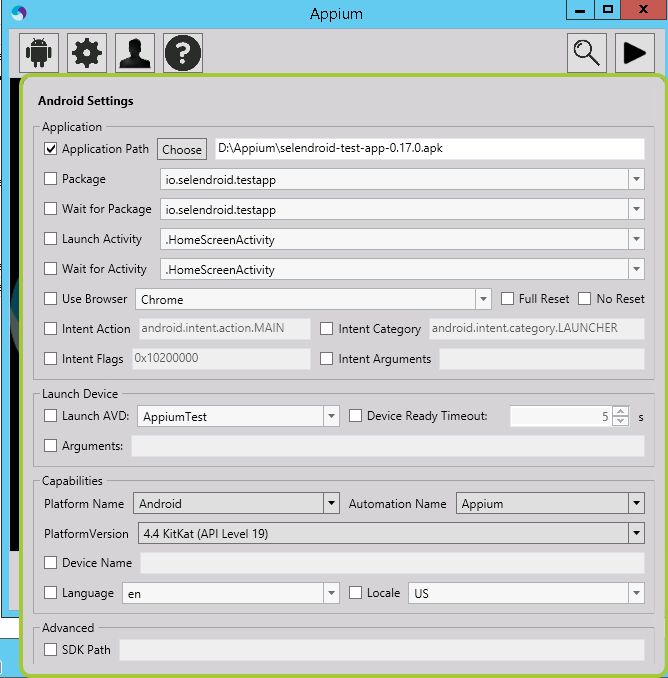


1. Your emulator should now start (this might take five or ten minutes):



**Automation test execution with Appium**

* Make sure Appium server is off.
* Click on Android ICON.
* Check the checkbox before application path
* Browse the .apk file which we downloaded at 1st step.
* Run the Appium server
* Click on Android ICON.
* Copy Package and Launch activity.



Sample program to automate test. This is just a small program just to make sure everything is up and running

import io.appium.java\_client.android.AndroidDriver;

import java.io.File;

import java.net.MalformedURLException;

import java.net.URL;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.remote.CapabilityType;

import org.openqa.selenium.remote.DesiredCapabilities;

public class StartApplication {

// create global variable

private static AndroidDriver driver;

public static void main(String[] args) throws MalformedURLException, InterruptedException

{

// Create object of DesiredCapabilities class

DesiredCapabilities capabilities = new DesiredCapabilities();

// Optional

capabilities.setCapability(CapabilityType.BROWSER\_NAME, "");

// Specify the device name (any name)

capabilities.setCapability("deviceName", "My New Phone");

// Platform version

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

// platform name

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

// specify the application package that we copied from appium

capabilities.setCapability("appPackage", "io.selendroid.testapp");

// specify the application activity that we copied from appium

capabilities.setCapability("appActivity", ".HomeScreenActivity");

// Start android driver I used 4727 port by default it will be 4723

driver = new AndroidDriver(new URL("http://127.0.0.1:4727/wd/hub"), capabilities);

// Specify the implicit wait of 5 second

driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

// Enter the text in textbox

driver.findElement(By.xpath("//android.widget.EditText[@content-desc='my\_text\_fieldCD']")).sendKeys("Selenium Appium demo testing");

// click on registration button

driver.findElement(By.id("io.selendroid.testapp:id/startUserRegistration")).click();

// Wait for 10 second

Thread.sleep(10000);

// close the application

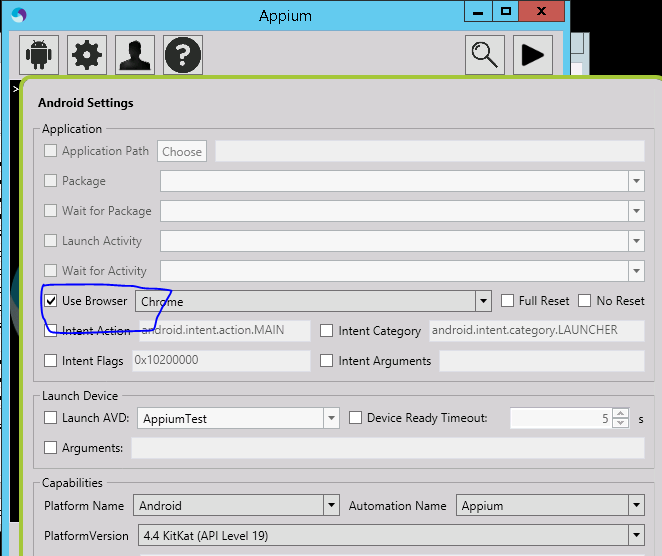
driver.quit();

}

}

**Appium Test Automation in Emulator Browser**

* We need to install the compatible chrome browser in the emulator
* We can download the chrome apk and can be installed with adb install <apkname.apk>
* Open Appium server
* Click on Android ICON.
* Select the option use browser
* Select the android target platforms
* Start the server and run the test in eclipse



**package** gmailDemo;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** io.appium.java\_client.android.AndroidDriver;

**import** io.appium.java\_client.remote.MobileCapabilityType;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.Platform;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.remote.BrowserType;

**import** org.openqa.selenium.remote.DesiredCapabilities;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.WebElement;

**import** org.openqa.selenium.support.ui.ExpectedConditions;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.support.ui.WebDriverWait;

**import** java.util.concurrent.\*;

**public** **class** StartGmail {

**public** **static** **void** main(String[] args) **throws** MalformedURLException, InterruptedException {

              // Create object of DesiredCapabilities class and specify android

              // platform

              DesiredCapabilities capabilities = DesiredCapabilities.*android*();

              // set the capability to execute test in chrome browser

              capabilities.setCapability(MobileCapabilityType.***BROWSER\_NAME***, BrowserType.***CHROME***);

              // set the capability to execute our test in Android Platform

              capabilities.setCapability(MobileCapabilityType.***PLATFORM***, Platform.***ANDROID***);

              // we need to define platform name

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

              // Set the device name as well (you can give any name)

              capabilities.setCapability(MobileCapabilityType.***DEVICE\_NAME***, "my phone");

              // set the android version as well

              capabilities.setCapability(MobileCapabilityType.***VERSION***, "4.4.2");

              // Create object of URL class and specify the appium server address

              URL url = **new** URL("<http://127.0.0.1:4723/wd/hub>");

              // Create object of AndroidDriver class and pass the url and capability

              // that we created

              WebDriver driver = **new** AndroidDriver(url, capabilities);

              // Open gmail

              driver.get("<http://www.gmail.com>");

              // Enter userd id

              WebElement element = driver.findElement(By.*id*("Email"));

              element.sendKeys("[abc@gmail.com](mailto:abc@gmail.com)");

              // wait 5 secs for userid to be entered

              driver.manage().timeouts().implicitlyWait(5, TimeUnit.***SECONDS***);

              // Submit button

              element.submit();

              // Enter Password

              WebElement element1 = driver.findElement(By.*id*("Passwd"));

              element1.sendKeys("abcpassword");

              // Submit button

              element1.submit();

              WebElement myDynamicElement = (**new** WebDriverWait(driver, 15))

                           .until(ExpectedConditions.*presenceOfElementLocated*(By.*id*("gbg4")));

              driver.findElement(By.*id*("gbg4")).click();

              // press signout button

              driver.findElement(By.*id*("gb\_71")).click();

              Thread.*sleep*(10000);

              // close the browser

              driver.quit();

       }

}