

## Executing Your Local Appium Code



### Overview

Appium is an open-source tool for automating mobile web, native, and hybrid applications on Android mobile, iOS mobile, and Windows desktops.

Appium is “cross-platform,” which means you can write tests for multiple platforms (iOS, Android, and Windows) using the same API. This allows for code reuse across iOS, Android, and Windows test suites.

With TestGrid, you can easily set up and test your mobile apps using your local Appium code for quicker results.

### Prerequisites

- TestGrid login credentials
- **Java Development Kit (JDK)**: Install the latest version of JDK on your machine. Appium is compatible with JDK 8 or later versions.
- **Integrated Development Environment (IDE)**: Choose an IDE to write and execute your Java Appium code. Popular choices include IntelliJ IDEA, Eclipse, or NetBeans. Make sure your IDE is properly installed and configured.
- **Appium Java Client**: Add the Appium Java Client dependency to your Java project. Using a build automation tool like **Maven or Gradle**, you can include the dependency. The Appium Java Client allows you to interact with the Appium server using Java code.
- There are client libraries in Java that support Appium’s WebDriver protocol extensions. You should use these client libraries instead of your regular WebDriver client when using Appium.

Start configuring your local Java or Python code right away.

### Step 1: Open the Device Cloud tab and locate the appropriate device information for running.

Login with TestGrid credentials and go to the Devices Cloud option tab.

The screenshot shows the TestGrid dashboard. On the left, there's a sidebar with links: Dashboard, Real Device Cloud (which is highlighted with a red box), Codeless + AI, API Testing, Test Management (with sub-links: Add Test Application, Test Runs (13)), Network Logs, More Tools (with sub-links: Docs), Latest build module status (tc01 SUCCESS), Bookmarks (8), and Notes (5).

The main area has a chart titled 'Test Cases' showing the number of test cases over time. The chart has three bars: one green bar for 'TOTAL SUCCESS' (17), one red bar for 'TOTAL FAIL' (1), and one blue bar for 'TOTAL STOP' (0). Below the chart is a timeline from 05/10 to 05/23, with a small red bar at the end of May 17.

A red arrow points from the 'Real Device Cloud' link in the sidebar to the 'Test Cases' chart area. Another red arrow points from the 'Test Grid Admin' dropdown menu to the 'TG-DeviceCloud' option.

Once the device cloud screen appears,

**iOS**

Connect Name	Model Name	Platform Version	Status	User	Action
iPhone 7 Dedicated	iPhone 7	iOS 15.7.2	Available	-	Connect
iPhone 11 Dedicated	iPhone 11	iOS 16.1.1	Disconnected	-	Connect

**Android**

Connect Name	Model Name	Platform Version	Status	User	Action
Samsung Galaxy A04 Dedicated	Samsung Galaxy A04	Android 13	Available	-	Connect
Redmi A1 Plus Dedicated	Redmi A1 Plus	Android 12	Available	-	Connect
Google Pixel 3 Dedicated	Google Pixel 3	Android 12	Available	-	Connect
Google Pixel 3 Dedicated	Google Pixel 3	Android 12	Disconnected	-	Connect
Samsung Galaxy A50 Dedicated	Samsung Galaxy A50	Android 11	Available	-	Connect
Samsung Galaxy M02 Dedicated	Samsung Galaxy M02	Android 11	Available	-	Connect
Samsung Galaxy M20 Dedicated	Samsung Galaxy M20	Android 10	Disconnected	-	Connect

**Web**

Browser name	Browser Type	Browser Version	Status	User	Action
Firefox 201	Firefox	92.0.1	Offline	-	Connect
Chrome 101	Chrome	94.0.4606.61	Offline	-	Connect

**Step 3: Select any iOS or Android device on which you want to run it. You will find the required device appium capabilities.**

**iOS**

Connect Name	Model Name	Platform Version	Status	User	Action
iPhone 7 Dedicated	iPhone 7	iOS 15.7.2	Available	-	Connect

Device ID: 5  
 Local Name: iPhone 7  
 Device Connect Name: iPhone 7  
 Screen Height: 667  
 Screen Width: 375  
 Pixel Density: 1.0  
 Appium Port: 36001  
 Appium URL: http://192.168.29.60:36001/wd/hub  
 MJpeg Port: 36301  
 WDA Local Port: 3601  
 Fake Device?: No  
 UDID: e4bf5b9175b1ed38ecf59ecb33112eb0c373ed4f

2

**Android**

Connect Name	Model Name	Platform Version	Status	User	Action
Samsung Galaxy A04 Dedicated	Samsung Galaxy A04	Android 13	Available	-	Connect

Notes: The below list of capabilities must be used for Android and iOS local execution with our device cloud.

```

1. # For Android
2.
3. {
4.   "appium:platformName": "Android",
5.   "appium:platformVersion": "12",
6.   "appium:deviceName": "Samsung Galaxy S10e",
7.   "appium:automationName": "UiAutomator2",
8.   "appium:udid": "R58M01147590X176B",
9.   "appium:systemPort": "37303"
10. }
11.
12. -----
13.
14. # For iOS
15.
16. {
17.   "appium:platformName": "iOS",
18.   "appium:platformVersion": "16.4",

```

```

23. "wdaLocalPort": "3606"
24. }

```

## Step 4: Steps to execute local Appium code

Obtain the run appium remote URL and device capabilities from TestGrid-Device Cloud.

The following variables need to be changed as provided for the organization and as per devices:

- TG\_DEVICE\_URL
- TG\_DEVICE\_NAME
- TG\_DEVICE\_UDID
- TG\_DEVICE\_PLATFORMNAME
- TG\_DEVICE\_PLATFORMVERSION
- TG\_DEVICE\_SYSTEM\_PORT (Android) \*
- TG\_WDA\_PORT (iOS) \*

```

1. // 1. Create an AppiumDriver
2. // 1.1 Set the capabilities of the driver
3. DesiredCapabilities capabilities = new DesiredCapabilities();
4. capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
5. capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, " < TG_DEVICE_NAME > ");
6. capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, " < TG_DEVICE_PLATFORMNAME > ");
7. capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, " < TG_DEVICE_PLATFORMVERSION > ");
8. capabilities.setCapability(AndroidMobileCapabilityType.DEVICE_UDID, " < TG_DEVICE_UDID > ");
9. capabilities.setCapability(AndroidMobileCapabilityType.systemPort, " < TG_SYSTEM_PORT > ");
10. capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, " ");
11. capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, " ");
12.
13. // Below the passed remote URL : TG_DEVICE_URL
14. driver = new AndroidDriver<MobileElement>(new URL("http://demo.testgrid.io:37001/wd/hub"), capabilities);
15. System.out.println("Created AppiumDriver");

```

**For example,** if you want to connect devices using the Appium Inspector tool,

The screenshot shows the Appium Inspector interface. At the top, it says "Appium Server" and "Select Cloud Providers". Below that, there are fields for "Remote Host" (Demo.testgrid.io), "Remote Port" (36001), "Remote Path" (/wd/hub), and an "SSL" checkbox. A "Advanced Settings" link is also present.

In the main area, there are tabs for "Desired Capabilities", "Saved Capability Sets 10", and "Attach to Session...". The "Desired Capabilities" tab is selected, displaying a table of key-value pairs:

appium:platformName	text	iOS
appium:platformVersion	text	16.4
appium:deviceName	text	iPhone 12
appium:udid	text	e577127e8ef534edssss0c
appium:bundleId	text	com.apple.mobilesafari
appium:automationName	text	XCUITest
wdaLocalPort	text	3606

A checkbox "Automatically add necessary Appium vendor prefixes on start" is checked. To the right of the table, there is a "JSON Representation" section containing the following JSON code:

```
{
  "appium:platformName": "iOS",
  "appium:platformVersion": "16.4",
  "appium:deviceName": "iPhone 12",
  "appium:udid": "e577127e8ef534edssss0c26ba12cdcf183c0d8ff6daa",
  "appium:bundleId": "com.apple.mobilesafari",
  "appium:automationName": "XCUITest",
  "wdaLocalPort": "3606"
}
```

At the bottom, there are links for "Desired Capabilities Documentation", "Save", "Save As...", and a large blue "Start Session" button.

To view the device inspector screen in Tools, click the "Start Session" button.

## Step 5: Execute your local Appium code. Below is the sample for Java & Python.

For an example Android-Appium using JAVA.

```
1. package com.sample.android;
```

```

6. import io.appium.java_client.remote.MobileCapabilityType;
7. import org.openqa.selenium.remote.DesiredCapabilities;
8. import org.testng.annotations.Test;
9.
10. import java.net.MalformedURLException;
11. import java.net.URL;
12. import java.util.concurrent.TimeUnit;
13.
14. public class MyFirstAppiumAndroidTest {
15.     public static AndroidDriver driver;
16.
17.     @Test
18.     public void runFirstAppiumTestAndroid () throws InterruptedException {
19.         try {
20.             // 1. Create a AppiumDriver
21.             // 1.1 Set the capabilities of the driver
22.             DesiredCapabilities capabilities = new DesiredCapabilities();
23.             capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
24.             capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, "< TG_DEVICE_NAME >");
25.             capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, "< TG_DEVICE_PLATFORMNAME >");
26.             capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, "< TG_DEVICE_PLATFORMVERSION >");
27.             capabilities.setCapability(AndroidMobileCapabilityType.DEVICE_udid, "< TG_DEVICE_UDID >");
28.             capabilities.setCapability(AndroidMobileCapabilityType.systemPORT, "< TG_SYSTEM_PORT >");
29.             capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, " ");
30.             capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, " ");
31.
32.             // Add here TestGrid Remote appium URL from Device cloud tab
33.             driver = new AndroidDriver<MobileElement> (new URL ("http://demo.testgrid.io:37001/wd/hub"), capabilities);
34.             System.out.println ("Created AppiumDriver Successfully");
35.             driver.manage ().timeouts ().implicitlyWait (30, TimeUnit.SECONDS);
36.
37.         } catch (MalformedURLException e) {
38.             e.printStackTrace ();
39.             throw new RuntimeException ("Error in creating Appium Driver");
40.         }
41.
42.         MobileElement elements = (MobileElement) driver.findElementByXPath("//body");
43.         System.out.println (elements);
44.         {
45.             if (elements.getText ().equals (elements)) {
46.                 elements.click ();
47.             }
48.         }
49.
50.         MobileElement element = (MobileElement) driver.findElementById("name");
51.         if (element.isDisplayed ()) {
52.             System.out.println (element);
53.             System.out.println ("Element Found!");
54.         } else {
55.             String pageSource = driver.getPageSource ();
56.             System.out.println (pageSource);
57.         }
58.         driver.quit();
59.     }
60. }

```

#### For an example iOS-Appium using JAVA

```

1. package com.test.ios;
2.
3. import io.appium.java_client.MobileElement;
4. import io.appium.java_client.ios.IOSDriver;
5. import org.apache.commons.io.FileUtils;
6. import org.openqa.selenium.OutputType;
7. import org.openqa.selenium.remote.DesiredCapabilities;
8. import java.io.File;
9. import java.io.IOException;
10. import java.net.MalformedURLException;
11. import java.net.URL;
12. import java.util.UUID;
13. import java.util.concurrent.TimeUnit;
14.
15. public class iOSTest {
16.     public static void main(String args[]) throws IOException {
17.         IOSDriver driver = null;
18.
19.         try {
20.             // -iOS Device capability as per metion TG device cloud
21.             DesiredCapabilities capabilities1 = new DesiredCapabilities ();
22.             capabilities1.setCapability ("platformVersion", "14.2");
23.             capabilities1.setCapability ("bundleId", " <Bundle_ID> ");
24.             capabilities1.setCapability ("deviceName", "iPhone 12 Pro Max");
25.             capabilities1.setCapability ("platformName", "iOS");
26.             capabilities1.setCapability ("automationName", "XCUITest");
27.             capabilities1.setCapability ("udid", "00008101-001870C01E");
28.             capabilities1.setCapability ("wdaPort", 3606);
29.
30.             // Change below remote URL as per device cloud
31.             driver = new IOSDriver<MobileElement> (new URL ("http://demo.testgrid.io:37001/wd/hub"), capabilities1);
32.             driver.manage ().timeouts ().implicitlyWait (30, TimeUnit.SECONDS);
33.
34.             Thread.sleep (3000);
35.
36.         } catch (MalformedURLException | InterruptedException e) {
37.             e.printStackTrace ();

```

```

42.         try {
43.             Thread.sleep (90000);
44.         } catch (InterruptedException e) {
45.             e.printStackTrace ();
46.             driver.getPageSource ();
47.         }
48.     }
49. }
```

#### For an example Python script for Android-Appium.

```

1.  from appium import webdriver
2.  from appium.webdriver.common.touch_action import TouchAction
3.  from appium.webdriver.common.mobileby import MobileBy
4.  from selenium.webdriver.support.ui import WebDriverWait
5.  from selenium.webdriver.support import expected_conditions as EC
6.
7. # Desired capabilities for the Android device
8. desired_caps = {
9.     "appium:platformName": "Android",
10.    "appium:platformVersion": "12",
11.    "appium:deviceName": "Samsung Galaxy S10e",
12.    "appium:automationName": "UiAutomator2",
13.    "appium:udid": "R58M90X178766B",
14.    "systemPort": "37303"
15. }
16.
17. # Appium server TestGrid Device Remote URL Here
18. appium_url = 'http://demo.testgrid.io:37001/wd/hub'
19.
20. # Initialize the driver
21. driver = webdriver.Remote(appium_url, desired_caps)
22.
23. # Wait for the app to load
24. wait = WebDriverWait(driver, 10)
25. app_loaded = wait.until(EC.presence_of_element_located((MobileBy.ID, 'com.example.app:id/mainLayout')))
26. assert app_loaded is not None
27.
28. # Perform actions on the app
29. element = driver.find_element(MobileBy.ID, 'com.example.app:id/button')
30. element.click()
31.
32. # Swipe from one element to another
33. element1 = driver.find_element(MobileBy.ID, 'com.example.app:id/element1')
34. element2 = driver.find_element(MobileBy.ID, 'com.example.app:id/element2')
35. action = TouchAction(driver)
36. action.press(element1).move_to(element2).release().perform()
37.
38. # Retrieve text from an element
39. text_element = driver.find_element(MobileBy.ID, 'com.example.app:id/textView')
40. text = text_element.text
41. print('Text:', text)
42.
43. # Close the app
44. driver.quit()
```

#### For an example Python script for iOS-Appium.

```

1.  from appium import webdriver
2.  from appium.webdriver.common.touch_action import TouchAction
3.  from appium.webdriver.common.mobileby import MobileBy
4.  from selenium.webdriver.support.ui import WebDriverWait
5.  from selenium.webdriver.support import expected_conditions as EC
6.
7. # Desired capabilities for the iOS device
8. desired_caps = {
9.     "appium:platformName": "iOS",
10.    "appium:platformVersion": "16.4",
11.    "appium:deviceName": "iPhone 8",
12.    "appium:udid": "e577127e8ef5383c0d8ff6daa",
13.    "appium:applicationName": "com.apple.mobilesafari",
14.    "appium:automationName": "XCUITest",
15.    "wdaLocalPort": "3606"
16. }
17.
18. # Appium server TestGrid Device Remote URL Here
19. appium_url = 'http://demo.testgrid.io:37001/wd/hub'
20.
21. # Initialize the driver
22. driver = webdriver.Remote(appium_url, desired_caps)
23.
24. # Wait for the app to load
25. wait = WebDriverWait(driver, 10)
26. app_loaded = wait.until(EC.presence_of_element_located((MobileBy.ID, 'com.example.app:id/mainLayout')))
27. assert app_loaded is not None
28.
29. # Perform actions on the app
30. element = driver.find_element(MobileBy.ID, 'com.example.app:id/button')
31. element.click()
32.
33. # Swipe from one element to another
34. element1 = driver.find_element(MobileBy.ID, 'com.example.app:id/element1')
35. element2 = driver.find_element(MobileBy.ID, 'com.example.app:id/element2')
36. action = TouchAction(driver)
```

```
41.     text = text_element.text
42.     print('Text:', text)
43.
44.     # Close the app
45.     driver.quit()
```

## Step 6: View live results on the TestGrid Device cloud.

Additionally, the remote execution of code can also be viewed live on the TestGrid Device Cloud.

As simple as that! Happy Testing 😊

### Additional Links

You can also do these with the TestGrid Platform : <https://testgrid.io/>

---

Tags: execute local appium code local appium code set up appium code

**Previous topic:** [Executing Your Local Selenium Code for Web Browsers](#)

**Next topic:** [Executing Appium Code from Windows Machine to Remote iOS Device](#)