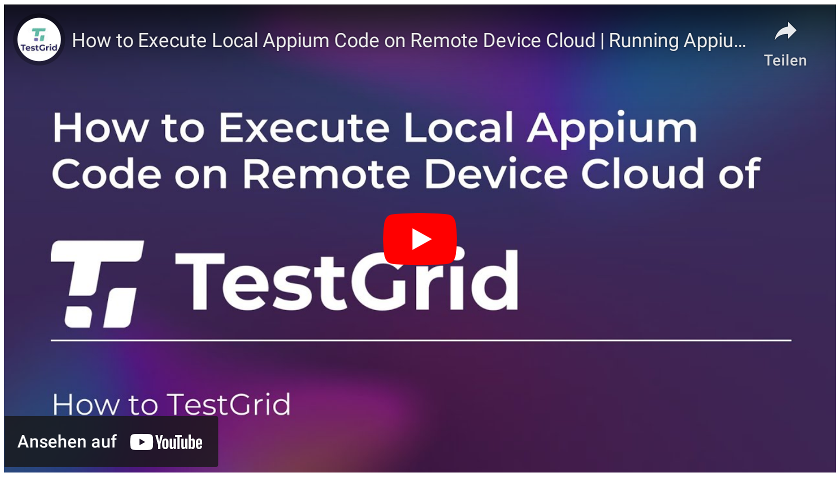
Executing Your Local Appium Code


How to Execute Local Appium Code on Remote Device Cloud | Running Appiu...
Teilen
How to Execute Local Appium Code on Remote Device Cloud of
TestGrid
How to TestGrid
Ansehen auf YouTube

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.

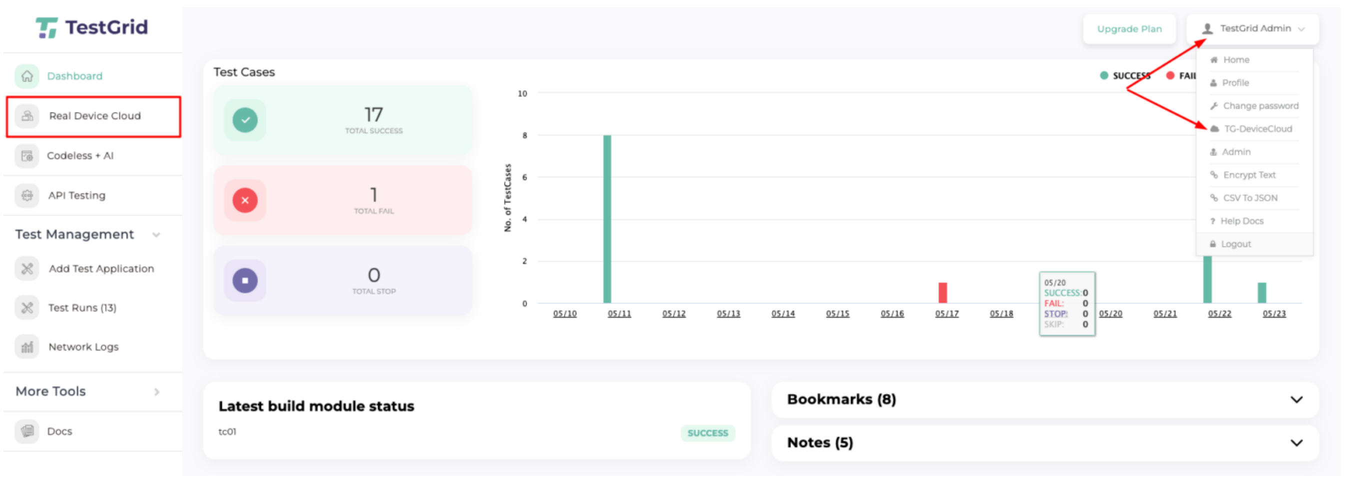
Prerequisites

- 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.

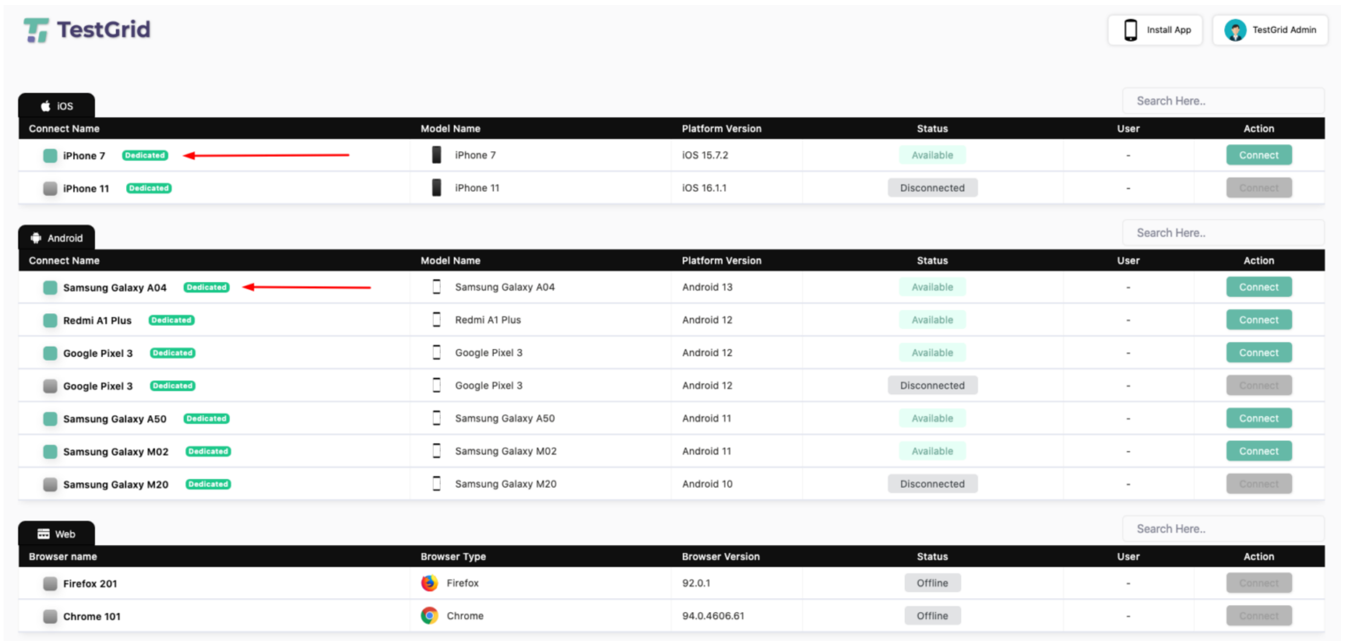
Start configuring your local Java or Python code right away.

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.



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

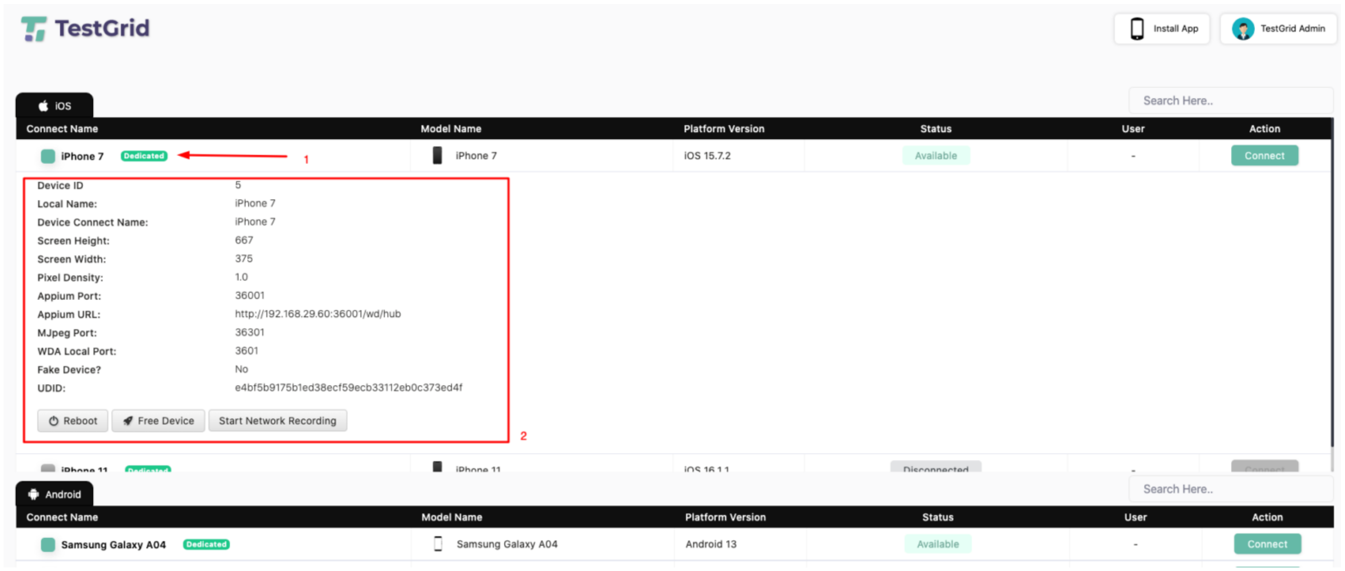
Android

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

Connect Name	Model Name	Platform Version	Status	User	Action
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

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

```
# For Android

{
"appium:platformName": "Android",
"appium:platformVersion": "12",
"appium:deviceName": "Samsung Galaxy S10e",
"appium:automationName": "UiAutomator2",
"appium:udid": "R58M01147590X176B",
"appium:systemPort": "37303"
}

 ----- ----- -----

# For iOS

{
"appium:platformName": "iOS",
"appium:platformVersion": "16.4",
"wdaLocalPort": "3606"
}
```

Step 4: Steps to execute local Appium code

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

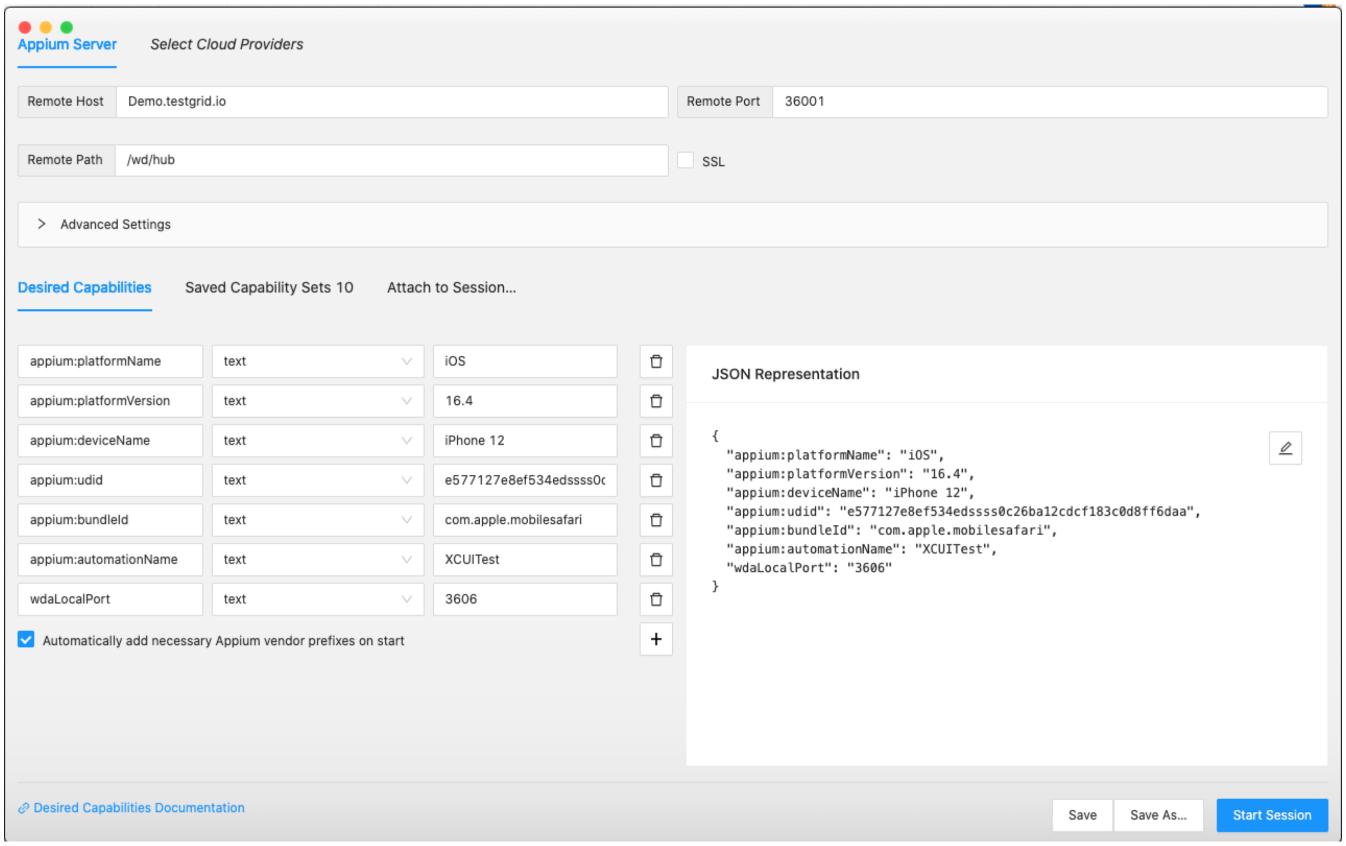
The following variables need to be changed as provided for the organization and as per devices:

- 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. Create an AppiumDriver
// 1.1 Set the capabilities of the driver
  DesiredCapabilities capabilities = new DesiredCapabilities();
  capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
  capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, " < TG_DEVICE_NAME > ");
  capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, " < TG_DEVICE_PLATFORMNAME > ");
  capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, " < TG_DEVICE_PLATFORMVERSION > ");
  capabilities.setCapability(AndroidMobileCapabilityType.DEVICE_UDID, " < TG_DEVICE_UDID > ");
  capabilities.setCapability(AndroidMobileCapabilityType.systemPort, " < TG_SYSTEm_PORT > ");
  capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, " ");
  capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, " ");

// Below the passed remote URL : TG_DEVICE_URL
  driver = new AndroidDriver<MobileElement>(new URL("http://demo.testgrid.io:37001/wd/hub"), capabilities);
  System.out.println("Created AppiumDriver");
```

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



Remote Host: Demo.testgrid.io
Remote Port: 36001
Remote Path: /wd/hub
SSL

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

[✓] Automatically add necessary Appium vendor prefixes on start

JSON Representation

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

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.

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

```java
import io.appium.java_client.remote.MobileCapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.testng.annotations.Test;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.concurrent.TimeUnit;

public class MyFirstAppiumAndroidTest {
    public static AndroidDriver driver;

    @Test
    public void runFirstAppiumTestAndroid () throws InterruptedException {
        try {
            // 1. Create a AppiumDriver
            // 1.1 Set the capabilities of the driver
            DesiredCapabilities capabilities = new DesiredCapabilities();
            capabilities.setCapability(MobileCapabilityType.AUTOMATION_NAME, "UiAutomator2");
            capabilities.setCapability(MobileCapabilityType.DEVICE_NAME, " < TG_DEVICE_NAME > ");
            capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, " < TG_DEVICE_PLATFORMNAME > ");
            capabilities.setCapability(MobileCapabilityType.PLATFORM_VERSION, " < TG_DEVICE_PLATFORMVERSION > ");
            capabilities.setCapability(AndroidMobileCapabilityType.DEVICE_udid, " < TG_DEVICE_UDID > ");
            capabilities.setCapability(AndroidMobileCapabilityType.systemPORT, " < TG_SYSTEM_PORT > ");
            capabilities.setCapability(AndroidMobileCapabilityType.APP_PACKAGE, " ");
            capabilities.setCapability(AndroidMobileCapabilityType.APP_ACTIVITY, " ");

            // Add here TestGrid Remote appium URL from Device cloud tab
            driver = new AndroidDriver<MobileElement> (new URL ("http://demo.testgrid.io:37001/wd/hub"), capabilities);
            System.out.println ("Created AppiumDriver Successfully");
            driver.manage ().timeouts ().implicitlyWait (30, TimeUnit.SECONDS);

        } catch (MalformedURLException e) {
            e.printStackTrace ();
            throw new RuntimeException ("Error in creating Appium Driver");
        }

        MobileElement elements = (MobileElement) driver.findElementByXPath("//body");
        System.out.println (elements);
        {
            if (elements.getText ().equals (elements)) {
                elements.click ();
            }
        }

        MobileElement element = (MobileElement) driver.findElementById("name");
        if (element.isDisplayed ()) {
            System.out.println (element);
            System.out.println ("Element Found!");
        } else {
            String pageSource = driver.getPageSource ();
            System.out.println (pageSource);
        }
        driver.quit();
    }
}
```

For an example iOS-Appium using JAVA

```java
package com.test.ios;

import io.appium.java_client.MobileElement;
import io.appium.java_client.ios.IOSDriver;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.UUID;
import java.util.concurrent.TimeUnit;

    public class iOSTest {
        public static void main(String args[]) throws IOException {
            IOSDriver driver = null;

            try {
                // -iOS Device capability as per metion TG device cloud
                DesiredCapabilities capabilities1 = new DesiredCapabilities ();
                capabilities1.setCapability ("platformVersion", "14.2");
                capabilities1.setCapability ("bundleId", " <Bundle_ID> ");
                capabilities1.setCapability ("deviceName", "iPhone 12 Pro Max");
                capabilities1.setCapability ("platformName", "iOS");
                capabilities1.setCapability ("automationName", "XCUITest");
                capabilities1.setCapability ("udid", "00008101-001870C01E");
                capabilities1.setCapability ("wdaPort", 3606);

                // Change below remote URL as per device cloud
                driver = new IOSDriver<MobileElement> (new URL ("http://demo.testgrid.io:37001/wd/hub"), capabilities1);
                driver.manage ().timeouts ().implicitlyWait (30, TimeUnit.SECONDS);

                Thread.sleep (3000);

            } catch (MalformedURLException | InterruptedException e) {
                e.printStackTrace ();
            try {
                Thread.sleep (90000);
            } catch (InterruptedException e) {
                e.printStackTrace ();
                driver.getPageSource ();
            }
        }
    }
```

For an example Python script for Android-Appium.

```python
from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction
from appium.webdriver.common.mobileby import MobileBy
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Desired capabilities for the Android device
desired_caps = {
  "appium:platformName": "Android",
  "appium:platformVersion": "12",
  "appium:deviceName": "Samsung Galaxy S10e",
  "appium:automationName": "UiAutomator2",
  "appium:udid": "R58M90X178766B",
  "systemPort": "37303"
}

# Appium server TestGrid Device Remote URL Here
appium_url = 'http://demo.testgrid.io:37001/wd/hub'

# Initialize the driver
driver = webdriver.Remote(appium_url, desired_caps)

# Wait for the app to load
wait = WebDriverWait(driver, 10)
app_loaded = wait.until(EC.presence_of_element_located((MobileBy.ID, 'com.example.app:id/mainLayout')))
assert app_loaded is not None

# Perform actions on the app
element = driver.find_element(MobileBy.ID, 'com.example.app:id/button')
element.click()

# Swipe from one element to another
element1 = driver.find_element(MobileBy.ID, 'com.example.app:id/element1')
element2 = driver.find_element(MobileBy.ID, 'com.example.app:id/element2')
action = TouchAction(driver)
action.press(element1).move_to(element2).release().perform()

# Retrieve text from an element
text_element = driver.find_element(MobileBy.ID, 'com.example.app:id/textView')
text = text_element.text
print('Text:', text)

# Close the app
driver.quit()
```

For an example Python script for iOS-Appium.

```python
from appium import webdriver
from appium.webdriver.common.touch_action import TouchAction
from appium.webdriver.common.mobileby import MobileBy
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Desired capabilities for the iOS device
desired_caps = {
  "appium:platformName": "iOS",
  "appium:platformVersion": "16.4",
  "appium:deviceName": "iPhone 8",
  "appium:udid": "e577127e8ef5383c0d8ff6daa",
  "appium:applicationName": "com.apple.mobilesafari",
  "appium:automationName": "XCUITest",
  "wdaLocalPort": "3606"
}

# Appium server TestGrid Device Remote URL Here
appium_url = 'http://demo.testgrid.io:37001/wd/hub'

# Initialize the driver
driver = webdriver.Remote(appium_url, desired_caps)

# Wait for the app to load
wait = WebDriverWait(driver, 10)
app_loaded = wait.until(EC.presence_of_element_located((MobileBy.ID, 'com.example.app:id/mainLayout')))
assert app_loaded is not None

# Perform actions on the app
element = driver.find_element(MobileBy.ID, 'com.example.app:id/button')
element.click()

# Swipe from one element to another
element1 = driver.find_element(MobileBy.ID, 'com.example.app:id/element1')
element2 = driver.find_element(MobileBy.ID, 'com.example.app:id/element2')
action = TouchAction(driver)
```

```
text = text_element.text
print('Text:', text)

# Close the app
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